UNIVERSITÉ DE PARIS 1 | PANTHEON-SORBONNE

THÈSE

pour obtenir

Le TITRE de DOCTEUR EN SCIENCES
SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

TITRE

ÉVALUATION DU BOOTSTRAP POUR LE CHOIX D'UN MODÈLE NEURONAL

LOCALISATION D'UN VÉHICULE EN MOUVEMENT SUR UNE VOIE ROUTIÈRE

par

Riadh KALLEL

Soutenue le : 29 novembre 2002

Jury :

Marie	Cottrell	*Directeur*
Luc	Jaulin	*Rapporteur*
Christian	Jutten	*Rapporteur*
Hichem	Maaref	*Examinateur*
Vincent	Vigneron	*Examinateur*
Jian-Feng	Yao	*Examinateur*

UNIVERSITÉ DE PARIS 1 | PANTHEON-SORBONNE

THÈSE

pour obtenir

Le TITRE de DOCTEUR EN SCIENCES
SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

TITRE

ÉVALUATION DU BOOTSTRAP POUR LE CHOIX D'UN MODÈLE NEURONAL

LOCALISATION D'UN VÉHICULE EN MOUVEMENT SUR UNE VOIE ROUTIÈRE

par

Riadh KALLEL

Soutenue le : 29 novembre 2002

Jury :

Marie	Cottrell	*Directeur*
Luc	Jaulin	*Rapporteur*
Christian	Jutten	*Rapporteur*
Hichem	Maaref	*Examinateur*
Vincent	Vigneron	*Examinateur*
Jian-Feng	Yao	*Examinateur*

Remerciements

Pour sa patience, pour son soutien permanent, pour son encadrement, pour tout ce qu'elle a fait pour le bon déroulement de cette thèse... je voudrais la remercier. Merci beaucoup Marie Cottrell.

Je remercie également Vincent Vigneron pour sa disponibilité, ses conseils et ses recommandations qui m'ont été très précieux.

Je remercie Luc Jaulin et Christian Jutten d'avoir accepté d'être les rapporteurs de cette thèse. Je les remercie pour leur lecture attentive de ce document et pour leurs remarques constructives. J'adresse aussi tous mes remerciements aux autres membres du jury.

Une grande pensée du fond du coeur à toute l'équipe du centre de recherche SAMOS qui m'a offert un cadre de travail merveilleux et motivant.

Je remercie aussi toute l'équipe du laboratoire LIVIC qui m'a si bien accueilli pendant un an, une grande reconnaissance à Morgan Mangeas qui m'a consacré un temps précieux.

Une grande pensée à mes parents, mon frère, mes deux soeurs et à tous mes amis. Enfin je remercie tous ceux qui, de près ou de loin ont participé à la réalisation de ce travail.

Table des matières

Introduction

Les phénomènes physiques peuvent être partagés en deux types :

- ceux qui sont déterministes, c'est-à-dire pour lesquels en partant des mêmes conditions initiales, on aboutit toujours au même résultat,
- ceux qui ne le sont pas, qu'on ne peut décrire exactement, même avec des formules mathématiques complexes, et pour lesquels avec les mêmes conditions initiales, on peut trouver des résultats différents. Cela peut être la conséquence d'une description incomplète du phénomène étudié, comme par exemple la relation entre le prix d'une action boursière et son prix futur. Mais cela peut aussi provenir d'une perturbation ou d'une erreur extérieure au phénomène, comme par exemple la lecture d'une mesure.

Dans ce mémoire de thèse, décomposé en deux parties distinctes pour des raisons de financement, on s'intéresse au deuxième type de phénomènes qui peut être vu comme une *relation stochastique* entre *cause* et *effet*. Dans la première partie (chapitres 1, 2 et 3), on essaie de décrire le mieux possible des phénomènes observés, dans le cas où la description complète est impossible. Dans la deuxième partie (chapitre 4 et 5), on essaie de contrôler les perturbations n'appartenant pas au phénomène étudié, et on applique les méthodes introduites à la localisation d'un véhicule en mouvement sur une voie.

Dans le langage scientifique, les relations *causes-effets* sont décrites par des *modèles*, les causes sont représentées par un vecteur $(X_1, \ldots, X_p)$ déterministe ou aléatoire, appelé *vecteur d'entrée*, et les effets par un vecteur $(Y_1, \ldots, Y_s)$, appelé *vecteur de sortie*. Un modèle s'écrit donc sous la forme :

$$(Y_1, \ldots, Y_s) = f(X_1, \ldots, X_p) + \epsilon.$$

La fonction f peut être paramétrique ou non, et ϵ est l'erreur ou le bruit du modèle. Trouver un modèle qui satisfasse cette égalité est par exemple équivalent à

trouver une fonction telle que l'erreur quadratique résiduelle soit minimale. Il existe de nombreuses méthodes qui proposent une solution à cette égalité, mais on peut aussi trouver de nombreux modèles satisfaisant une relation donnée.

Nous nous intéressons dans la première partie de ce mémoire à une famille de modèles non-paramétriques connus sous le nom de réseaux de neurones ou perceptrons multicouches (MLP). Dans le chapitre 1, nous rappelons l'aspect théorique de ces modèles et les résultats asymptotiques qui en découlent. Nous présentons aussi les avantages et les inconvénients de ces modèles. En dépit de leurs inconvénients, de nombreux logiciels ont été programmés pour l'utilisation des réseaux de neurones (SNNS, REGRESS...) et beaucoup de logiciels statistiques ou mathématiques comprennent désormais des modules qui les implémentent. Ces logiciels sont basés sur des approximations des résultats asymptotiques. Cependant, un des problèmes restant d'actualité, malgré la qualité des approximations et les performances de ces logiciels, est le problème du choix entre deux ou plusieurs modèles. Ce problème a reçu, comme on le verra plus loin, de nombreuses réponses, mais aucune n'est vraiment totalement satisfaisante. De plus, la plupart supposent que l'on dispose de données très nombreuses. Nous proposons donc d'utiliser les méthodes de ré-échantillonnage pour tenter de trouver une solution à ce genre de problème.

Une des méthodes de ré-échantillonnage est connue sous le nom de *bootstrap*[1]. Elle intéresse beaucoup de chercheurs pour des applications réelles dans le cas où on dispose d'un nombre de données qui n'est pas assez grand pour que les résultats asymptotiques soient valides. Elle a surtout été utilisée sur des modèles paramétriques. Elle permet d'évaluer le biais, la variance ou toute autre mesure de dispersion autour de l'estimé de la valeur d'un paramètre. D'où l'intérêt que nous portons à l'application de cette méthode aux modèles non-paramétriques et non-linéaires. Le paramètre qui nous intéresse dans le cas de la sélection de modèles est alors l'erreur quadratique résiduelle.

Bien que plusieurs auteurs aient eu l'idée de recourir au bootstrap pour la sélection de modèles, leurs travaux en sont restés aux préliminaires [79]. Notre contribution est ici de rapprocher ces deux domaines (MLP et bootstrap), en allant plus loin

[1] *Bootstrap* est le mot employé en anglais pour la languette de cuir cousue sur les bottes pour permettre de les enfiler, et l'expression "to put oneself up by one's own bootstrap" est traduite en français par "s'élever à la force du poignet". Le verbe bootstrap apparaît en anglais en 1951, et signifie "promouvoir ou développer une initiative par ses propres moyens". À peu près à la même époque apparaît le verbe "to boot a computer" en dérivation de l'adjectif bootstrap , qui signifie alors "capable d'utiliser une fonction ou processus interne pour en contrôler un autre". (Webster's Revised Unabridged Dictionary).

que [79] pour faire de la sélection de modèles (choix d'un modèle parmi plusieurs) et pour bâtir des tests d'un modèle par rapport à un autre qui le contient (sur-modèle).

La deuxième partie du chapitre 1, est consacré à rappeler la méthode du bootstrap non-paramétrique pour l'estimation de l'écart-type de l'estimateur d'un paramètre. Nous rappelons aussi les méthodes bootstrap utilisées pour la régression linéaire, et nous introduisons une deuxième méthode de ré-échantillonnage : le jackknife[2]. Ces rappels présentent les méthodes de ré-échantillonnage dans le but de les étendre au cas non-linéaire, en particulier à la famille de modèles de réseaux de neurones.

Les chapitres 2 et 3 constituent une partie de notre contribution originale. Dans le chapitre 2, nous présentons la méthodologie du bootstrap adaptée aux MLP, et nous montrons comment on peut proposer un critère permettant de choisir entre différents modèles. Nous présentons plusieurs applications simulées ou réelles. Une comparaison avec la méthode du jackknife est donnée à la fin du chapitre.

Dans le chapitre 3, une autre méthode de sélection de modèles sous forme de test entre deux hypothèses est étudiée. Nous nous intéressons en particulier aux modèles autorégressifs non-linéaires dont les MLP constituent un cas particulier. Nous rappelons les propriétés asymptotiques des modèles autorégressifs non-linéaires, et les tests asymptotiques de différence de contrastes[3] qui en découlent. Nous mettons en évidence les limites pratiques de l'application de ces tests à des données réelles ou simulées, puisqu'il faut disposer d'échantillons de très grande taille pour obtenir des résultats satisfaisants. Nous montrons alors comment on peut utiliser le bootstrap pour construire des tests beaucoup plus puissants, permettant le choix entre deux modèles. Enfin nous montrons la consistance du bootstrap paramétrique dans ce cas de figure.

Dans la deuxième partie de la thèse (chapitres 4 et 5) nous voulons traiter le problème de la localisation, en coordonnées géographiques, d'un véhicule en mouvement sur une voie routière à partir de capteurs embarqués. Le contrôle des sorties (la position du véhicule) se fait alors en tenant compte des erreurs des différentes variables d'entrée et de la différence de leurs fréquences. Il s'agit d'un problème de fusion de données.

[2] *The term "Jackknife" (qui signifie couteau de poche) was proposed by Tukey (1958) because, like the more traditional form of the jackknife, it is a tool that can perform useful work in many different situations.* John Porter (1998).

[3] On appelle fonction de contraste relative à un paramètre θ d'un modèle statistique, une fonction $\alpha \to k(\alpha, \theta)$ d'un compact Θ dans l'ensemble des réels positifs ayant un minimum strict pour $\alpha = \theta$.

La fusion de données consiste dans notre application en une opération d'intégration des données collectées par des capteurs en vue d'en extraire une nouvelle information plus représentative. Elle doit permettre de compenser les limites d'un capteur par les observations d'un ou de plusieurs autres. Elle doit aussi permettre d'exploiter à la fois les effets de redondance et de complémentarité de ces informations.

Les principales difficultés de cette étude viennent de la faible précision du calibrage et de la synchronisation entre les différents capteurs. Le niveau de précision requis pour la localisation ne peut être atteint qu'au prix d'un recalibrage préliminaire des capteurs, bien qu'ils aient été présentés par leurs constructeurs comme parfaitement calibrés. Un problème de synchronisation a donc été mis en évidence.

Dans le chapitre 4, nous détaillons le fonctionnement des différents capteurs utilisés. Nous introduisons la méthode du Filtre de Kalman classique puis étendu (Extended Kalman Filter, que nous désignerons dans la suite de façon abrégée *EKF*). Pour valider notre étude, nous avons utilisé l'EKF sur des jeux de données simulées pour la localisation d'un véhicule. Enfin nous avons appliqué la méthode EKF à des données réelles *calibrées et synchronisées* fournies par des capteurs embarqués à bord du véhicule.

Dans cette seconde partie (chapitre 5), notre principale contribution consiste en un algorithme de positionnement du véhicule à partir des informations ajoutées par 2 caméras embarquées, en utilisant les techniques dites de *l'analyse par intervalles*. Nous montrons que cette méthode peut résoudre des problèmes complexes de manière garantie, et nous l'appliquons au problème de la localisation d'un véhicule sur des données simulées.

Première partie

Évaluation du bootstrap pour le choix d'un modèle neuronal

Chapitre 1

Réseaux de neurones et bootstrap non-paramétrique

Pour étudier les relations *cause-effet*, on s'intéresse dans ce travail à des modèles statistiques comportant des termes aléatoires. Ces modèles permettent de bâtir une *relation* entre des variables *explicatives* et des variables *à expliquer* d'un échantillon statistique. Les modèles les plus classiques sont le plus souvent linéaires, mais se révèlent inadaptés dans beaucoup de situations concrètes. C'est dans cette optique que nous nous intéressons aux modèles de réseaux de neurones, qui sont non-linéaires et dont les propriétés sont maintenant assez bien connues.

Une fois qu'une relation entre les deux ensembles de variables est trouvée, un problème se pose : celui de la validité de cette relation pour de nouvelles données de même distribution que les données utilisées. Les statisticiens disposent pour cela de critères mathématiques à optimiser, par exemple le risque empirique [76], le risque bayésien [69]. Mais on peut aussi utiliser des méthodes de ré-échantillonnage, dont la plus connue est appelée *bootstrap*[1] par son auteur [30].

Le Bootstrap repose sur des techniques de simulation de plusieurs échantillons à partir d'un seul échantillon. Ce sont des procédures qui rendent possibles l'estimation de paramètres lorsque la complexité du problème ne permet pas d'inférence

[1] *Bootstrap* est le mot employé en anglais pour la languette de cuir cousue sur les bottes pour permettre de les enfiler, et l'expression "to put oneself up by one's own bootstrap" est traduite en français par "s'élever à la force du poignet". Le verbe bootstrap apparaît en anglais en 1951, et signifie "promouvoir ou développer une initiative par ses propres moyens". À peu près à la même époque apparaît le verbe "to boot a computer" en dérivation de l'adjectif bootstrap , qui signifie alors "capable d'utiliser une fonction ou processus interne pour en contrôler un autre". (Webster's Revised Unabridged Dictionary).

classique ou quand le nombre de données est insuffisant. Elles permettent d'éviter d'avoir à estimer la loi du bruit à partir des résidus, comme pour les approches de type Monte-Carlo qui exigent de choisir la loi utilisée pour générer des données fictives. Ces techniques permettent aussi d'estimer empiriquement la précision de l'estimation d'un paramètre.

Dans ce chapitre, nous introduisons la classe de modèles des réseaux de neurones en présentant leurs propriétés et leurs inconvénients. Ces notions sont maintenant classiques et peuvent être trouvées par exemple dans le livre de Hertz et al. [42]. Cependant, nous rappelons ici l'essentiel, dans un souci de clarté et de commodité pour le lecteur. Ensuite, nous définissons les techniques de bootstrap pour les modèles linéaires en vue de les étendre aux réseaux de neurones dans le chapitre 2.

1.1 Les Perceptrons MultiCouches (PMC)

1.1.1 Neurone formel

Le neurone formel introduit par Mc Culloch & Pitts [23], est un modèle mathématique reproduisant, en le schématisant, le fonctionnement du neurone biologique. Chacune des p connexions synaptiques d'entrées, appelées *poids*, est affectée d'une valeur $\theta_i, i = 1, \ldots, p$, qui pondère le signal d'entrée $x_i, i = 1, \ldots, p$. L'activité du neurone est évaluée en calculant successivement :

① son *potentiel*, c'est-à-dire la somme pondérée des p entrées, *i.e.* $\sum_{i=1}^{p} \theta_i x_i$,

② sa valeur de sortie ou *activité* : $\phi(\sum_{i=1}^{p} \theta_i x_i + \theta_0)$, à travers le filtre d'une fonction ϕ, où θ_0 est une constante.

La fonction ϕ est la *fonction de transfert*, ou la *fonction d'activation du neurone*, θ_0 est une constante appelée *biais*. Par convention, l'écriture peut être simplifiée en introduisant une entrée fictive $x_0 = 1$. La constante est alors considérée comme un poids de valeur θ_0 associé à l'entrée $x_0 = 1$. L'activité à la sortie du neurone s'écrit donc :

$$y = \phi(\sum_{i=0}^{p} \theta_i x_i). \tag{1.1}$$

Le choix de la fonction d'activation utilisée pour les neurones de Mc Culloch & Pitts [23] est la fonction *signe*, notée $sg : \mathbb{R} \longrightarrow \{-1, 1\}$, définie par :

$$sg(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{sinon.} \end{cases} \tag{1.2}$$

C'est une fonction non-linéaire, qui présente une discontinuité en zéro. Ce premier modèle de neurone formel s'appelle *neurone binaire* et il reste très schématique. Pour les applications pratiques et les possibilités de calcul, on préfère choisir des fonctions d'activation continues, dérivables et bornées sur $\mathbb{R}$, telles que la famille des fonctions sigmoïdes.

Les fonctions sigmoïdes sont paramétrées par 3 réels c, k, r tels que $c > 0, k > 0$ et peuvent s'écrire de façon générale :

$$\phi_{c,k,r}(x) = c\frac{e^{kx} - 1}{e^{kx} + 1} + r. \tag{1.3}$$

Elles sont croissantes sur $\mathbb{R}$, et ont deux asymptotes, $c + r$ en $+\infty$ et $r - c$ en $-\infty$. Pour $r = 0$ et $c = 1$, cette fonction s'écrit :

$$\phi_{1,k,0}(x) = \frac{e^{kx} - 1}{e^{kx} + 1} = \tanh(\frac{kx}{2}). \tag{1.4}$$

Sa dérivée en $x = 0$ vaut $k/2$. Et on montre facilement que la fonction $\phi_{1,k,0}(x)$ tend, lorsque $k \to \infty$, vers la fonction *signe*, et que le modèle de neurone formel associé tend vers le neurone binaire.

Un cas particulier de cette famille de fonctions, souvent utilisé, est la fonction tangente hyperbolique $\tanh(x) = \phi_{1,2,0}(x)$. On peut aussi prendre comme fonction d'activation la fonction identité, comme on le verra au paragraphe suivant.

Le neurone formel étant défini, on peut donc construire un *perceptron multi-couches*, qui est constitué d'un certain nombre de neurones formels, reliés par des connexions selon une architecture particulière.

1.1.2 Les perceptrons multicouches

Il s'agit du modèle le plus courant et le plus simple de réseau de neurones. C'est un réseau de neurones formels, constitué d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couches cachées.

Le nombre de neurones sur les couches d'entrée et de sortie dépend du problème examiné et du codage adopté. Le nombre de neurones sur les couches cachées est fixé par l'utilisateur. Le perceptron multicouches décrit Fig.(1.1), comporte p variables d'entrées $(X_1, \ldots, X_p)$, s variables de sorties $(Y_1, \ldots, Y_s)$, et une seule couche cachée avec H neurones. On note θ_{hj} le poids de la connexion de la j-ième entrée avec la h-ième unité cachée, et θ_{ih} le poids de la connexion de la h-ième unité cachée avec la i-ième sortie. Comme pour le neurone formel, les biais (ou constantes) sont les poids des connexions de chaque unité avec une entrée fictive fixée à 1. On note $\boldsymbol{\theta}$ le vecteur formé de l'ensemble des poids (il contient $H(p+s+1)+s$ paramètres), et par abus de langage on notera aussi $\boldsymbol{\theta}$ le réseau de neurones lui même.

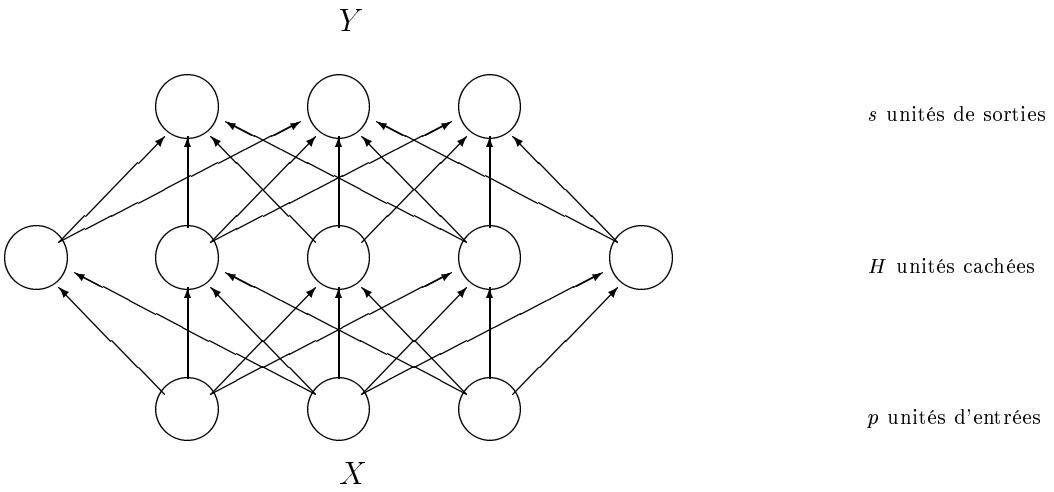


FIG. 1.1 – Perceptron multicouches muni d'une architecture $p - H - s$.

La fonction représentée par le réseau de neurones est définie de $\mathbb{R}^p$ dans $\mathbb{R}^s$, on la note $Y = f_{\boldsymbol{\theta}}(X)$, où $X = (X_1, \ldots, X_p)$ et $Y = (Y_1, \ldots, Y_s)$. C'est une combinaison des fonctions d'activation des différentes unités. Par exemple pour la i-ième sortie, la fonction s'écrit :

$$Y_i = \phi_i \left(\theta_{i0} + \sum_{h=1}^{H} \theta_{ih} \phi_h (\theta_{h0} + \sum_{j=1}^{p} \theta_{hj} X_j) \right), \quad 1 \leq i \leq s \tag{1.5}$$

où h décrit l'ensemble des unités cachées et j celles d'entrées. Les fonctions d'activation ϕ_i, ϕ_h peuvent être choisies identiques ou différentes. En général, les ϕ_h sont des fonctions sigmoïdes toutes identiques et les fonctions ϕ_i sont des fonctions identités, en particulier pour les applications que nous allons traiter. La fonction $f_{\boldsymbol{\theta}}(.)$ est donc une combinaison linéaire ou non-linéaire de fonctions sigmoïdes, sa complexité variant selon le nombre d'unités cachées et les valeurs des paramètres. Soulignons que l'équation (1.5) ne définit rien de plus qu'un modèle de régression linéaire ou non-linéaire paramétré par le vecteur $\boldsymbol{\theta}$.

De façon générale, pour un perceptron multicouches ayant un nombre quelconque de couches cachées, chaque neurone d'une couche est relié à tous les neurones de la couche suivante (il n'y a pas de connexions latérales à l'intérieur d'une couche, ni de connexions arrières).

1.1.3 Propriétés des perceptrons multicouches

Les perceptrons multicouches ont rapidement suscité l'intérêt des mathématiciens. Il a été ainsi démontré qu'un perceptron multicouches à une seule couche cachée, pourvue d'un nombre suffisant de neurones, peut approximer n'importe quelle fonction continue sur un compact de $\mathbb{R}^p$ avec la précision souhaitée. Cette propriété ne permet pas de choisir le nombre de neurones optimal dans la couche cachée, c'est-à-dire qu'elle ne mène pas à une technique de construction d'architecture.

Dans cette partie, on précise cette propriété, ainsi que d'autres qui ont contribué au succès du perceptron multicouches.

➤ **Approximateur universel** : Le problème de l'approximation de fonctions par des perceptrons multicouches a suscité une abondante littérature. Par exemple Cybenko [24], Barron [6], Hornik [43] ont étudié la propriété de l'approximation d'une fonction continue sur un compact par des perceptrons multicouches à une seule couche cachée. On reprend les résultats de Hornik [43] pour les perceptrons multicouches munis de fonctions d'activation non-linéaires et non-polynômiales.

Théorème 1 *: Hornik [43]*

Soit un modèle de perceptron multicouches défini par l'équation (Eq.(1.5)), où ϕ est une fonction d'activation strictement croissante et bornée, et où H est le nombre de neurones sur la couche cachée. Soit K un compact de $\mathbb{R}^p$, et $C(K, \mathbb{R}^s)$ l'ensemble des fonctions continues sur K à valeur dans $\mathbb{R}^s$. Alors :

$\forall f \in C(K, \mathbb{R}^s)$ et $\forall \epsilon > 0$, $\exists H \in \mathbb{N}^*$ et un vecteur de poids $\boldsymbol{\theta} \in \mathbb{R}^{H(p+s+1)+s}$, tel que $\forall X = (X_1, \ldots, X_p) \in K$:

$$||f(X) - f_{\boldsymbol{\theta}}(X)|| < \epsilon.$$

□

➤ **Vitesse de convergence de l'approximateur** : On dispose de très peu de résultats sur la vitesse d'approximation d'une fonction de régularité donnée, en fonction du nombre de paramètres ou du nombre de neurones sur la couche cachée. Nous reprenons les résultats de Attali & Pagés [1], établis pour une variable de sortie de dimension $s = 1$.

Théorème 2 *: Attali & Pagés [1]*

Soit K un compact de $\mathbb{R}^p$.
On pose $M_K = sup_{X \in K}||X||$ et $\delta_K = sup_{(X,Y) \in K^2}||X - Y||$. Soit $\phi \in C^\infty(\mathbb{R}, \mathbb{R})$ une fonction non-polynômiale telle que $\forall k \in \mathbb{N}$, $\phi^{(k)} \neq 0$. Soit f une fonction quelconque dont toutes les dérivées jusqu'à l'ordre p appartiennent à $C(K, \mathbb{R})$ et telle que $\forall i$, $1 \leq i \leq k$, $\frac{\partial f^{(k)}}{\partial X_i}$ soit ρ-lipschitz.
Soit $(\epsilon_H)_{H \geq 1}$ une suite de valeurs strictement positives, et convergente vers 0. Alors il existe une suite $(g_H)_{H \geq 0}$ de fonctions de perceptrons multicouches d'architecture $(p - H - 1)$ munis de fonctions d'activation ϕ associées aux H neurones de la couche cachée telle que :

$$||f - g_H|| \leq \rho A_p M_K^{p+1} \frac{1 + \epsilon_H}{H^{p+1}}$$

où A_p est une constante qui dépend uniquement de p.

□

Les bornes trouvées sont de l'ordre de $O(\frac{1}{H^{p+1}})$ pour une fonction continue sur un compact, ce qui implique un nombre de paramètres important dès lors que la dimension p des entrées est grande. Ce résultat peut s'appliquer pour les fonctions d'activation de la classe des fonctions sigmoïdes.

Les résultats de Barron [6] sont plus intéressants en terme de vitesse d'approximation, mais ils s'appliquent à une classe de fonctions très particulières. On suppose ici que l'on tente d'approximer la fonction f à l'aide d'une base finie de réalisations comprenant n individus.

Théorème 3 *: Barron [6]*

Soit K un compact de $\mathbb{R}^p$ et f une fonction continue sur K à valeurs dans $\mathbb{R}$. Soit g_H son estimateur issu de la classe des perceptrons multicouches comprenant H neurones sur la couche cachée. Soit n la taille de l'échantillon sur lequel on effectue l'estimation de f. Alors :

$$E(||f - g_H||^2) \leq O(\frac{C_f^2}{H}) + O(\frac{Hp}{n} \log n)$$

où C_f est défini comme une constante dépendant de f, appelée critère de complexité.

□

La valeur de C_f croît linéairement avec la dimension p, ce qui entraîne une croissance linéaire de la vitesse de convergence en fonction de p.

➤ **Résistance aux variables d'entrées aberrantes** : Dans le cas linéaire, une entrée aberrante donne automatiquement une sortie aberrante. Les fonctions sigmoïdes constituent des filtres saturants, ce qui limite ce genre d'inconvénient.

1.2 Apprentissage et validation d'un modèle neuronal

Dans ce qui suit, les perceptrons multicouches sont utilisés comme modèles de régression non-linéaire. On se restreint pour simplifier au cas où Y est une variable réelle de dimension $s = 1$.

Soit donc un perceptron multicouches ayant en entrée un vecteur de p variables explicatives $X = (X_1, \ldots, X_p)$, une seule couche cachée de H neurones et en sortie une variable à expliquer Y de dimension $s = 1$. On prend comme fonction d'activation de l'unité de sortie la fonction identité, notée ϕ_0. Les fonctions d'activation ϕ_h sont des fonctions sigmoïdes identiques pour tous les neurones de la couche cachée. On note $\boldsymbol{\theta}$ le perceptron correspondant.

Le modèle s'écrit alors (voir Eq.(1.5)) :

$$Y = \phi_0 \left(\theta_0 + \sum_{h=1}^{H} \theta_h \phi(\theta_{h0} + \sum_{j=1}^{p} \theta_{jh} X_j) \right) + \epsilon = f_{\boldsymbol{\theta}}(X) + \epsilon \qquad (1.6)$$

où ϵ est le résidu du modèle (de distribution normale ou non), de moyenne 0 et de variance σ^2. Le vecteur paramètre $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_h, \ldots, \theta_{h0}, \ldots, \theta_{jh})$ pour $1 \leq h \leq H$ et $1 \leq j \leq p$, est de dimension $(H(p+2)+1)$. On dispose d'un échantillon de taille n de variables indépendantes $(\boldsymbol{x}_i, y_i) = (x_{i1}, \ldots, x_{ip}, y_i)$ pour $1 \leq i \leq n$ de même loi que le vecteur (X, Y). On note $(\boldsymbol{x}, y) = (x_1, \ldots, x_p, y) = (x_{ij}, y_i)_{1 \leq i \leq n, 1 \leq j \leq p}$ la matrice des données.

Pour estimer le vecteur paramètre $\boldsymbol{\theta}$, on minimise, comme il est classique en statistique, une fonction *coût*, en général quadratique, donnée par :

$$MSE(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^2 . \tag{1.7}$$

Dans le domaine des réseaux de neurones, cette étape s'appelle *apprentissage* du réseau. La minimisation de la fonction coût se fait par des techniques d'optimisation classiques (1er ou 2nd ordre). On rencontre plusieurs difficultés pour bien estimer le modèle.

- **Non-unicité du modèle par rapport aux paramètres :** On peut trouver deux jeux de poids différents qui générent la même sortie :
  par exemple ceci peut être obtenu en permutant l'ordre des neurones de la couche cachée.

- **Sur-paramétrisation :** Ces modèles sont en général sur-paramétrés. Utiliser un modèle comportant un trop grand nombre de paramètres peut entraîner un *surajustement*. Dans ce cas, on sous-estime la variance σ^2 par une somme des carrés résiduelle anormalement faible. L'ajustement des données disponibles sera artificiellement *presque parfait*, mais sans que cela corresponde à une véritable qualité du modèle. La qualité de *généralisation* (calcul de la sortie associée à de nouvelles données) risque d'être très insuffisante.

- **Existence de minima locaux :** A la suite de l'apprentissage, la solution trouvée est en général un minimum local de la fonction coût. Tous les algorithmes déterministes ou stochastiques basés sur le gradient se stabilisent sur des minima locaux. En pratique, des conditions initiales différentes conduisent à des solutions différentes.

Assurer une bonne convergence, évaluer un modèle est donc un problème difficile. De nombreuses études ont été réalisées pour résoudre ces problèmes.

➤ **Méthodes d'élagage :** Le Cun [56] a proposé la méthode *Optimal Brain Damage : (OBD)*. Cette méthode s'appuie sur une évaluation de l'importance de chaque poids du réseau par le calcul des dérivées secondes de la fonction coût. Une autre méthode a été proposée par Cottrell et al. [21], c'est la méthode *Statistical Stepwise Method : (SSM)* (Mangeas [61], Rynkiewicz [71]). C'est une technique d'élagage basée sur un test statistique de nullité des paramètres du réseau. Ces deux méthodes permettent de diminuer le sur-paramétrage (et donc le risque de sur-apprentissage).

➤ **Méthodes constructives :** Elles consistent à faire croître le réseau à partir d'une configuration initiale. Les premières approches sont *tiling algorithm* par Mézard et Nadel [65], et *Upstart algorithm* par Frean [33]. D'autres approches ont vu le jour et beaucoup d'applications et algorithmes concrétisent ces approches, citons par exemple Lengellé et Denoeux [58], Jutten [19] ...

➤ **Méthodes pour obtenir des minima locaux de qualité la meilleure possible :** Citons par exemple l'utilisation du gradient du second ordre (Hassibi et Stork [41], Battiti [7]). Cette méthode améliore la qualité des minima locaux, mais elle ne résout pas parfaitement le problème. Citons aussi les méthodes du recuit simulé (Siarry et Dreyfus [73]) et les algorithmes génétiques (Goldberg [38], Davis [25], Jones [48]). Mais ces méthodes demandent des temps de calcul extrêmement longs.

➤ **Méthode de validation sur une base de test :** Classiquement, lorsque les données sont suffisamment abondantes (n assez grand), la première étape consiste à partager l'ensemble des observations en deux sous-ensembles : une *base d'apprentissage* et une *base de test*. La base d'apprentissage avec m observations $m < n$, est utilisée pour estimer le vecteur paramètre $\boldsymbol{\theta}$ des poids du modèle en minimisant la fonction coût (Eq.(1.7)).

Au terme de l'apprentissage, on obtient un estimateur $\hat{\boldsymbol{\theta}}$ de $\boldsymbol{\theta}$, et une erreur d'apprentissage :

$$MSE_a(\hat{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i)\right)^2 . \tag{1.8}$$

Le modèle résultant après l'apprentissage est ensuite testé sur la base de test. Un bon modèle généralisera bien sûr des exemples qui ne lui ont jamais été présentés. L'erreur de test est définie par :

$$MSE_t(\hat{\boldsymbol{\theta}}) = \frac{1}{n-m} \sum_{i=m+1}^{n} \left(y_i - f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i)\right)^2 . \tag{1.9}$$

Ces méthodes ont montré leur efficacité pour beaucoup d'applications dans des domaines très variés. Néanmoins, on n'a pas une définition exacte du critère d'arrêt de l'apprentissage. De plus, le découpage des données en deux sous-ensembles n'est pas toujours pertinent, il faut être sûr qu'il n'y a pas de changement de distribution entre la base d'apprentissage et la base de test. Par ailleurs, on n'a pas toujours des données assez nombreuses.

Une autre méthode statistique de validation croisée a fait preuve de son efficacité pour évaluer la qualité des estimateurs des paramètres d'un modèle. Cette méthode est basée sur le *ré-échantillonnage*, et elle est appelée *Bootstrap* par son auteur Efron [30]. Elle a surtout été utilisée sur des modèles paramétriques pour calculer les estimateurs et déterminer leurs précisions.

Dans le paragraphe suivant, on rappelle comment utiliser le bootstrap dans le cadre classique de l'estimation d'un paramètre, et on énonce ses propriétés dans le cadre du modèle linéaire, avant de les étendre au cas non-linéaire dans le chapitre 2.

1.3 Bootstrap : Estimation de l'écart-type de l'estimateur d'un paramètre

Le bootstrap, développé par Efron [30], a une place privilégée dans les nouvelles méthodes de *ré-échantillonnage*. C'est une technique d'estimation non paramétrique qui permet d'évaluer le biais, la variance ou toute autre mesure de dispersion autour de l'estimé de la valeur d'un vecteur de paramètres. On considère ici des variables aléatoires réelles ($p = 1$).

Soit $\boldsymbol{x} = (x_1, \ldots, x_n)$ un échantillon de n variables aléatoires indépendantes et de même loi que la variable réelle X. La variable X a une distribution F inconnue et dépendant d'un vecteur paramètre $\boldsymbol{\theta}$. Le problème consiste à estimer $\boldsymbol{\theta}$ par une statistique $\boldsymbol{\theta} = s(\boldsymbol{x})$. La question qui se pose est donc la précision de cet estimateur, étant donné que F est inconnue.

Dans le but d'évaluer cette précision, on réplique B échantillons à partir de l'échantillon initial $\boldsymbol{x}$, par *ré-échantillonnage*. Ces échantillons sont appelés *échan-*

tillons bootstrappés et sont notés $\boldsymbol{x}^{*b}$.

Un *échantillon bootstrappé* (dit aussi *base bootstrapée*) $\boldsymbol{x}^{*b} = (x_1^{*b}, \ldots, x_n^{*b})$ est construit par un tirage aléatoire uniforme et avec remise sur les composantes de l'échantillon $\boldsymbol{x}$:

$$P_U(x_i^{*b} = x_j) = \frac{1}{n}; \qquad (i, j = 1, \ldots, n) \tag{1.10}$$

où P_U est la loi uniforme sur les données originales $\boldsymbol{x} = (x_1, \ldots, x_n)$. La fonction de distribution de l'échantillon bootstrappé $\boldsymbol{x}^{*b}$ est $\hat{F}$, *i.e.* la distribution empirique de la variable X. Une réplication bootstrap de l'estimateur $\hat{\boldsymbol{\theta}} = s(\boldsymbol{x})$ est donnée par $\hat{\boldsymbol{\theta}}^{*b} = s(\boldsymbol{x}^{*b})$. Par exemple, pour la moyenne de l'échantillon $\boldsymbol{x}$, l'estimateur est $s(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$ et une *replication bootstrap* de cet estimateur sera $s(\boldsymbol{x}^{*b}) = \frac{1}{n}\sum_{i=1}^{n} x_i^{*b}$.

Ainsi, l'estimateur bootstrap $\hat{\sigma}_{boot}(\hat{\boldsymbol{\theta}})$ de l'écart-type de $\hat{\boldsymbol{\theta}}$ est calculé par :

$$\hat{\sigma}_{boot} = \left[\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\boldsymbol{\theta}}^{*b} - \overline{\hat{\boldsymbol{\theta}}}^{*}\right)^2\right]^{\frac{1}{2}} \tag{1.11}$$

où

$$\overline{\hat{\boldsymbol{\theta}}}^{*} = \frac{1}{B}\sum_{b=1}^{B}\hat{\boldsymbol{\theta}}^{*b} \tag{1.12}$$

En conjonction avec ces procédures de ré-échantillonnage, des tests d'hypothèses et des régions de confiance peuvent être construits.

1.4 Bootstrap pour la régression linéaire

Dans cette partie, on considère le modèle linéaire défini par :

$$y_i = \boldsymbol{x}_i\boldsymbol{\theta} + \epsilon_i \quad 1 \leq i \leq n \tag{1.13}$$

où les ϵ_i sont des variables aléatoires indépendantes, centrées, de variance σ^2, et de même distribution inconnue (normale ou non). Les $\boldsymbol{x}_i$ sont des vecteurs déterministes de dimension p, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ est le vecteur paramètre inconnu, de dimension p, et les y_i pour $1 \leq i \leq n$ sont des observations indépendantes. On note $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ la matrice $n \times p$, et $y = (y_1, \ldots, y_n)^T$ le vecteur de taille n.

Distribution empirique	Échantillons bootstrappés	Réplications bootstrap de $\hat{\boldsymbol{\theta}}$
	$\boldsymbol{x}^{*1}$	$\hat{\boldsymbol{\theta}}^{*1} = s(\boldsymbol{x}^{*1})$
	$\boldsymbol{x}^{*2}$	$\hat{\boldsymbol{\theta}}^{*2} = s(\boldsymbol{x}^{*2})$
	$\boldsymbol{x}^{*3}$	$\hat{\boldsymbol{\theta}}^{*3} = s(\boldsymbol{x}^{*3})$
$\hat{F}$	$\vdots$	$\vdots$
	$\boldsymbol{x}^{*B}$	$\hat{\boldsymbol{\theta}}^{*B} = s(\boldsymbol{x}^{*B})$

L'estimateur bootstrap
de l'écart-type

$$\hat{\sigma}_{boot} = \left[\frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\boldsymbol{\theta}}^{*b} - \overline{\hat{\boldsymbol{\theta}}}^{*} \right)^2 \right]^{\frac{1}{2}}$$

avec

$$\overline{\hat{\boldsymbol{\theta}}}^{*} = \frac{1}{B} \sum_{b=1}^{B} \hat{\boldsymbol{\theta}}^{*b}$$

TAB. 1.1 – Algorithme Bootstrap pour le calcul de l'estimateur bootstrap de l'écart-type. Le nombre B de réplications est généralement compris entre 25 et 200.

On suppose que la matrice $\boldsymbol{x}$ est de rang p. L'estimateur de $\boldsymbol{\theta}$ par la méthode des moindres carrés est donné par :

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T y. \tag{1.14}$$

Rappelons que cet estimateur est sans biais, que

$$Var(\hat{\boldsymbol{\theta}}) = \sigma^2(\boldsymbol{x}^T\boldsymbol{x})^{-1}, \tag{1.15}$$

et que si la matrice $(\boldsymbol{x}^T\boldsymbol{x})^{-1}$ converge vers 0, quand $n \longrightarrow +\infty$, alors $\hat{\boldsymbol{\theta}}$ converge presque sûrement vers le vecteur paramètre $\boldsymbol{\theta}$ quand n tend vers $+\infty$:

$$\hat{\boldsymbol{\theta}} \xrightarrow{p.s.} \boldsymbol{\theta}. \tag{1.16}$$

On note $\hat{\epsilon} = y - \boldsymbol{x}\hat{\boldsymbol{\theta}}$ le vecteur résidu estimé. L'estimateur de σ^2 est donné par :

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i\hat{\boldsymbol{\theta}})^2 = \frac{1}{n-p}\sum_{i=1}^{n}\hat{\epsilon}_i^2 \tag{1.17}$$

où $\hat{\epsilon}_i = y_i - \boldsymbol{x}_i\hat{\boldsymbol{\theta}}$ est la i-ième composante du vecteur résidu $\hat{\epsilon}$.

Si de plus, la suite des matrices $\frac{1}{n}\boldsymbol{x}^T\boldsymbol{x}$ converge vers une matrice définie positive V quand $n \longrightarrow +\infty$, alors $n^{\frac{1}{2}}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})$ est asymptotiquement normal de moyenne 0 et de matrice de variance-covariance σ^2V^{-1}, et l'estimateur $\hat{\sigma}^2$ de σ^2 est fortement consistant :

$$\hat{\sigma}^2 \xrightarrow{p.s.} \sigma^2. \tag{1.18}$$

Ces propriétés asymptotiques permettent de calculer des intervalles de confiance par l'application du théorème central limite. Dans le cas où ce théorème ne s'applique pas, faute d'un nombre suffisant d'observations, on a recours à des méthodes de ré-échantillonnage pour améliorer l'estimateur du vecteur paramètre, et calculer des intervalles de confiance.

Dans le cadre du modèle linéaire avec variables explicatives déterministes Eq.(1.13), Efron et Tibshirani [30] proposent de construire les bases bootstrappées à partir du vecteur résidu estimé : c'est le *bootstrap résiduel* ([34] [11] [74] [32] ...).

Mais dans les applications réelles, les variables explicatives sont plutôt aléatoires et le modèle à considérer est du type :

$$Y = X\boldsymbol{\theta} + \epsilon \tag{1.19}$$

où ϵ est le résidu centré du modèle, de distribution inconnue (normale ou non) et de variance σ^2, $X = (X_1, \ldots, X_p)$ est un vecteur de p variables aléatoires explicatives, et Y est une variable aléatoire réelle.

Soit $(\boldsymbol{x}, y) = (\boldsymbol{x}_{ij}, y_i)_{1 \leq i \leq n, 1 \leq j \leq p}$ un échantillon de taille n de variables indépendantes de même loi que le vecteur $(X, Y) = (X_1, \ldots, X_p, Y)$, tel que $E(||(\boldsymbol{x}_i, y_i)||^4) < +\infty$, où $(\boldsymbol{x}_i, y_i)$ est la i-ème ligne de l'échantillon pour $1 \leq i \leq n$. Dans ce cas, on estime encore $\boldsymbol{\theta}$ par :

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{x}^T \boldsymbol{x})^{-1} \boldsymbol{x}^T y, \tag{1.20}$$

et sous certaines hypothèses sur les moments, on obtient encore des propriétés de normalité asymptotique de $\hat{\boldsymbol{\theta}}$.

Dans le cas de ce modèle linéaire à variables explicatives aléatoires, Efron et Tibshirani [30] proposent de construire les bases bootstrappées à partir des couples $(\boldsymbol{x}_i, y_i)$, c'est ce qu'on appelle le *bootstrap par paires*.

La construction d'une base bootstrappée par le bootstrap par paires, s'effectue de la façon suivante :

① On fait un tirage aléatoire uniforme et avec remise sur les lignes $(\boldsymbol{x}_i, y_i)$ de l'échantillon $(\boldsymbol{x}, y)$. Soit $(\boldsymbol{x}^*, y^*) = (\boldsymbol{x}_i^*, y_i^*)_{1 \leq i \leq n}$ la base bootstrappée obtenue.

② On estime à partir de cette base le vecteur de paramètres $\boldsymbol{\theta}$ par la méthode des moindres carrés Eq.(1.20), soit

$$\hat{\boldsymbol{\theta}}^* = ({\boldsymbol{x}^*}^T \boldsymbol{x}^*)^{-1} {\boldsymbol{x}^*}^T y^*.$$

Un estimateur de la variance du vecteur résidu est donné par l'équation (1.17) :

$${\hat{\sigma}^*}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i^* - \boldsymbol{x}_i^* \hat{\boldsymbol{\theta}}^*)^2.$$

Dans ce cadre, Freedman [34] a montré que l'estimateur bootstrap, basé sur un échantillon bootstrappé, a des propriétés asymptotiques similaires à celles de l'estimateur $\hat{\boldsymbol{\theta}}$.

Dans la suite, puisque nous considérons des variables explicatives a priori aléatoires, nous adoptons la méthode du bootstrap par paires en l'étendant à des modèles

non-linéaires comme les perceptrons multicouches. À titre de comparaison, nous définissons dans la section suivante une alternative au bootstrap, très utilisée comme méthode de ré-échantillonnage, le jackknife[2].

Il s'agit en fait d'une sorte de cas particulier du bootstrap, plus léger à mettre en oeuvre.

1.5 Le jackknife ou la méthode leave-one-out

Le jackknife est une technique d'estimation du biais ou de l'erreur quadratique moyenne. Cette méthode ressemble à celle du bootstrap dans la technique de calcul. Nous donnons une brève définition de cette méthode dans cette partie. Pour plus de détails, voir le chapitre 11 du livre d'Efron [31].

On considère ici $p = 1$ pour simplifier, et on suppose qu'on dispose d'un échantillon de n variables réelles indépendantes x_i

$$\boldsymbol{x} = (x_1, \ldots, x_n)$$

de même loi qu'une variable aléatoire réelle X, de fonction de densité f dépendant d'un vecteur paramètre $\boldsymbol{\theta}$ inconnu. Soit $\hat{\boldsymbol{\theta}} = s(\boldsymbol{x})$ un estimateur du paramètre $\boldsymbol{\theta}$. On cherche à estimer l'erreur quadratique de cet estimateur.

Le jackknife est basé sur la construction de n bases de taille $n-1$, obtenues en enlevant une composante à la fois de l'échantillon initial, d'où l'autre nom de cette méthode "leave-one-out". Soit donc le vecteur colonne

$$\boldsymbol{x}_{(i)} = (x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n)$$

la i-ième réplication jackknife. L'estimateur de l'erreur quadratique moyenne de $\hat{\boldsymbol{\theta}} = s(\boldsymbol{x})$ calculé par cette méthode est donné par :

$$\hat{\sigma}_{loo} = \left(\frac{1}{n} \sum_i (\hat{\boldsymbol{\theta}}_{(i)} - \overline{\hat{\boldsymbol{\theta}}})^2 \right)^{\frac{1}{2}}$$

où

$$\hat{\boldsymbol{\theta}}_{(i)} = s(\boldsymbol{x}_{(i)})$$

[2] *The term "Jackknife" (qui signifie couteau de poche) was proposed by Tukey (1958) because, like the more traditional form of the jackknife, it is a tool that can perform useful work in many different situations.* John Porter (1998).

et

$$\overline{\hat{\boldsymbol{\theta}}} = \sum_i \hat{\boldsymbol{\theta}}_{(i)}.$$

Comme dans le cas de l'estimateur de l'erreur quadratique moyenne par la méthode du bootstrap, la formule de $\hat{\sigma}_{loo}$ est vue comme une erreur quadratique des n valeurs de l'estimateur de $\boldsymbol{\theta}$ calculées sur chaque réplication jackknife.

Nous nous servons dans le chapitre 2, de la technique du jackknife (extrêmement utilisée dans les applications) pour la comparer avec la méthode de sélection de modèles que nous proposons dans le même chapitre.

1.6 Conclusion

Après avoir rappelé ci-dessus les définitions du bootstrap dans le cas des modèles linéaires, nous proposons d'étendre la même démarche au cas non-linéaire des réseaux de neurones, pour résoudre le problème de la sélection de modèles. En effet, comme on l'a vu (voir le paragraphe (1.2)), les modèles de réseaux de neurones sont en général surparamétrés, les paramètres n'ont pas de véritable interprétation concrète, et le problème qui se pose est plutôt celui de choisir précisement un modèle, c'est-à-dire une architecture : le nombre d'entrées, le nombre d'unités et de couches cachées.

Dans ce cadre (non-linéaire et non-paramétrique) les résultats théoriques sont incomplets, mais les techniques du bootstrap nous servent de guide. On verra que les performances du "bootstrap étendu" restent très intéressantes, et permettent de proposer des solutions pratiques et pas trop coûteuses.

Chapitre 2

Application du bootstrap à la sélection de modèles

Nous avons vu dans le chapitre précédent les inconvénients de l'apprentissage d'un modèle neuronal ainsi que quelques méthodes pour les surmonter. Ces méthodes, bien qu'elles soient performantes, restent problématiques pour toute application réelle, soit parce qu'on manque de données, soit parce qu'on a des difficultés à localiser le minimum global, soit pour les deux raisons. Nous avons vu aussi comment utiliser la méthode du bootstrap pour évaluer la qualité d'un estimateur dans le cas d'un modèle linéaire. Nous nous inspirons de ces principes pour appliquer le bootstrap à la sélection de modèles de perceptrons multicouches.

Notre contribution consiste ici à proposer une nouvelle méthode permettant de choisir un modèle de perceptron multicouches parmi une famille de perceptrons multicouches. Cette méthode présente l'avantage, contrairement aux méthode OBD ou SSM, section 1.2, qu'il n'est pas indispensable de disposer d'un très grand nombre de données.

Dans ce chapitre, nous adaptons donc la technique du bootstrap par paires aux modèles non linéaires, et nous montrons comment se servir de cette méthode pour choisir un modèle. Nous présentons deux exemples simulés et une application à des données réelles. Nous comparons à la fin du chapitre, le bootstrap et le jackknife sur les mêmes exemples.

2.1 Méthodologie

Pour la simplicité de l'écriture, notons $\mathcal{B}_0$ un échantillon (appelé aussi *base initiale*) de taille n de variables indépendantes de même loi que le vecteur $(X, Y) = (X_1, \ldots, X_p, Y)$:

$$\mathcal{B}_0 = \{(\boldsymbol{x}_i, y_i);\ \ 1 \leq i \leq n\}, \tag{2.1}$$

où $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$ est la i-ième ligne de la matrice $(n \times p)$ $\boldsymbol{x} = (x_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$, et

$$Y = f_{\boldsymbol{\theta}}(X) + \epsilon \tag{2.2}$$

le modèle à étudier. La fonction $f_{\boldsymbol{\theta}}$ peut être linéaire, comme dans le cas de l'équation (1.19), ou non-linéaire comme dans le cas des perceptrons multicouches (1.6). Le terme ϵ est centré, de variance σ^2, gaussien ou non.

À partir des données de l'échantillon $\mathcal{B}_0$, on génère B échantillons bootstrappés

$$\mathcal{B}^{*b} = \{(\boldsymbol{x}_i^{*b}, y_i^{*b});\ \ 1 \leq i \leq n\}$$

pour $b = 1, \ldots, B$, par la méthode du bootstrap par paires expliquée dans le paragraphe (1.4). Une fois les B bases bootstrappées obtenues, on fait l'apprentissage sur chacune d'elles pour estimer le vecteur paramètre $\boldsymbol{\theta}$. Cette estimation se fait par la méthode des moindres carrés pour un modèle linéaire, et par la méthode de la rétropropagation du gradient [70], ou par n'importe quel algorithme d'optimisation pour les perceptrons multicouches (voir le paragraphe (1.2)). Notons $\hat{\boldsymbol{\theta}}^{*b}$ cet estimateur, on obtient ainsi B réalisations $\hat{\boldsymbol{\theta}}^{*b}$ et B vecteurs résidus d'apprentissage $\hat{\epsilon}_a^{*b} = y^{*b} - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x}^{*b})$ (voir Eq.(1.8), dans le cas linéaire, ou Eq.(1.17) dans le cas des perceptrons) pour chaque modèle Eq.(2.2). Une erreur d'apprentissage est donc calculée pour chaque jeu de données $\mathcal{B}^{*b}$, c'est l'estimateur $\widehat{\sigma_a^2}^{*b}$ de la variance σ^2 . On a donc :

$$MSE_a(\hat{\boldsymbol{\theta}}^{*b}) = \widehat{\sigma_a^2}^{*b} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i^{*b} - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x}_i^{*b})\right)^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{\epsilon}_{a,i}^{*b})^2 \tag{2.3}$$

où $\hat{\epsilon}_{a,i}^{*b} = y_i^{*b} - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x}_i^{*b})$ est la i-ième composante du vecteur $\hat{\epsilon}_a^{*b}$.

Dans le cas des perceptrons multicouches, la base initiale $\mathcal{B}_0$ est considérée comme la base de test pour chaque apprentissage effectué sur les bases bootstrappées $\mathcal{B}^{*b}$ pour $1 \leq b \leq B$. On définit ainsi un vecteur résidu test $\hat{\epsilon}_t^{*b} = y - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x})$ et par suite une erreur test Eq.(1.9) pour $1 \leq b \leq B$, notée par abus de langage $\widehat{\sigma_t^2}^{*b}$. On a donc :

$$MSE_t(\hat{\boldsymbol{\theta}}^{*b}) = \widehat{\sigma_t^2}^{*b} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x}_i)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{\epsilon}_{t,i}^{*b})^2 \tag{2.4}$$

où $\hat{\epsilon}_{t,i}^{*b} = y_i - f_{\hat{\boldsymbol{\theta}}^{*b}}(\boldsymbol{x}_i)$ est la i-ième composante du vecteur $\hat{\epsilon}_t^{*b}$.

Ainsi, on obtient un vecteur noté $\Sigma_t = (\widehat{\sigma_t^2}^{*1}, \ldots, \widehat{\sigma_t^2}^{*B})$ de taille B, de moyenne $\hat{\mu}_{boot} = \overline{\widehat{\sigma_t^2}}^{*}$ et de variance $\hat{\sigma}^2_{boot}$:

$$\hat{\mu}_{boot} = \overline{\widehat{\sigma_t^2}}^{*} = \frac{1}{B}\sum_{b=1}^{B}\widehat{\sigma_t^2}^{*b} \qquad \hat{\sigma}^2_{boot} = \frac{1}{B}\sum_{b=1}^{B}\left(\widehat{\sigma_t^2}^{*b} - \overline{\widehat{\sigma_t^2}}^{*}\right)^2. \tag{2.5}$$

---

1. Générer B échantillons de taille n par tirage aléatoire uniforme et avec remise sur la base initiale $\mathcal{B}_0 = \{(\boldsymbol{x}_i, y_i);\quad 1 \leq i \leq n\}$ par la méthode du bootstrap par paires. Désignons par $\mathcal{B}^{*b} = \{(\boldsymbol{x}_i^{*b}, y_i^{*b});\quad 1 \leq i \leq n\}$ le b-ième échantillon bootstrappé, pour $1 \leq b \leq B$.

2. Pour chaque échantillon bootstrappé, $b = 1, \ldots, B$, estimer $\boldsymbol{\theta}$ en minimisant
$\sum_{i=1}^{n}\left(y_i^{*b} - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{*b})\right)^2$, on obtient $\hat{\boldsymbol{\theta}}^{*b}$.

3. On calcule alors :

$$\hat{\sigma}^2_{boot} = \frac{1}{B}\sum_{b=1}^{B}\left(\widehat{\sigma_t^2}^{*b} - \overline{\widehat{\sigma_t^2}}^{*}\right)^2 \tag{2.6}$$

et

$$\hat{\mu}_{boot} = \overline{\widehat{\sigma_t^2}}^{*} = \frac{1}{B}\sum_{b=1}^{B}\widehat{\sigma_t^2}^{*b} \tag{2.7}$$

TAB. 2.1 – Algorithme de ré-échantillonnage de type bootstrap choisi pour nos simulations, (typiquement $20 \leq B \leq 200$).

La variable $\hat{\mu}_{boot}$ est un estimateur de la variance résiduelle σ^2 du modèle, égal à la moyenne des B estimateurs bootstraps $\widehat{\sigma_t^2}^{*b}$ pour $1 \leq b \leq B$. La variable $\hat{\sigma}^2_{boot}$ est

la dispersion de ces B estimateurs. Cette technique permet donc d'évaluer un modèle simplement à partir d'une seule base sans la partager en base d'apprentissage et en base de test, ce qui diminue le nombre d'observations utilisées pour l'estimation.

Pour choisir entre plusieurs modèles $M_1, M_2, \ldots$, ces calculs sont répétés pour chacun d'entre eux. La meilleure architecture sera celle qui présente le meilleur compromis $(\hat{\mu}_{boot}, \hat{\sigma}^2_{boot})$, l'idéal étant de minimiser simultanément $\hat{\mu}_{boot}$ et $\hat{\sigma}^2_{boot}$.

2.2 Exemples

Dans cette section, nous illustrons l'application du bootstrap à la sélection de modèle. On montre que cette méthode peut s'appliquer à la sélection de modèle dans le cas linéaire pour un exemple de données simulées. Un deuxième exemple confirme notre approche dans le cas des perceptrons multicouches pour des données simulées aussi. Enfin, on applique cette méthode à des données réelles (Kallel *et al.* [49], [51]).

2.2.1 Exemple 1 : modèle linéaire simulé

Reprenons le modèle linéaire défini par l'Eq.(1.13) :

$$Y = X\boldsymbol{\theta} + \epsilon = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \epsilon$$

Le modèle est simulé de la façon suivante :

- $p = 2$, les observations des variables explicatives sont données par : $X_1 = 2\mathcal{N}(0,1) + 0.2$ et $X_2 = 0.5\mathcal{N}(0,1) - 0.2$,
- l'erreur du modèle est $\epsilon \sim 2\mathcal{N}(0,1)$, de variance 4,
- le vecteur paramètre $\boldsymbol{\theta}$ est choisi égal à $(2, 0.7, 1)$.

On simule donc la base de données $\mathcal{B}_0 = (\boldsymbol{x}, y)$ de taille $n = 500$ observations, et on applique la méthode du bootstrap par paires (voir le paragraphe (1.4)) sur trois modèles notés M_1, M_2, M_3 avec $B = 50$ bases bootstrappées :

- Modèle $M_1 : Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \epsilon$, le vrai modèle,
- Modèle $M_2 : Y = \theta_0 + \theta_1 X_1 + \epsilon$,
- Modèle $M_3 : Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \epsilon$, où $X_3 = 1.5\mathcal{N}(0,1) - 1$.

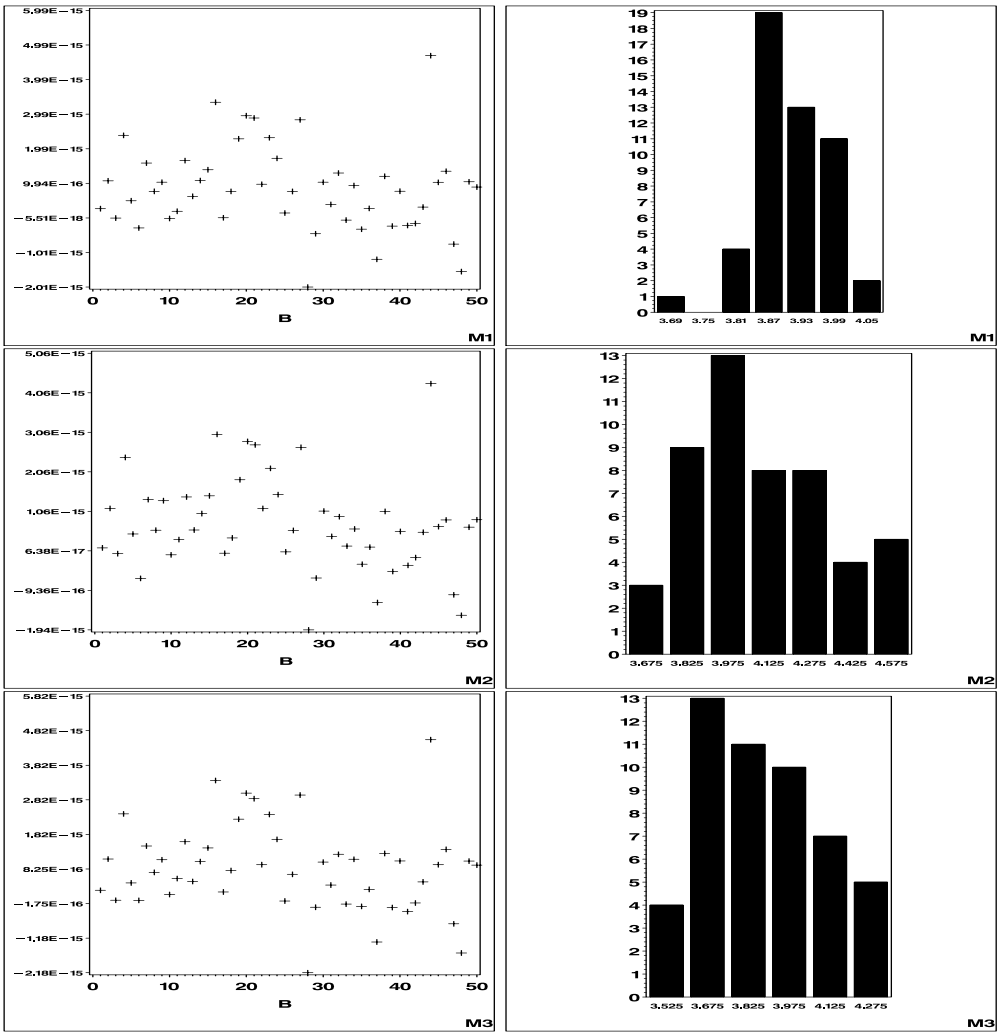


FIG. 2.1 – Première colonne : La moyenne des B vecteurs résidus $\hat{\epsilon}_t^{*b}$ pour les trois modèles. Seconde colonne : Les histogrammes des composantes du vecteur Σ_t pour les trois modèles.

Modèle	$\hat{\mu}_{boot}$	$\hat{\sigma}_{boot}$
M_1	3.8875	0.0668
M_2	4.0897	0.2523
M_3	3.8796	0.2124

TAB. 2.2 – Valeurs des $\hat{\mu}_{boot}$ et des $\hat{\sigma}_{boot}$ pour les trois modèles.

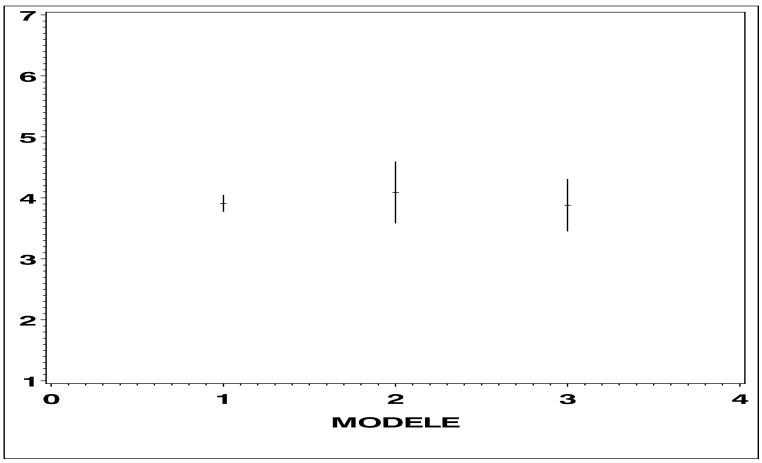


FIG. 2.2 – Boxplots de $\hat{\mu}_{boot}$ pour l'exemple 1.

Pour chaque modèle proposé, $\hat{\mu}_{boot}$ et $\hat{\sigma}^2_{boot}$ sont calculées selon l'équation (2.5). On obtient ainsi le tableau (2.2).

Dans cet exemple, la figure (3.2) montre que les vecteurs $\hat{\epsilon}^{*b}_t$ sont bien centrés et que les histogrammes du vecteur[1] Σ ont une allure correspondant à une loi de χ^2 pour tous les modèles. D'après la table (2.2) et les boxplots[2] de la figure (2.5), il est évident que le bon modèle est le modèle M_1 qui est justement le vrai modèle.

2.2.2 Exemple 2 : modèle non-linéaire avec données simulées

On simule une base de donnée $\mathcal{B}_0 = \{(\boldsymbol{x}_i, y_i)\}; \quad i = 1, \ldots, 500\}$, en calculant y_i comme la sortie bruitée d'un perceptron multicouches Eq.(1.6), défini par :

- ➤ $p = 2$ variables d'entrées, $x_1 = 2\mathcal{N}(0,1) + 0.2$ et $x_2 = 0.5\mathcal{N}(0,1) - 0.1$,
- ➤ une couche cachée de 4 neurones,
- ➤ $\boldsymbol{\theta} = (0.5, -0.1, 0.2, 0.5, -0.4, 0.2, 0.1, 3, 0.3, 2, 0.5, 0.1, 0.2, 2, 0.2, 3, 0.1)$, (figure 2.3),
- ➤ $\epsilon \sim 0.2\mathcal{N}(0,1)$, de variance 0.04.

On construit $B = 50$ bases bootstrappées à partir de la base initiale $\mathcal{B}_0$ en appliquant la méthode du bootstrap par paires (voir le paragraphe (1.4)), et on compare

[1]Dans le cas linéaire, on n'utilise pas $\mathcal{B}_0$ pour faire un test. Les vecteurs des erreurs Σ_t sont construits à partir des B bases bootstrappées.

[2]Le boxplot est un segment dont le milieu est la moyenne et dont la demi-longueur est deux fois l'écart-type.

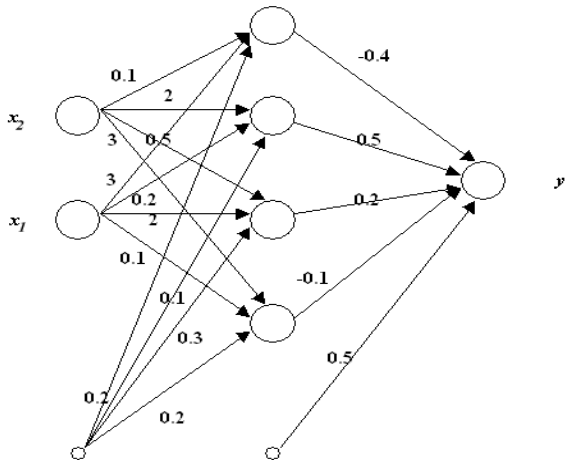


FIG. 2.3 – Le perceptron multicouches pour l'exemple 2.

trois modèles notés M_2, M_4, M_6, d'architectures différentes.

- ➤ Modèle M_2 : deux entrées, une couche cachée avec deux neurones cachés,
- ➤ Modèle M_4 : deux entrées, une couche cachée avec quatre neurones cachés, le vrai modèle,
- ➤ Modèle M_6 : deux entrées, une couche cachée avec six neurones cachés.

La comparaison s'effectue par le calcul de $\hat{\mu}_{boot}$ et $\hat{\sigma}^2_{boot}$ (Eq.(2.5)) à partir de la base de test $\mathcal{B}_0$. On obtient ainsi le tableau (2.3).

Modèle	$\hat{\mu}_{boot}$	$\hat{\sigma}_{boot}$
M_2	0,04277	0,00019
M_4	0,04271	0,00029
M_6	0,04277	0,00028

TAB. 2.3 – Valeurs des $\hat{\mu}_{boot}$ et des $\hat{\sigma}_{boot}$ pour les trois modèles.

Dans cet exemple, la figure (2.4) montre que les vecteurs résidus sont bien centrés pour chaque modèle et pour $1 \leq b \leq B$, et que les histogrammes des vecteurs résidus ont une allure correspondant à une loi de χ^2 pour les trois modèles. Le tableau (2.3) et la figure (2.5) montrent qu'il est évident que le meilleur modèle est le modèle M_2. Ce n'est pas le vrai modèle, mais c'est le meilleur. Ce n'est pas surprenant puisque

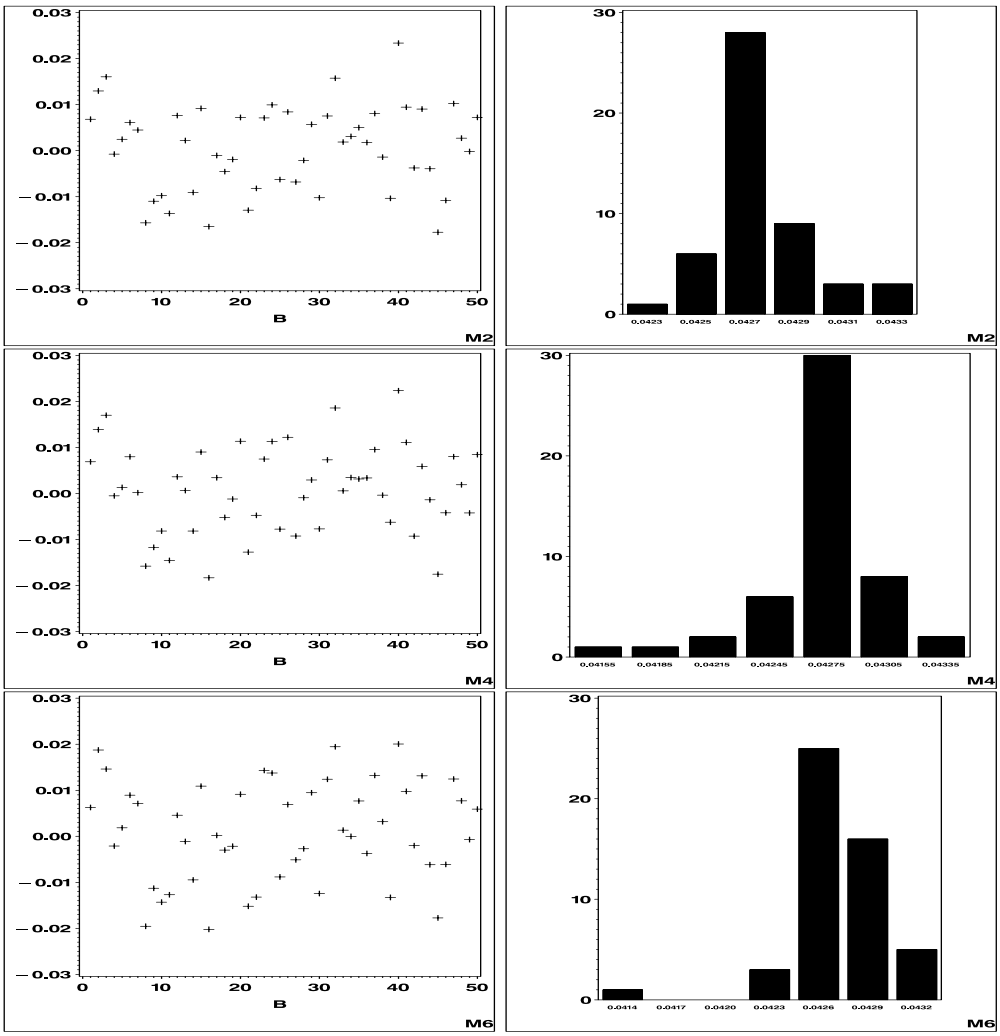


FIG. 2.4 – Première colonne : La moyenne des B vecteurs résidus $\hat{\epsilon}_t^{*b}$ pour les trois modèles. Seconde colonne : Les histogrammes des composantes du vecteur Σ_t pour les trois modèles.

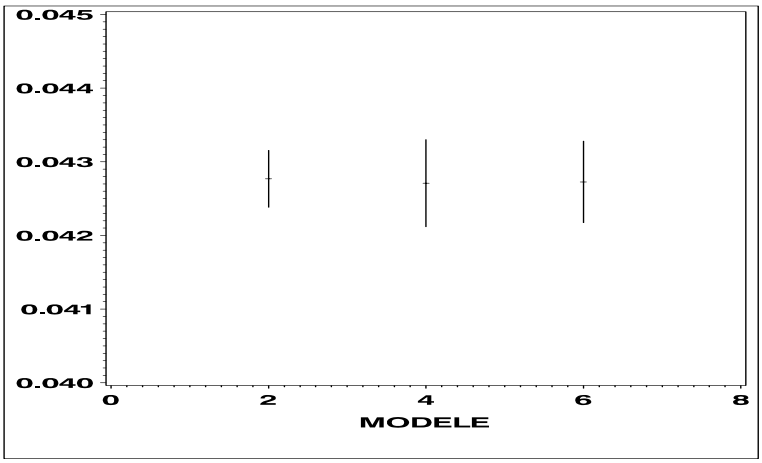


FIG. 2.5 – Boxplots de $\hat{\mu}_{boot}$ pour l'exemple 2.

les MLP sont toujours surparamétrés et qu'il n'y a pas d'unicité de la fonction perceptron pour modéliser une fonction donnée.

2.2.3 Exemple 3 : modèle non-linéaire avec données réelles

Dans cette section, nous étudions un ensemble de données réelles pour établir l'efficacité de la méthode de sélection de modèle que nous proposons. Il s'agit du problème du contrôle du pic de puissance pour le coeur des réacteurs nucléaires. Ce problème a été traité par Gaudier [37] qui proposait un modèle neuronal comportant $p = 22$ entrées, une sortie, $n = 3000$ observations et deux couches cachées ; la première couche cachée contient 26 neurones, la seconde 40 neurones. Le modèle choisi prenait en compte la position des barres d'uranium et les phénomènes de diffusion. Il «devait» permettre de gagner en rapidité sur les codes de calculs numériques classiques au terme de l'apprentissage, tout en conservant leurs performances. La question est de savoir si un modèle est vraiment approprié au problème.

$B = 50$ échantillons bootstrappés sont construits et 3 modèles d'architectures différentes sont comparés :

- ➤ Modèle M_{40} : 22 entrées, deux couches cachées avec respectivement 26 et 40 neurones cachés,
- ➤ Modèle M_{35} : 22 entrées, deux couches cachées avec respectivement 26 et 35 neurones cachés,

➤ Modèle M_{30} : 22 entrées, deux couches cachées avec respectivement 26 et 30 neurones cachés.

Modèle	$\hat{\mu}_{boot}$	$\hat{\sigma}_{boot}$
M_{30}	0,0473	0,0052
M_{35}	0,0599	0,0069
M_{40}	0,0492	0,0049

TAB. 2.4 – Valeurs des $\hat{\mu}_{boot}$ et des $\hat{\sigma}_{boot}$ pour les trois modèles.

Dans cet exemple, on remarque la même chose que dans l'exemple 1 et l'exemple 2 pour la moyenne des vecteurs résidus et les histogrammes du vecteur Σ_t (voir Fig.(2.6)). Mais (voir Tab.2.4 et Fig.2.7), la conclusion n'est pas évidente, le modèle M_{30} semble être le meilleur modèle (son erreur est la plus petite), mais le plus stable est le modèle M_{40}. Dans ce cas, il est nécessaire d'étudier d'autres architectures, différentes des 3 que nous avons considérées (cf. Tab.2.4).

2.2.4 Comparaison avec le jackknife : leave-one-out

Dans cette partie, on compare les deux méthodes : bootstrap et jackknife (leave-one-out) pour la sélection de modèles. Pour cela, on a réalisé le même travail pour les trois exemples avec la méthode jackknife (Kallel *et al.* [51]). On a appliqué les mêmes équations que dans le cas du bootstrap Eq.(2.5) sur des bases construites en enlevant une seule observation à la fois de la base initiale suivant le principe de la méthode leave-one-out.

On a utilisé un nombre fixe de réplications dans les deux cas ($B = 50$). Pour la construction de ces B replications jackknife, on a fait un tirage aléatoire uniforme et sans remise de B observations parmi les n observations de la base initiale. À chaque construction d'une replication, on enlève une observation parmi les n. On choisit le nombre $B < n$ de façon à économiser du temps de calcul et à utiliser le même nombre de réplications pour les deux méthodes. Par exemple, dans l'exemple 3, si on utilisait autant de réplications que d'observations, on aurait $B = n = 3000$ bases à apprendre, ce qui devient impraticable.

Par analogie, on note $\hat{\mu}_{loo}$ l'estimateur de la variance résiduelle σ^2 et $\hat{\sigma}_{loo}$ sa dispersion, calculées par la méthode leave-one-out. Les résultats sont résumés dans le tableau (2.5).

On remarque que dans les trois exemples et pour les trois modèles de chaque exemple, les résultats fournis par le bootstrap sont plus nets que ceux qui ont été

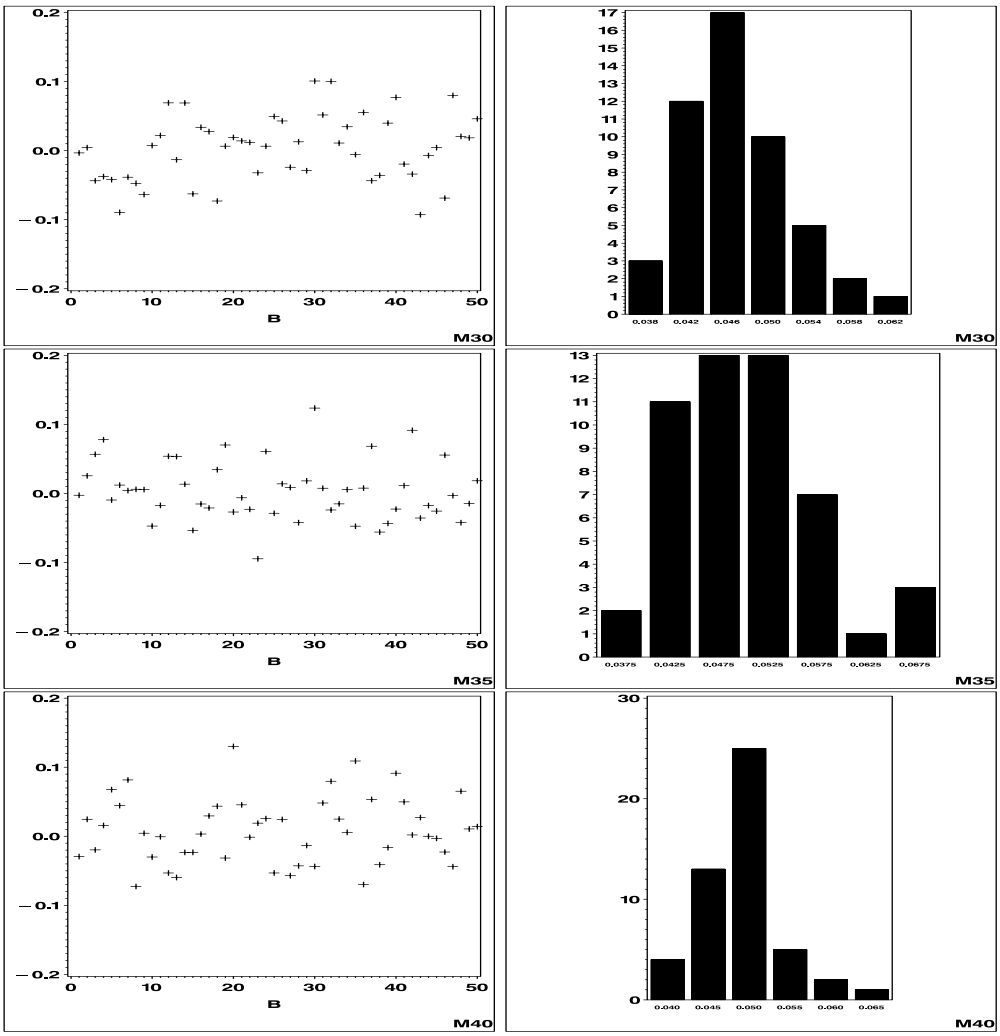


FIG. 2.6 – Première colonne : La moyenne des B vecteurs résidus $\hat{\epsilon}_t^{*b}$ pour les trois modèles. Seconde colonne : Les histogrammes des composantes du vecteur Σ_t pour les trois modèles.

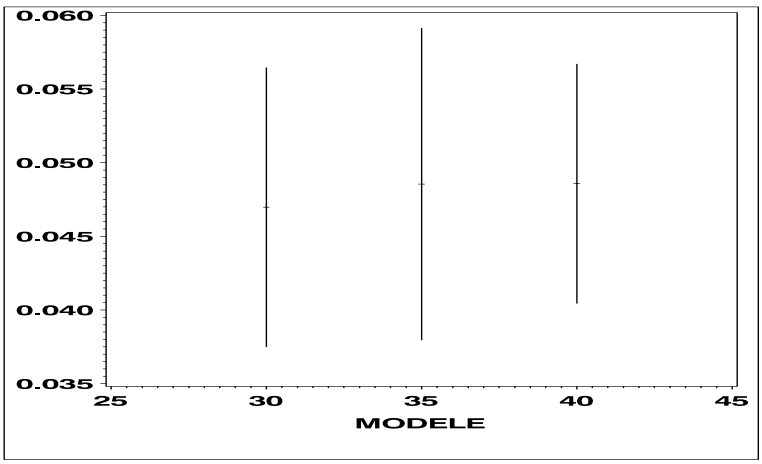


FIG. 2.7 – Boxplots de $\hat{\mu}_{boot}$ pour l'exemple 3.

Modèle		Bootstrap		Leave-one-out[a]	
		$\hat{\mu}_{boot}$	$\hat{\sigma}_{boot}$	$\hat{\mu}_{loo}$	$\hat{\sigma}_{loo}$
Exp 1	M_1	**3.8875**	**0.0668**	4.56489	0.13394
	M_2	4.0897	0.2523	4.70375	0.76858
	M_3	3.8796	0.2124	4.49751	0.69901
Exp 2	M_2	**0,04277**	**0.00019**	0.04999	0.06807
	M_4	0.04271	0.00029	0,05303	0.07553
	M_6	0.04277	0.00028	0.04895	0.06772
Exp 3	M_{30}	**0,0473**	**0.0052**	0.03961	0.05347
	M_{35}	0.0599	0.0069	0.05132	0.07873
	M_{40}	0.0492	0.0049	0.04763	0.08161

[a] On a utilisé 50 réplications de la base initiale pour chaque modèle

TAB. 2.5 – Tableau résumé : Comparaison des deux méthodes : bootstrap et leave-one-out.

donnés par le jackknife. L'erreur quadratique moyenne est la plupart du temps plus faible pour le bootstrap et la dispersion beaucoup plus faible.

2.3 Conclusion

L'utilisation du bootstrap pour la sélection de modèles dans le cadre des réseaux de neurones a permis d'obtenir de bons résultats. Les exemples traités dans ce chapitre montrent l'efficacité de cette approche. De nombreuses autres applications ont été réalisées sur des exemples simulés et ont tous confirmé cette efficacité. Il y a cependant deux inconvénients à cette méthode :

- *le temps de simulation* : si n ou p sont grands, le calcul peut être assez coûteux, car il peut nécessiter un nombre important d'itérations, même avec des techniques d'optimisation du second-ordre de type BFGS [18]. Il est cependant moins coûteux que la sélection empirique soumise à de nombreux aléas qui en rendent l'emploi délicat.
- *la répétition des données extrêmes* : le risque existe de sélectionner un jeu de données pour lequel la méthode itérative d'apprentissage converge difficilement. Ignorer ces répétitions risque alors d'introduire un biais. La mise en oeuvre du bootstrap dans les modèles non-linéaires nécessite en général un plus grand nombre de répétitions que pour les modèles de régression linéaire, $B = 50$ paraît ici un nombre raisonnable.

Chapitre 3

Application du bootstrap au test asymptotique de différence de contrastes

Dans ce chapitre, on considère le cas où la variable à expliquer à un instant donné dépend des variables passées. Dans ce cas, on utilise un modèle auto-régressif linéaire ou non-linéaire (en particulier des perceptrons multicouches) pour déterminer cette relation. On se trouve également confronté au problème de sélection de modèles sous forme de choix d'un sous-modèle emboîté dans un modèle plus grand. Classiquement, on utilise dans ce cadre les tests asymptotiques de différence de contrastes[1] pour choisir entre deux modèles envisagés.

Nous commençons par rappeler (Mangeas [61], [62], Yao [78]) les propriétés essentielles d'un modèle auto-régressif fonctionnel, qui comprend les perceptrons multicouches comme cas particulier. Nous rappelons ensuite les propriétés des tests asymptotiques de différence de contrastes. Une application à des données simulées montre la faiblesse de la vitesse de convergence de ce test, et son manque de validité dès lors que le nombre d'observations n'est pas de l'ordre de la centaine de milliers. Notre contribution consiste à proposer d'appliquer la méthode du bootstrap paramétrique à ce test, et nous montrons la consistance de cette méthode.

Soient deux entiers $p, d \geq 1$. Un processus autorégressif fonctionnel sur $\mathbb{R}^d$ est une suite de vecteurs aléatoires de $\mathbb{R}^d$ vérifiant :

[1] On appelle fonction de contraste relative à un paramètre $\theta \in \Theta$ d'un modèle statistique, une fonction $\alpha \to k(\alpha, \theta)$ d'un compact de Θ vers l'ensemble des réels positifs, ayant un minimum strict pour $\alpha = \theta$. Exemples classiques : l'erreur quadratique, la vraisemblance ; Dacunha (1983).

$$X_t = f(X_{t-1}, \ldots, X_{t-p}, \theta) + \epsilon_t \tag{3.1}$$

où ϵ_t est un bruit i.i.d. de matrice de covariance Γ et f est une fonction qui dépend d'un paramètre θ. Le paramètre θ appartient à un sous-ensemble Θ de $\mathbb{R}^s$ $(s \in \mathbb{N}^*)$. Un tel modèle est désigné par la suite par $ARF_d(p)$. On notera aussi $X^{(p)} = (X_t^{(p)})_{t\geq 1}$ le processus vectorisé associé, défini par $X_t^{(p)} = (X_t, \ldots, X_{t-p+1})^T$.

Notons, pour un vecteur $x = (x_1, \ldots, x_p) \in (\mathbb{R}^d)^p$, la norme $|x| = ||x_1|| + \ldots + ||x_p||$, où $||.||$ est la norme euclidienne de $\mathbb{R}^d$. Pour une matrice A, on définit sa norme par $||A|| = \{\sup ||Ax||, ||x|| = 1\}$. La vraie valeur du paramètre à estimer est notée θ_0.

On définit *le processus de contraste des moindres carrés* par :

$$U_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} ||X_t - f(X_{t-1}, \ldots, X_{t-p}, \theta)||^2, \tag{3.2}$$

et *l'estimateur des moindres carrés* par :

$$\hat{\theta}_n = \operatorname{Arg}\min_{\theta\in\Theta} U_n(\theta). \tag{3.3}$$

Nous nous intéressons, dans ce chapitre, aux propriétés asymptotiques de l'estimateur $\hat{\theta}_n$ et aux tests asymptotiques de différence de contrastes.

3.1 Loi forte des grands nombres pour les fonctions non bornées d'un $ARF_d(p)$

On considère ici le processus vectorisé associé et on note

$$\begin{aligned} X_t^{(p)} &= \begin{pmatrix} X_t \\ . \\ . \\ . \\ X_{t-p+1} \end{pmatrix} = \begin{pmatrix} f(X_{t-1}, \ldots, X_{t-p}, \theta) \\ . \\ . \\ . \\ X_{t-p+1} \end{pmatrix} + \begin{pmatrix} \epsilon_t \\ 0 \\ . \\ . \\ . \\ 0 \end{pmatrix} \\ &= F(X_t^{(p)}, \theta) + \eta_t \end{aligned} \tag{3.4}$$

avec les définitions explicites correspondantes pour F et η. On note aussi $\mathbb{P}_{\theta_0}$ la loi sous le vrai modèle, $\xrightarrow{p.s}$ (respectivement $\xrightarrow{C.L}$) la convergence $p.s$ (respectivement la

convergence en loi) sous $\mathbb{P}_{\theta_0}$.

La loi forte des grands nombres (LFGN) pour les fonctions moments d'un ordre suffisant de $X_t^{(p)}$ constitue l'une des clefs principales permettant d'établir les propriétés asymptotiques de l'estimateur $\hat{\theta}_n$. Plus précisément, nous nous plaçons dans le cadre suivant :

Hypothèse de stabilité [S] :

On suppose que :

① $X_t^{(p)}$ converge en loi (quand $t \longrightarrow +\infty$) vers une variable aléatoire X de mesure invariante μ,

② $\exists\ a > 1$, tel que pour tout t et quelle que soit la loi initiale, on a $E(|X_t^{(p)}|^a) < \infty$,

③ Pour toute fonction $\Phi : (\mathbb{R}^d)^p \to \mathbb{R}$ μ-p.s-continue, satisfaisant $|\Phi(.)| \leq c^{te}(1 + |.|^a)$, on a la LFGN :

$$n^{-1} \sum_{t=1}^{n} \Phi(X_t^{(p)}) \xrightarrow{p.s} \int_{(\mathbb{R}^d)^p} \Phi(x)\mu(dx).$$

Théorème 4

Si le processus $ARF_d(p)$ Eq.(3.1) vérifie la condition suivante :

➤ *le bruit ϵ_t a une densité strictement positive par rapport à la mesure de Lebesgue, et possède un moment d'ordre $a > 1$,*

➤ *il existe p nombres positifs $\lambda_1, \ldots, \lambda_p$ tels que $\sum_{i=1}^{p} \lambda_i < 1$, et une constante $\kappa \leq 0$ satisfaisant pour tout $x \in (\mathbb{R}^d)^p$,*

$$||f(x, \theta_0)|| \leq \sum_{i=1}^{p} \lambda_i ||x_i|| + \kappa,$$

Alors, le modèle $ARF_d(p)$ Eq.(3.1) satisfait l'hypothèse de stabilité **[S]** *sous θ_0.*

Ce critère est bien connu (voir Doukhan [26], Doukhan et Tsybakov [27], Duflo [29]).

3.2 Modèle : estimation des moindres carrés et fonction de contraste associée

Introduisons le module de continuité comme une fonction $g : [0, \infty] \to [0, \infty]$ vérifiant les deux conditions suivantes :

- g est croissante,
- $\lim_{x \to 0} g(x) = g(0) = 0$.

Hypothèse [M] : Cadre des modèles étudiés

① La famille des fonctions utilisée dans l'équation (3.1) est représentée par :

$$f(., \theta) : (\mathbb{R}^d)^p \to \mathbb{R}^d$$

avec $\theta \in \Theta$, où Θ est un compact de $\mathbb{R}^s$ et $s \in \mathbb{N}^*$.

② Le bruit $(\epsilon_t)_{t>0}$ est i.i.d. à valeurs dans $\mathbb{R}^d$, centré, de matrice de variance-covariance Γ et indépendant de la valeur initiale $X_0^{(p)}$.

③ Stabilité : la fonction f est continue par rapport à θ, et f et le bruit ϵ_t satisfont les hypothèses du théorème 4 avec un $a \geq 2$.

Proposition 1

Dans le cadre de l'hypothèse **[M]** *et quelle que soit la loi initiale de* $X_0^{(p)}$*, on a :*

$$\lim_{n \to \infty} [U_n(\theta) - U_n(\theta_0)] = \int_{(\mathbb{R}^d)^p} ||f(x, \theta) - f(x, \theta_0)||^2 \mu(dx) = K(\theta, \theta_0). \tag{3.5}$$

De plus $K(\theta, \theta_0)$ *est continue en* θ.

θ_0 est clairement un minimum absolu de la fonction K. C'est le seul si le modèle vérifie la condition d'identifiabilité suivante :

Condition d'identifiabilité [D] :

Le modèle est dit identifiable si :

$$\forall \theta_1, \theta_2 \in \Theta^2 \quad \text{on a} \quad f(., \theta_1) = f(., \theta_2) \quad \mu\text{-}p.s \Longrightarrow \theta_1 = \theta_2.$$

3.3 Consistance forte et normalité asymptotique

Nous allons maintenant rappeler la consistance de l'estimateur des moindres carrés $\hat{\theta}_n$. On a le théorème de consistance forte suivant :

Théorème 5

Sous les hypothèses **[M]** *et* **[D]**, *l'estimateur* $\hat{\theta}_n = Arg\min_{\theta\in\Theta} U_n(\theta)$, *estimateur des moindres carrés, est fortement consistant.*

Pour la normalité asymptotique, on a besoin de conditions supplémentaires usuelles sur la dérivabilité d'ordre deux du processus de contraste $U_n(\theta)$ (Eq.3.2). Si $\phi(\theta)$ est une fonction différentiable, ses dérivées partielles sont notées $\nabla_i\phi = \frac{\partial\phi}{\partial\theta_i}$, $\nabla^2_{i,j}\phi = \frac{\partial^2\phi}{\partial\theta_i\partial\theta_j}$, sa dérivée d'ordre 1, $\nabla\phi$ et sa dérivée d'ordre 2, $\nabla^2\phi$. On se place dans la suite sous les hypothèses suivantes :

Hypothèse [N] :

On suppose que **[M]** et **[D]** sont satisfaites. On suppose aussi qu'il existe V voisinage de θ_0 tel que, $\forall x \in (\mathbb{R}^d)^p$, la fonction f est deux fois continûment dérivable et telle que $\forall i, j = 1, \ldots, s$, on ait :

① $\forall\theta \in V$, les deux fonctions $x \longmapsto \nabla_i f(x,\theta)$ et $x \longmapsto \nabla^2_{i,j} f(x,\theta)$ sont μ-p.s-continues.

② $\forall x \in (\mathbb{R}^d)^p$, on a $\nabla_i f(x,\theta_0) \leq c^{te}(1+|x|^{a/2})$ et $\nabla^2_{i,j} f(x,\theta_0) \leq c^{te}(1+|x|^{a/2})$.

③ Il existe un module de continuité $\sigma_{i,j}$ tel que :

$$|\nabla^2_{i,j} f(x,\theta) - \nabla^2_{i,j} f(x,\theta_0)| \leq \sigma_{i,j}(\theta-\theta_0)(1+|x|^{a/2}).$$

Notons que la dernière hypothèse de **[N]** fournit un contrôle de la croissance de ces fonctions à l'infini. Cette croissance est assurée par la condition

$$\nabla^3 f(x,\theta_0) \leq c^{te}(1+|x|^{a/2})$$

De même, la compacité de Θ et les deux dernières hypothèses de **[N]** impliquent :

$$\begin{array}{c} \forall i,j, \quad \forall\theta \in V \quad \text{et} \quad \forall x \in (\mathbb{R}^d)^p, \quad \exists\gamma > 0 \quad \text{tel que} \\ |\nabla^2_{i,j} f(x,\theta)| \leq \gamma(1+|x|^{a/2}). \end{array} \tag{3.6}$$

On en déduit un contrôle d'accroissement des dérivées premières :

$$\begin{array}{l} \forall i,j, \quad \forall\theta \in V \quad \text{et} \quad \forall x \in (\mathbb{R}^d)^p, \quad \exists\gamma > 0 \quad \text{tel que} \\ |\nabla_i f(x,\theta) - \nabla_i f(x,\theta_0)| \leq \gamma(||\theta-\theta_0||)(1+|x|^{a/2}). \end{array} \tag{3.7}$$

Et enfin :

$$\forall i,j, \quad \forall \theta \in V \quad \text{et} \quad \forall x \in (\mathbb{R}^d)^p, \quad \exists \gamma' > 0 \quad \text{tel que} \quad |\nabla_i f(x,\theta)| \leq \gamma'(1+|x|^{a/2}). \tag{3.8}$$

On note :

$$\nabla f(x,\theta) = [\nabla_j f(x,\theta)]_{1\leq j\leq s}$$

une matrice de dimensions $d \times s$, et

$$M(x,\theta) = \nabla^T f(x,\theta) \nabla f(x,\theta)$$

une matrice de dimensions $s \times s$.

La matrice $M(x,\theta)$ vérifie alors en raison des deux contrôles (Eq.3.7), (Eq.3.8) :

$$||M(x,\theta) - M(x,\theta_0)|| \leq c^{te}||\theta - \theta_0||(1+|x|^a), \tag{3.9}$$

$$||M(x,\theta)|| \leq c^{te}(1+|x|^a). \tag{3.10}$$

Les dérivées du contraste U_n s'écrivent respectivement :

$$\nabla U_n(\theta) = -\frac{2}{n} \sum_{0\leq t\leq n} \epsilon_{t+1}^T \nabla f(X_t^{(p)}, \theta), \tag{3.11}$$

$$\frac{1}{2}\nabla^2 U_n(\theta) = \frac{1}{n}\sum_{0\leq t\leq n} M(X_t^{(p)},\theta) - \frac{1}{n}\left[\sum_{0\leq t\leq n} \epsilon_{t+1}^T \nabla^2_{i,j} f(X_t^{(p)},\theta)\right]_{1\leq i,j\leq s}. \tag{3.12}$$

On a alors les deux résultats suivants sur le gradient et la hessienne du contraste au point θ_0.

Proposition 2

Dans le cadre de l'hypothèse **[N]** *on a :*

$$\nabla^2 U_n(\theta_0) \xrightarrow{p.s} I_0$$

avec $I_0 = 2\int_{(\mathbb{R}^d)^p} M(x,\theta_0)\mu(dx)$
On a aussi

$$\sqrt{n}\nabla U_n(\theta_0) \xrightarrow{C.L} N(0, J_0)$$

avec $J_0 = 4\int_{(\mathbb{R}^d)^p} \nabla^T f(x,\theta_0)\Gamma\nabla f(x,\theta_0)\mu(dx)$.

Remarque 1 *Dans le cas scalaire* ($d=1$), *la variance* $\Gamma = \sigma^2$ *du bruit est scalaire. Alors on a* $J_0 = 2\sigma^2 I_0$.

On note :

$$\Delta_n(\hat{\theta}_n) = \int_0^1 \nabla^2 U_n[\hat{\theta}_n + u(\hat{\theta}_n - \theta_0)]du \tag{3.13}$$

Lemme 1 *Sous l'hypothèse* **[N]**, *on a :*

$$\Delta_n(\hat{\theta}_n) - \nabla^2 U_n(\theta_0) \xrightarrow{p.s} 0 \quad et \quad \Delta_n(\hat{\theta}_n) \xrightarrow{p.s} I_0 \tag{3.14}$$

Compte tenu du lemme 1 et de la proposition 2, on a le théorème suivant :

Théorème 6

Dans le cadre de l'hypothèse **[N]**, *et si de plus,* I_0 *est inversible, on a :*

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow{C.L} N(0, I_0^{-1} J_0 I_0^{-1}).$$

Nous avons rappelé la consistance forte et la normalité asymptotique de l'estimateur des moindres carrés $\hat{\theta}_n$ d'un paramètre θ dans un compact d'un espace de dimension finie s supposée connue. Si celle-ci est inconnue, on doit choisir la meilleure dimension possible pour le modèle. C'est l'objet de la section suivante.

3.4 Test asymptotique de différence de contrastes

Soit q un entier inférieur à s. On note H_s l'hypothèse $\{\theta \in \Theta \subset \mathbb{R}^s\}$. Une sous-hypothèse H_q de H_s exprime le fait que θ appartient à un sous-ensemble de Θ de dimension paramétrique q plus petite que s, obtenu en annulant $s - q$ composantes du paramètre (sans fixer les positions de ces composantes nulles).

On note $\hat{\theta}_n^s$ (resp. $\hat{\theta}_n^q$), les estimateurs des moindres carrés du paramètre θ, sous l'hypothèse H_s (resp. H_q). Pour tester H_q contre H_s, on utilise la statistique de différence de contrastes :

$$T_n = 2n[U_n(\hat{\theta}_n^s) - U_n(\hat{\theta}_n^q)].$$

On a alors le résultat suivant (Guyon [40], et Bayomog et al. [9]) qui précise la loi asymptotique de la statistique de test T_n sous l'hypothèse H_q :

Proposition 3

Si les hypothèses du théorème Th.6 sont vérifiées par les deux modèles H_s et H_q, alors sous l'hypothèse H_q on a ;

$$T_n \xrightarrow{C.L} \sum_{i=1}^{s-q} \lambda_i \chi^2_{i,1},$$

où les $\chi^2_{i,1}$ sont $s-q$ variables aléatoires indépendantes de loi χ^2_1, et où les λ_i sont des constantes positives.

Remarque 2 *Dans le cas scalaire, $\lambda_1 = \ldots = \lambda_{s-q} = 2\sigma^2$. La statistique de test suit alors à $2\sigma^2$ près (constante multiplicative), une loi du χ^2 à $s-q$ degrés de liberté.*

3.5 Estimation par la méthode du maximum de vraisemblance : Propriétés asymptotiques

On rappelle dans cette partie (Rynkiewicz [71]), la définition de l'estimateur du maximum de vraisemblance pour un processus autorégressif fonctionnel $ARF_d(p)$ (Eq.3.1), ainsi que ses propriétés asymptotiques. Pour plus de détails et pour les démonstrations des théorèmes qu'on va citer, voir le chapitre 4 de la thèse de J. Rynkiewicz [71].

On introduit le *processus de contraste associé à la vraisemblance* par :

$$V_n(\theta) = \log\det\left(\frac{1}{n}\sum_{t=1}^{n}(X_t - f(X_{t-1},\ldots,X_{t-p},\theta))(X_t - f(X_{t-1},\ldots,X_{t-p},\theta))^T\right) \tag{3.15}$$

et *l'estimateur du maximum de vraisemblance* par :

$$\tilde{\theta}_n = \operatorname{Arg}\min_{\theta\in\Theta} V_n(\theta) \tag{3.16}$$

$V_n(\theta)$ est la vraisemblance concentrée qui correspond au cas gaussien, mais ses propriétés asymptotiques sont vérifiées même pour des bruits non gaussiens. Dans le cadre des hypothèses **[S],[M]** et **[D]**, et si de plus la matrice de variance-covariance

Γ est définie positive, on a le théorème suivant :

Théorème 7 *:*

Sous les hypothèses **[S]**, **[M]** *et* **[D]**, *l'estimateur* $\tilde{\theta}_n = Arg\min_{\theta\in\Theta} V_n(\theta)$ *est fortement consistant.*

Si de plus les conditions de l'hypothèse **[N]** sont vérifiées pour le processus de contraste associé à la vraisemblance $V_n(\theta)$, on peut appliquer le théorème de normalité asymptotique à l'estimateur $\tilde{\theta}_n$, et on a le théorème suivant :

Théorème 8 *:*

On suppose que l'hypothèse **[N]** *est satisfaite, et que le bruit a un moment d'ordre* $2a$ *avec* $a \geq 2$. *Si de plus* I_0 *est inversible, alors pour toute loi initiale du processus vectorisé* $X_t^{(p)}$

$$\sqrt{n}\left(\tilde{\theta}_n - \theta_0\right) \xrightarrow{C.L} N(0, I_0^{-1} J_0 I_0^{-1})$$

où I_0 *et* J_0, *sont deux matrices symétriques définies de manière analogue à celles de la proposition 2.*

Remarque 3 *Si de plus la densité du bruit est gaussienne,* $I_0 = J_0$ *est la matrice d'information de Fisher du modèle.*

De la même façon que dans le cadre de l'estimateur des moindres carrés (section 3.4), on définit la statistique de différence de contrastes par :

$$TL_n = n[V_n(\tilde{\theta}_n^s) - V_n(\tilde{\theta}_n^q)]. \tag{3.17}$$

On a alors le résultat suivant qui définit la loi asymptotique de test TL_n sous l'hypothèse H_q (section 3.4) :

Proposition 4

Si les hypothèses du théorème Th.8 sont vérifiées par les deux modèles H_s *et* H_q, *et si le bruit* ϵ *est gaussien, alors sous l'hypothèse* H_q *on a ;*

$$TL_n \xrightarrow{C.L} \sum_{i=1}^{s-q} \chi_{i,1}^2,$$

où les $\chi_{i,1}^2$ *sont* $s-q$ *variables aléatoires indépendantes de loi* χ_1^2.

3.6 Vérification du comportement empirique de la statistique du test asymptotique de différence de contrastes sur un modèle MLP

Un perceptron multicouche $MLP(p,k,d)$, où p est le nombre des unités d'entrée, k est le nombre des unités cachées et d est le nombre des unités de sortie, est un cas particulier d'un modèle autorégressif non-linéaire $ARF_d(p)$. Il vérifie bien les hypothèses **[S]**, **[M]**, **[N]** et la condition d'identifiabilité **[D]**. (Rynkiewicz [71]).

On peut donc en théorie appliquer les résultats asymptotiques présentés dans les paragraphes précédents, en particulier la proposition (3) dans le cas de l'estimation par la méthode des moindres carrés, et la proposition (4) dans le cas de l'estimation par la méthode du maximum de vraisemblance. Il s'agit ici dans un premier temps de vérifier si les résultats théoriques sont utilisables en pratique.

L'étude consiste à appliquer la procédure **[P]** suivante :

① simuler B échantillons indépendants de taille n, sous l'hypothèse H_q, avec un modèle $MLP(p,k,d)$, correspondant au nombre de paramètres $q = k(p+d+1)+d$, ϵ de loi $\mathcal{N}(0,\sigma^2)$ et pour un jeu de paramètres fixés,

② sur chaque échantillon, faire l'apprentissage par la méthode des moindres carrés d'un perceptron dont l'architecture est la même que celle du modèle de simulation, c'est-à-dire un $MLP(p,k,d)$, calculer l'estimateur $\hat{\theta}_n^{q,b}$ et calculer $U_n(\hat{\theta}_n^{q,b})$ pour chacun des $b \in \{1,...,B\}$,

③ faire l'apprentissage de la même façon avec un modèle $MLP(p,k',d)$, avec $k' > k$, donc pour un nombre de paramètres $s = k'(p+d+1)+d > q$. Calculer l'estimateur $\hat{\theta}_n^{s,b}$ et calculer $U_n(\hat{\theta}_n^{s,b})$ pour chacun des $b \in \{1,...,B\}$,

④ calculer les statistiques $T_n(b) = 2n(U_n(\hat{\theta}_n^{q,b}) - U_n(\hat{\theta}_n^{s,b}))$ pour chaque base simulée. Nous obtenons ainsi le vecteur D_n de B observations :

$$D_n = (T_n(b))_{1\leq b\leq B} = (2n(U_n(\hat{\theta}_n^{q,b}) - U_n(\hat{\theta}_n^{s,b})))_{1\leq b\leq B},$$

⑤ représenter l'histogramme du vecteur D_n,

⑥ répéter les étapes 2 à 5 en utilisant le maximum de vraisemblance. On obtient ainsi le vecteur

$$DL_n = (TL_n(b))_{1\leq b\leq B} = (n(V_n(\tilde{\theta}_n^{q,b}) - V_n(\tilde{\theta}_n^{s,b})))_{1\leq b\leq B}$$

où $\tilde{\theta}_n^{q,b}$ (resp. $\tilde{\theta}_n^{s,b}$) sont les estimateurs du maximum de vraisemblance obtenus après l'apprentissage par le $MLP(p,k,d)$ (resp. $MLP(p,k^{'},d)$), pour $b \in \{1,...,B\}$.

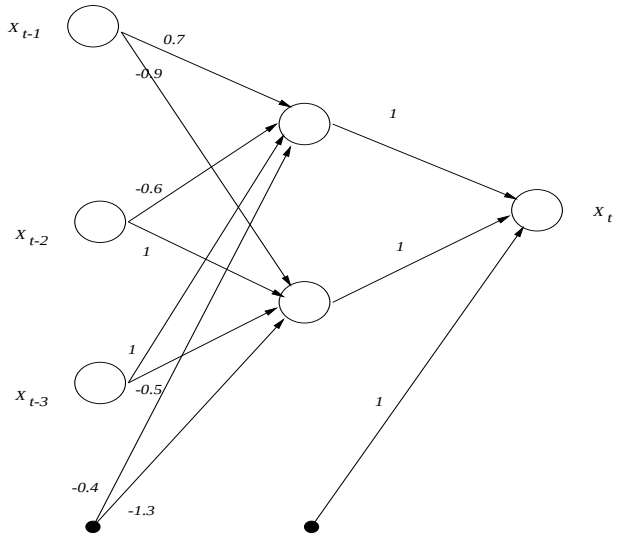


FIG. 3.1 – Le perceptron multicouches $MLP(3,2,1)$ des simulations.

Dans notre exemple de simulation :

➤ pour la construction des échantillons, nous avons pris un $MLP(p,k,d)$ avec $p=3, k=2, d=1$, $\epsilon \sim \mathcal{N}(0,\sigma^2)$, $\sigma^2 = 0.25$. Les coefficients sont fixés comme indiqués sur la figure Fig.3.1. La fonction tanh(.) est la fonction d'activation des unités cachées, et on a $q = 11$,

➤ nous avons testé ce modèle contre le modèle $MLP(p,k^{'},d)$ avec $k^{'} = 3$, pour lequel $s = 16$,

➤ nous avons répété la procédure **[P]** pour un nombre d'observations $n = 1000$, 10000 et 100000, et nous avons simulé $B = 100$ bases dans chaque cas.

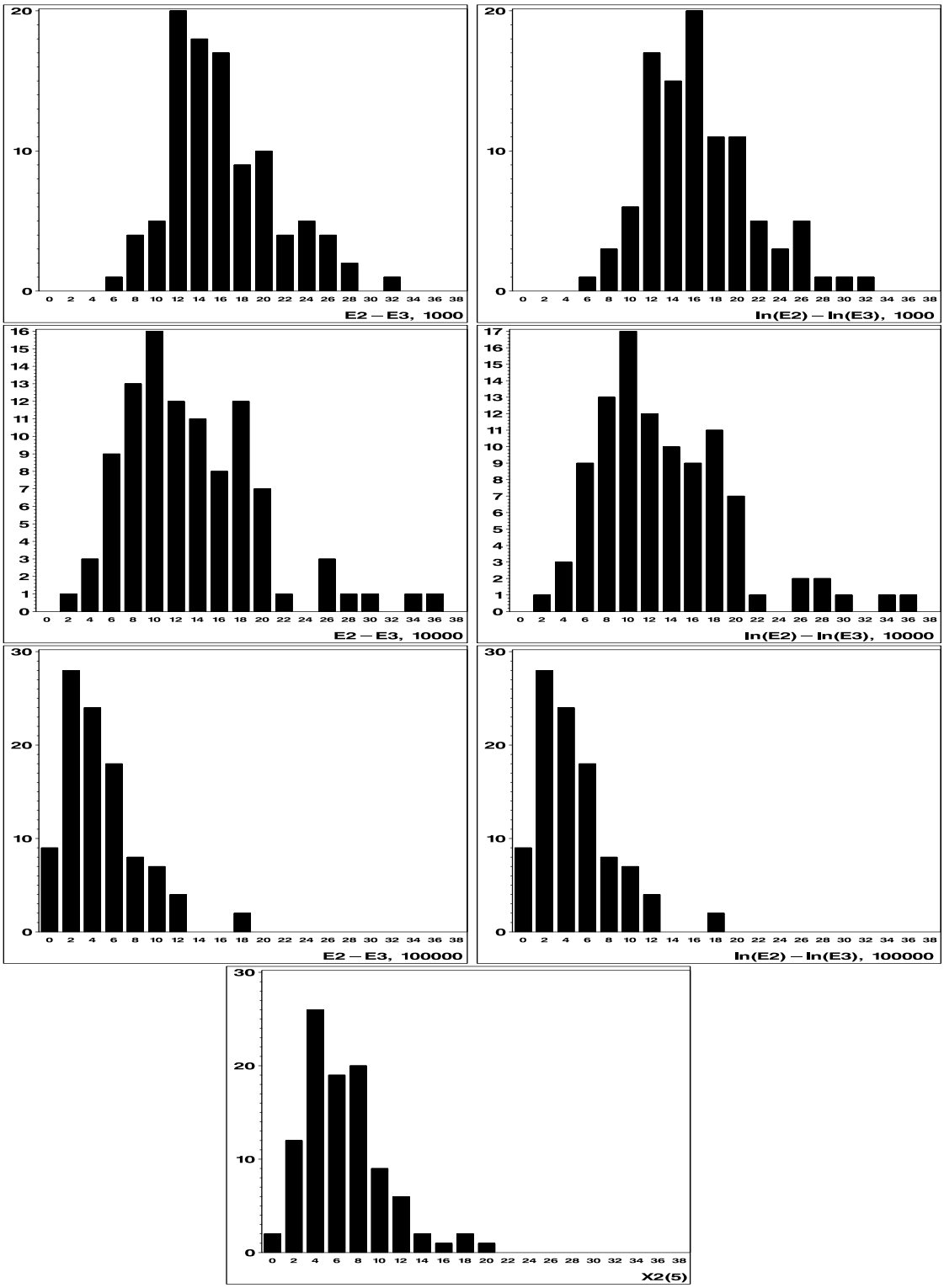


FIG. 3.2 – Première colonne : Histogramme de la variable $2D_n$ pour $n = 1000, n = 10000, n = 100000$. Seconde colonne : Histogramme de la variable DL_n pour les trois valeurs de n. La dernière ligne est l'histogramme d'une simulation d'une loi $\chi^2_{(5)}$.

Les distributions des deux variables $2D_n$ et DL_n devraient être, puisque l'hypothèse H_q est vraie, proche d'une loi de chi-deux à $s - q = 5$ degrés de liberté. Les histogrammes (Fig.3.2) obtenus montrent que nous avons bien une convergence vers la loi $\chi^2_{(5)}$ lorsque n croît, mais que la vitesse de convergence est très faible : nous retrouvons l'allure d'une loi $\chi^2_{(5)}$ seulement pour $n = 100000$. Si on utilise le test asymptotique de différence de contrastes (proposition 3) comme critère pour choisir entre les deux modèles, on choisit le modèle $MLP(3, 3, 1)$ dans 85% des cas pour $n = 1000$, et dans 50% des cas pour $n = 10000$ avec un niveau de confiance $\alpha = 95\%$, alors que le vrai modèle est le $MLP(3, 2, 1)$.

Or dans les applications avec des données réelles, nous n'avons pas toujours un très grand nombre d'observations. L'application de cette méthode telle quelle risque donc de conduire à un modèle de perceptron multicouches surparamétré si on se base sur la proposition 3 dans le cas de l'estimation par la méthode des moindres carrés, ou la proposition 4 dans le cas de l'estimation par la méthode du maximum de vraisemblance, pour faire le choix entre les perceptrons étudiés. Nous essayons donc d'appliquer une méthode de ré-échantillonnage appelée "bootstrap paramétrique" pour améliorer ce résultat, et construire plus précisement un test permettant de choisir entre deux modèles.

3.7 Bootstrap paramétrique

On peut résumer la méthode du bootstrap paramétrique, (différent du bootstrap non-paramétrique introduit jusqu'ici) dans une situation générale de la manière suivante. Nous avons des données $Y = (Y_1, ..., Y_n)$ pas nécessairement i.i.d.. P est un modèle statistique sous lequel ces données sont obtenues. Généralement P peut être décrit par la distribution jointe F de Y. Soit $R_n(Y, P)$ une variable aléatoire et supposons que nous voulions estimer sa distribution. La première étape est d'estimer le modèle P à l'aide des données Y. Soit ensuite Y^* un ensemble de données générées à partir du modèle estimé $\widehat{P}$. La distribution conditionnelle de $R_n(Y^*, \widehat{P})$ sachant Y, est alors l'estimateur bootstrap de la distribution de $R_n(Y, P)$.

Le bootstrap peut être appliqué à toutes les situations où un modèle P peut être estimé par $\widehat{P}$. Cependant ce qui différencie le bootstrap paramétrique du bootstrap non-paramétrique, c'est la façon dont on estime $\widehat{P}$ et dont on génère Y^*.

Dans le cas du bootstrap paramétrique, on suppose que la distribution F des données Y appartient à une famille paramétrique $F = F_\theta$ où θ est un vecteur pa-

ramètre inconnu. Dans le cas paramétrique, la première étape est donc d'estimer ce paramètre par $\widehat{\theta}$. (Dans le cas non-paramétrique, F est estimée par la distribution empirique de Y). L'ensemble des données bootstrap est alors généré grâce à $F_{\widehat{\theta}}$. On notera que, contrairement au bootstrap non-paramétrique, le bootstrap paramétrique dépend de la forme du modèle paramétrique.

Maintenant l'application du bootstrap paramétrique aux tests statistiques est aisée. En effet pour construire un test, on doit trouver une région de rejet W telle que sous l'hypothèse nulle (souvent notée H_0) on ait $P_{H_0}(W) = \alpha$ où α est appelé risque de 1ère espèce ou niveau du test. On rejette l'hypothèse nulle si et seulement si $Y \in W$.

Pour déterminer cette région de rejet, on utilise une statistique de test $T_n(Y)$ et la région de rejet est par exemple de la forme :

$$W = \{T_n(Y) > C\}$$

où C est une constante déterminée par la valeur de α. A moins que le problème de test ne soit particulièrement simple, il est difficile ou bien impossible de calculer exactement la constante C en fonction de α.

Une approche traditionnelle est d'utiliser la distribution asymptotique de $T_n(Y)$ sous l'hypothèse nulle. Cependant nous avons vu dans la section précédente, que cette approche peut être, par exemple dans le cas des MLP, extrêmement imprécise. C'est pourquoi nous allons utiliser la distribution bootstrap de $T_n(Y)$ pour déterminer plus précisement la constante "C". Nous emploierons donc la méthode du bootstrap paramétrique appliquée au cas : $R_n(Y, P) = T_n(Y)$ ou $R_n(Y, P) = TL_n(Y)$.

3.8 Application du bootstrap paramétrique au cas du perceptron multi-couches

Nous reprenons les mêmes notations que dans la section (3.6), et pour simplifier, nous formulons le test de la façon suivante :

$$\begin{cases} H_q & = & MLP(p, k, d) \\ H_s & = & MLP(p, k^{'}, d) \end{cases} \tag{3.18}$$

Nos simulations sont établies sous l'hypothèse H_q :

- nous avons repris l'exemple de la section (3.6) (Fig.3.1), c'est-à-dire un modèle $MLP(p,k,d)$ avec $p=3$, $k=2$ et $d=1$. La dimension $q=11$. Nous avons fait un apprentissage, par la méthode des moindres carrés, sur une base simulée de taille n, nous obtenons ainsi un estimateur $\hat{\theta}_n^q$ du paramètre du modèle et un estimateur $\hat{\sigma}_q^2 = U_n(\hat{\theta}_n^q)$ de l'erreur quadratique moyenne,

- sur la même base de données, nous avons fait un apprentissage, par la méthode des moindres carrés, avec le modèle $MLP(p,k^{'},d)$, $p=3$, $k^{'}=3$, $d=1$. Le nombre de paramètres à estimer est $s=16$. Nous obtenons ainsi un estimateur $\hat{\theta}_n^s$ du paramètre du modèle et un estimateur $\hat{\sigma_s^2} = U_n(\hat{\theta}_n^s)$ de l'erreur quadratique moyenne,

- nous avons calculé la statistique $T_n = 2n(U_n(\hat{\theta}_n^q) - U_n(\hat{\theta}_n^s))$,

- avec le paramètre estimé $\hat{\theta}_n^q$, et l'erreur quadratique moyenne estimée du modèle $U_n(\hat{\theta}_n^q)$ (Fig.3.3), nous avons simulé B bases suivant le modèle $MLP(p,k,d)$,

- pour chacun des B échantillons simulés, nous avons fait l'apprentissage par la méthode des moindres carrés, avec un perceptron dont l'architecture est la même que celle du modèle de simulation, c'est-à-dire $MLP(p,k,d)$, et calculé $\hat{\theta}_n^{q,b*}$, $U_n(\hat{\theta}_n^{q,b*})$, pour $b \in \{1,...,B\}$,

- nous avons fait l'apprentissage par la même méthode, avec un modèle de perceptron $MLP(p,k^{'},d)$, et calculé $\hat{\theta}_n^{s,b*}$, $U_n(\hat{\theta}_n^{s,b*})$, pour $b \in \{1,...,B\}$,

- nous avons calculé les statistiques $T_n^{b*} = 2n(U_n(\hat{\theta}_n^{q,b*}) - U_n(\hat{\theta}_n^{s,b*}))$ pour chaque base simulée. Nous avons obtenu ainsi le vecteur D_n^* de B observations :

$$D_n^* = (T_n^{b*})_{1\leq b\leq B} = (2n(U_n(\hat{\theta}_n^{q,b*}) - U_n(\hat{\theta}_n^{s,b*}))_{1\leq b\leq B},$$

- nous avons représenté l'histogramme du vecteur D_n^* et calculé son α-quantile $D^*{}_{n,\alpha}$,

- de la même manière, nous avons calculé la différence de contrastes associée au maximum de vraisemblance. Nous avons obtenu les statistiques

$$TL_n^{b*} = n(V_n(\tilde{\theta}_n^{q,b*}) - V_n(\tilde{\theta}_n^{s,b*})) = n\left(\log(\tilde{\sigma}_q^2) - \log(\tilde{\sigma}_s^2)\right),$$

le vecteur

$$DL_n^* = (TL_n^{b*})_{1 \leq b \leq B}$$

et son α-quantile $DL^*{}_{n,\alpha}$.

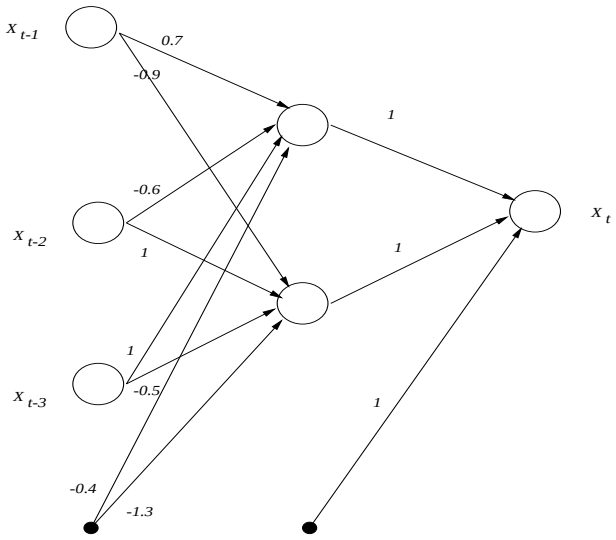


FIG. 3.3 – Le perceptron multicouches $MLP(3, 2, 1)$ estimé.

Comme dans la section (3.6), l'application est réalisée pour $n = 1000$, 10000, 100000 observations et un nombre de réplications $B = 100$.

Les histogrammes des deux figures (3.2) et (3.4) montrent que nous trouvons les mêmes distributions calculées pour les trois valeurs de n. En appliquant la proposition (3), nous accepterons l'hypothèse H_s dans 87% des cas pour $n = 1000$ et 55% des cas pour $n = 10000$. Mais, si nous remplaçons la distribution de la loi du test (la loi $\chi^2_{(5)}$) par les distributions des variables D_n^* et DL_n^* (Fig.3.4) pour faire le test, nous trouvons que les α-quantiles $D^*{}_{n,\alpha} > T_n$ et $DL^*{}_{n,\alpha} > TL_n$ dans les trois cas (Tab.3.1), c'est-à-dire que nous acceptons l'hypothèse H_q dans tous les cas. Ce qui nous permet de choisir le vrai modèle $MLP(3, 2, 1)$.

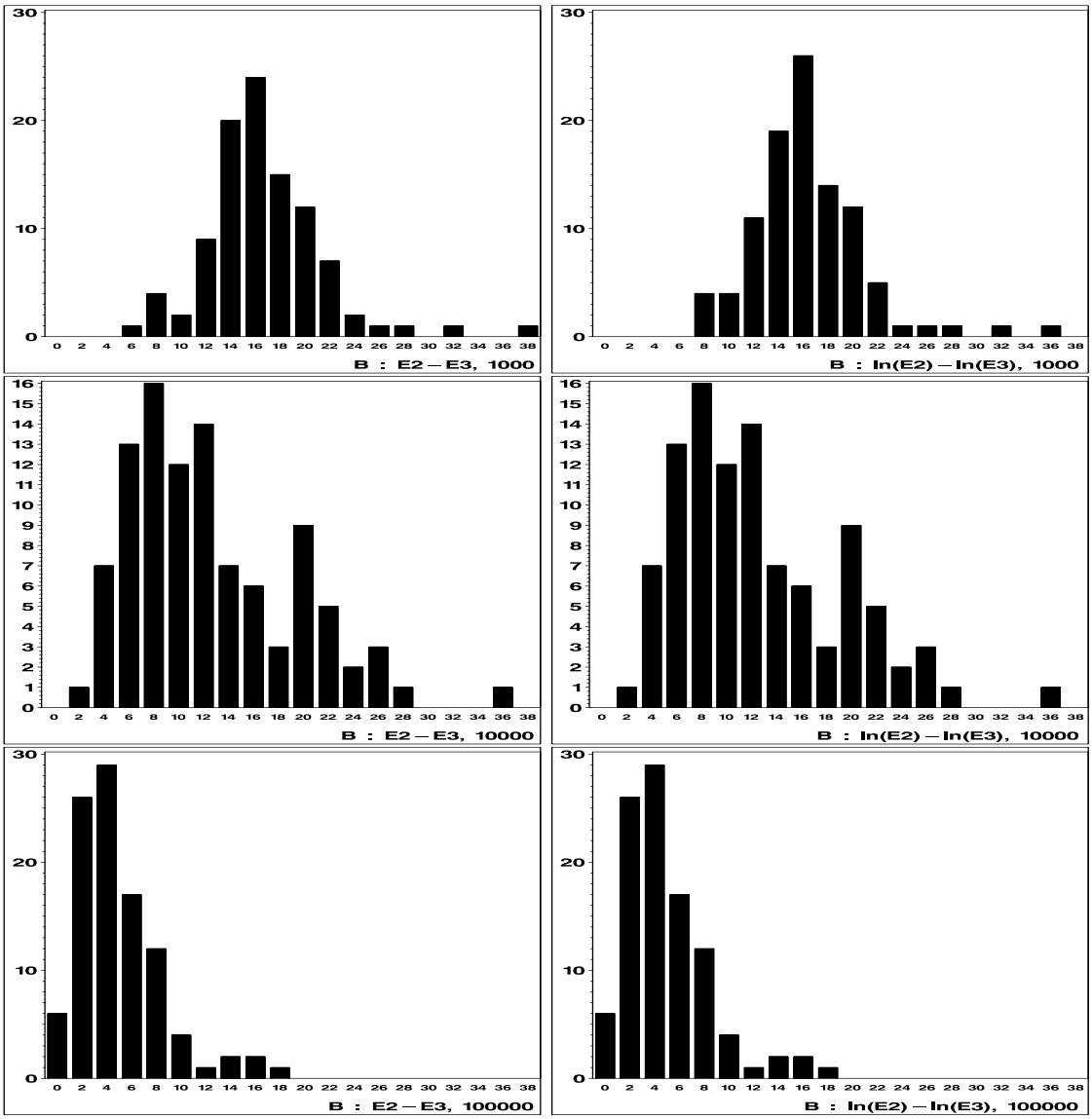


FIG. 3.4 – Première colonne : Histogramme de la variable D_n^* pour les trois applications du bootstrap paramétrique ($n = 1000, n = 10000, n = 100000$). Seconde colonne : Histogramme de la variable DL_n^* pour les mêmes valeurs de n.

n	$D^{*}{}_{n,\alpha}$	T_n	$DL^{*}{}_{n,\alpha}$	TL_n
1000	23.76	14.87	23.15	14.44
10000	24.56	15.04	24.43	14.83
100000	11.13	4.00	11.14	4.03

TAB. 3.1 – α-quantiles des deux variables D_n^* et DL_n^* et les statistiques T_n et TL_n, avec $\alpha = 95\%$.

3.9 Vérification sur des simulations

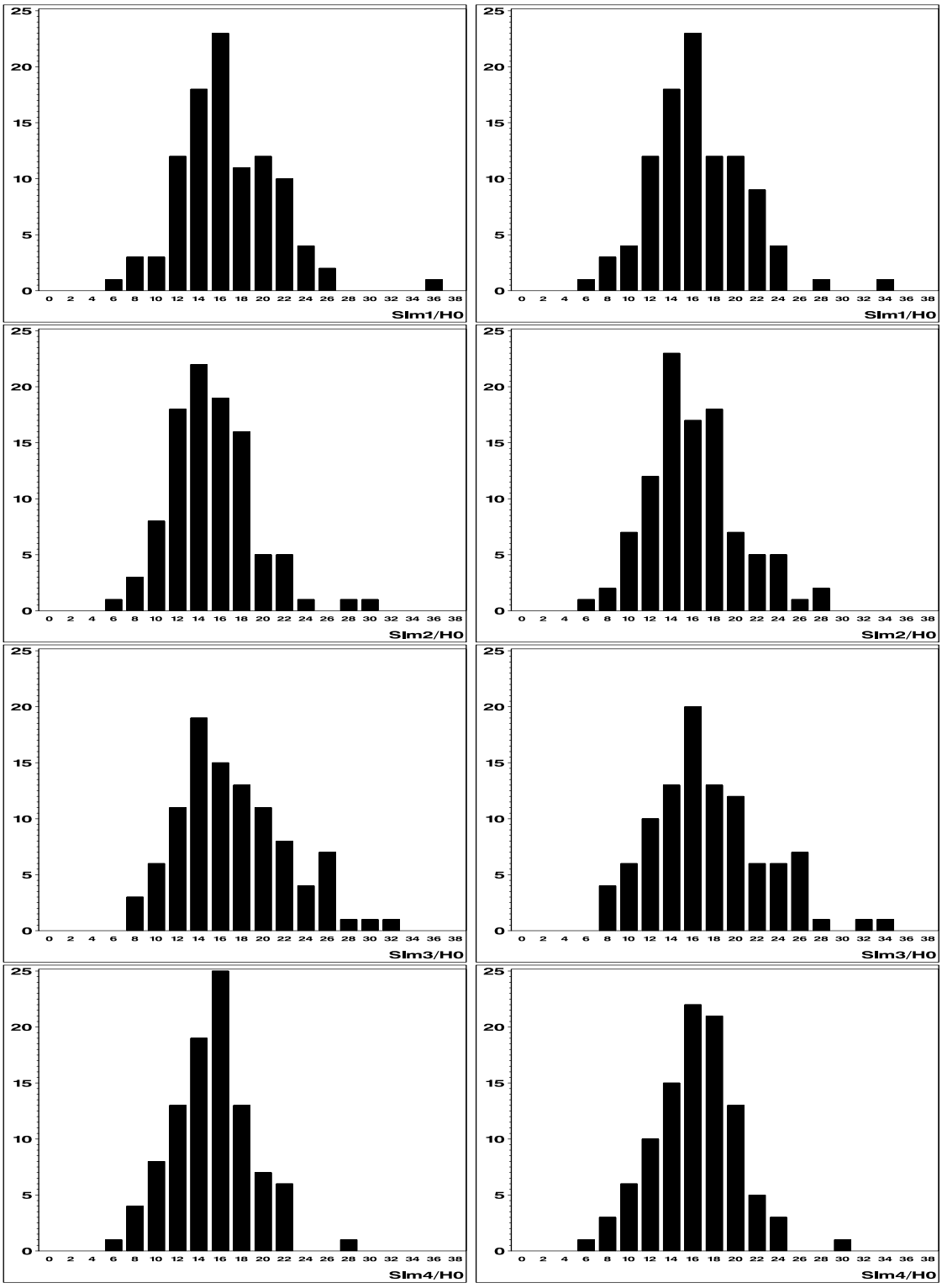
3.9.1 Vérification

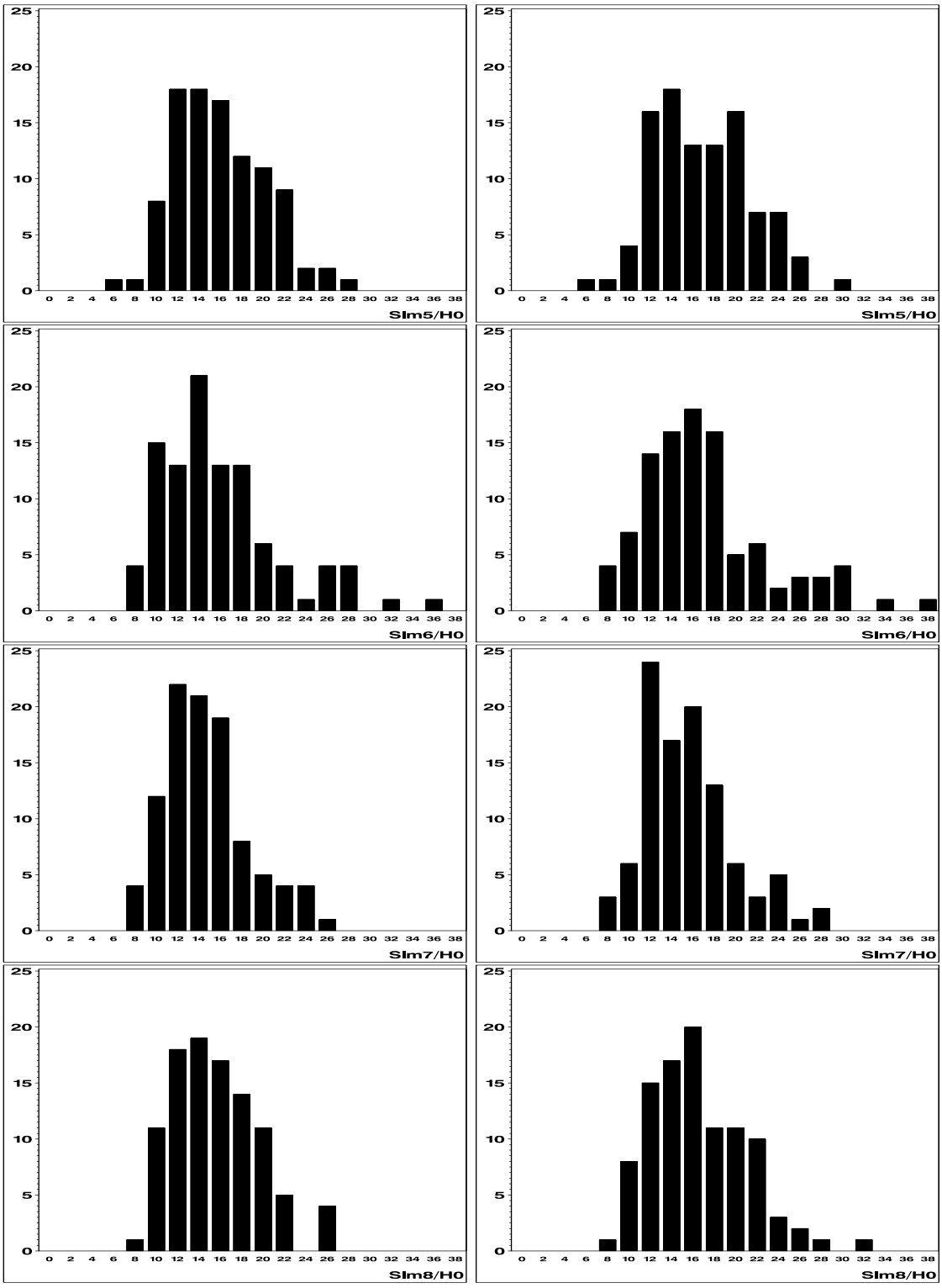
Dans le même cadre de l'application de la méthode du bootstrap paramétrique aux modèles neuronaux, nous essayons de vérifier les résultats du paragraphe précédent. Pour cela, nous avons appliqué la procédure de la partie précédente pour dix simulations indépendantes avec un nombre d'observations $n = 1000$. Les histogrammes de la figure (3.5) représentent les distributions des variables D_n^* et DL_n^* pour les 10 simulations. Nous remarquons que ces distributions sont les mêmes que celles de la première ligne des deux figures (3.2) et (3.4). Nous remplaçons alors la distribution du test asymptotique de différence de contrastes (la loi de $\chi^2_{(5)}$) par ces distributions (Fig.3.5). Nous trouvons que les inégalités $D^{*}{}_{n,\alpha} > T_n$ et $DL^{*}{}_{n,\alpha} > TL_n$ sont vérifiées dans toutes les simulations (Tab.3.2). Nous acceptons alors l'hypothèse H_q dans toutes les simulations, c'est-à-dire que nous choisissons le vrai modèle $MLP(3, 2, 1)$ dans tous les cas.

3.9.2 Puissance du test

Nous nous intéressons dans cette partie à la puissance du test de différence de contrastes réalisé par la méthode du bootstrap paramétrique. Nous avons fait exactement le même travail que dans le paragraphe 3.8, mais sous l'hypothèse H_s. C'est-à-dire que nous avons fait les simulations avec le $MLP(3, 3, 1)$ représenté ci-dessous.

Nous remarquons que les distributions des deux variables D_n^* et DL_n^* ressemblent à celles calculées dans le paragraphe précédent (Fig.3.5, Fig.3.7). Si on remplace la distribution du test de différence de contrastes par celles des variables D_n^* et DL_n^*, on constate que les inégalités $D^{*}{}_{n,\alpha} < T_n$ et $DL^{*}{}_{n,\alpha} < TL_n$ (Tab.3.3) dans toutes les simulations. Nous rejetons donc l'hypothèse H_q, et nous choisissons le vrai modèle $MLP(3, 3, 1)$.


Slm1/H0
Slm1/H0
Slm2/H0
Slm2/H0
Slm3/H0
Slm3/H0
Slm4/H0
Slm4/H0


Slm5/H0
Slm5/H0
Slm6/H0
Slm6/H0
Slm7/H0
Slm7/H0
Slm8/H0
Slm8/H0

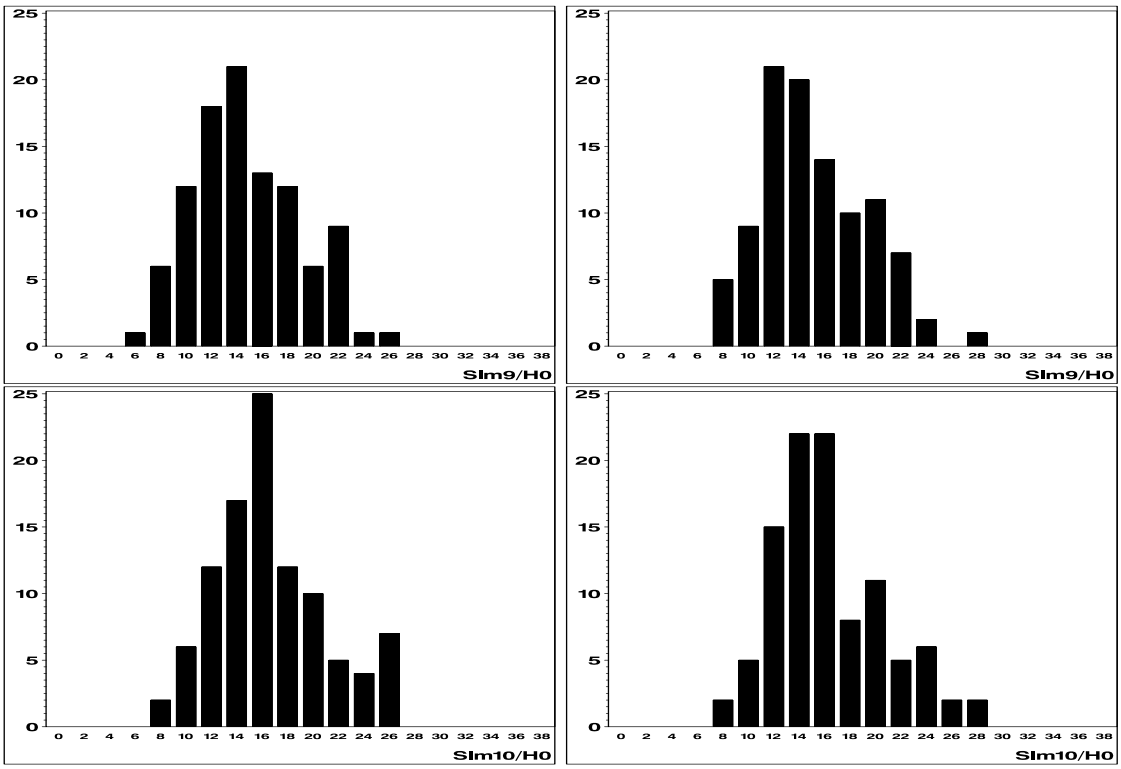


FIG. 3.5 – Première colonne : Histogramme de la variable D_n^* pour les dix simulations sous l'hypotèse H_q. Seconde colonne : Histogramme de la variable DL_n^* pour les mêmes valeurs de n.

Sous H_q	${D^*}_{n,\alpha}$	T_n	${DL^*}_{n,\alpha}$	TL_n
sim1	23.86	18.98	23.13	18.57
sim2	22.56	15.51	23.40	16.03
sim3	25.53	8.87	26.33	8.98
sim4	21.76	6.73	22.68	7.12
sim5	22.86	16.62	24.43	17.18
sim6	27.45	11.50	29.06	12.19
sim7	22.52	12.56	23.80	12.91
sim8	22.12	16.47	23.10	16.74
sim9	22.17	12.03	22.52	12.19
sim10	25.54	13.62	24.91	13.30

TAB. 3.2 – α-quantiles des deux variables D_n^* et DL_n^* et statistiques T_n et TL_n, avec $\alpha = 95\%$ et $n = 1000$.

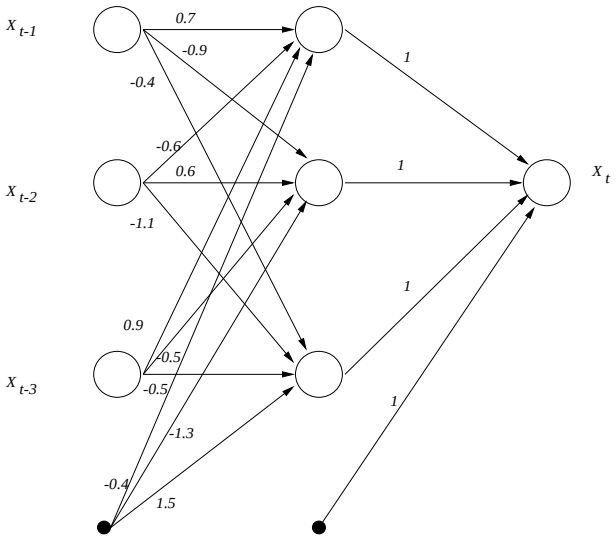
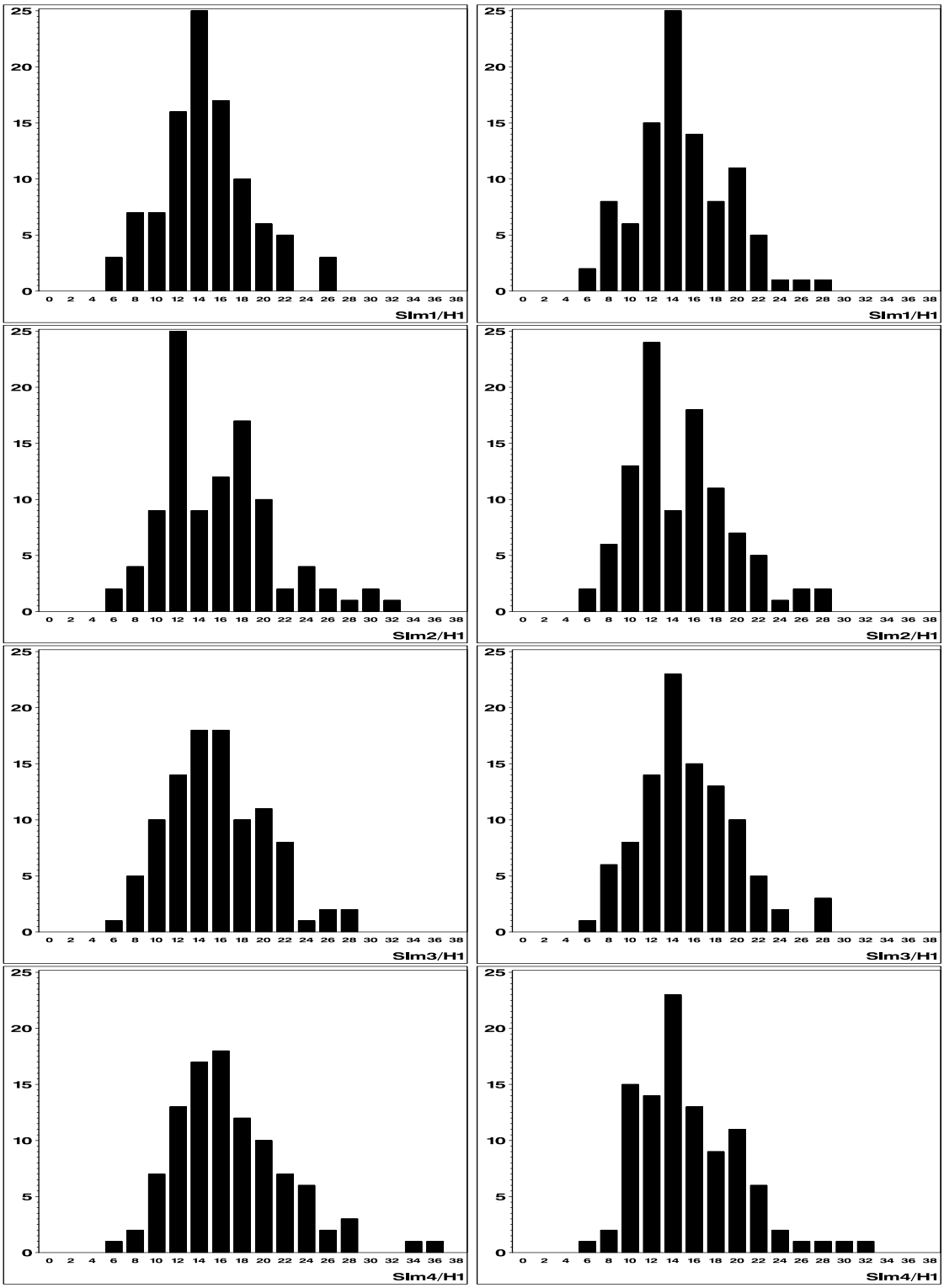
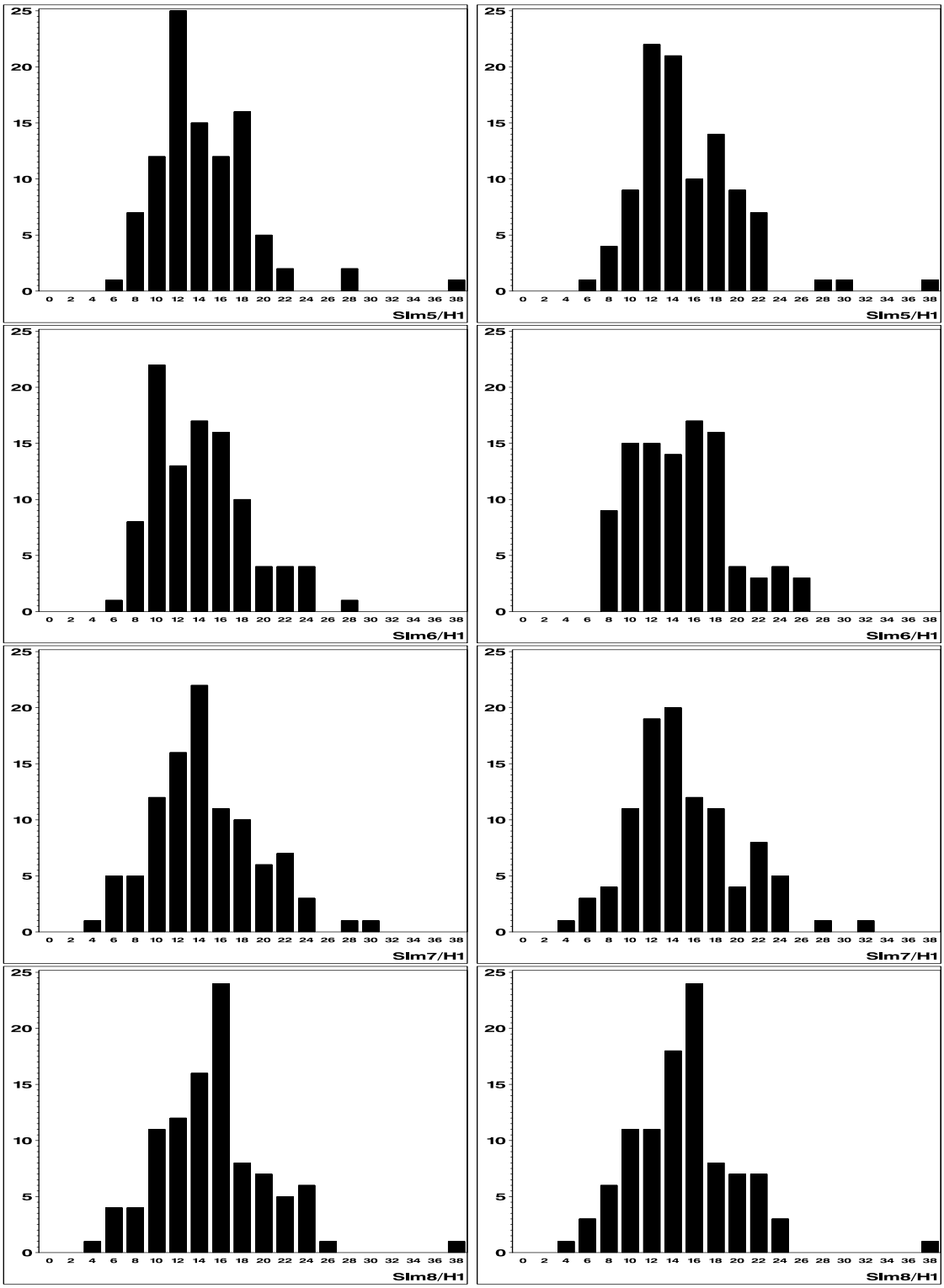


FIG. 3.6 – Le perceptron multicouches $MLP(3,3,1)$.

Sous H_s	${D^*}_{n,\alpha}$	T_n	${DL^*}_{n,\alpha}$	TL_n
sim1	22.07	57.70	21.73	58.26
sim2	25.62	48.83	22.89	45.89
sim3	22.34	66.80	22.77	65.92
sim4	25.70	86.40	24.12	79.84
sim5	20.70	48.60	22.35	52.31
sim6	21.82	44.97	23.16	48.02
sim7	23.57	45.33	22.72	45.93
sim8	21.59	83.52	22.58	83.61
sim9	21.59	69.70	22.57	71.62
sim10	23.50	86.34	22.36	81.64

TAB. 3.3 – α-quantiles des deux variables D_n^* et DL_n^* et les statistiques T_n et TL_n sous l'hypothèse H_s, avec $\alpha = 95\%$ et $n = 1000$.


Slm1/H1
Slm1/H1
Slm2/H1
Slm2/H1
Slm3/H1
Slm3/H1
Slm4/H1
Slm4/H1


Slm5/H1
Slm5/H1
Slm6/H1
Slm6/H1
Slm7/H1
Slm7/H1
Slm8/H1
Slm8/H1

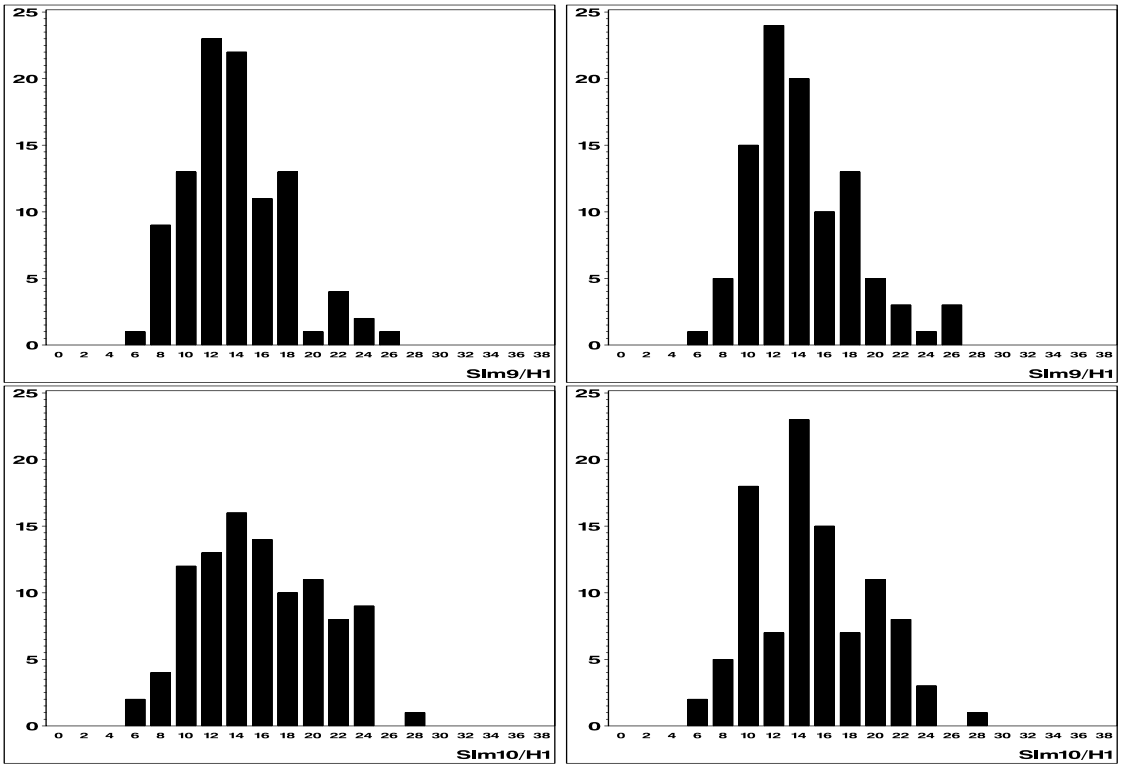


FIG. 3.7 – Première colonne : Histogramme de la variable D_n^* pour les dix simulations sous l'hypothèse H_s. Seconde colonne : Histogramme de la variable DL_n^* pour les mêmes valeurs de n.

3.10 Consistance du bootstrap paramétrique

Reprenons les notations de la première partie, et considérons un modèle autorégressif fonctionnel $ARF_d(p)$. D'après la proposition (3), on a

$$T_n = 2n[U_n(\hat{\theta}_n^q) - U_n(\hat{\theta}_n^s)] \xrightarrow{C.L} \sum_{i=1}^{s-q} \lambda_i \chi_{i,1}^2,$$

où les $\chi_{i,1}^2$ sont $s-q$ variables aléatoires indépendantes de loi χ_1^2, et où les λ_i sont des valeurs positives.

Pour $d = 1$, on a A_0 est de rang $s-q$ avec $\lambda_i = 2\sigma^2$ pour $1 \leq i \leq s-q$. On trouve alors un test du χ^2 à $s-q$ degrés de liberté. C'est-à-dire

$$T_n = \xrightarrow{C.L} 2\sigma^2 \chi_{s-q}^2.$$

Pour un perceptron multicouche $MLP(p, k, 1)$, l'hypothèse de stabilité **[S]**, l'hypothèse de continuité **[M]**, l'hypothèse d'identifiabilité **[D]** et l'hypothèse **[N]**, sont bien vérifiés si $a > 6$. On pourra donc appliquer la proposition (3) sur une base construite par l'application de la méthode du bootstrap paramétrique, pour $B = 1$ et sous l'hypothèse H_q, On aura alors

$$T_n^* = 2n[U_n(\hat{\theta}_n^{q*}) - U_n(\hat{\theta}_n^{s*})] \xrightarrow{C.L} 2\hat{\sigma}^2 \chi_{s-q}^2.$$

Comme $\hat{\sigma}^2 \xrightarrow{p.s} \sigma^2$, et en appliquant le théoreme de Slutsky

Théorème 9 *Slutsky*

Si $X_n \xrightarrow{C.L} X$, $Y_n \xrightarrow{pr} c^{te}$ *et* $f(.,.)$ *est une fonction continue, alors*

$$f(X_n, Y_n) \xrightarrow{C.L} f(X, c^{te}),$$

on a le résultat suivant :

Lemme 2 *Sous les hypothèses de la proposition 3, on a*

$$T_n^* \xrightarrow{C.L} 2\sigma^2 \chi_{s-q}^2.$$

Donc, dans ce cas, le bootstrap paramétrique est consistant puisqu'il est équivalent au test théorique. Dans le cas d'un modèle autorégressif fonctionnel $ARF_d(p)$ avec $d > 1$, nous ne pouvons appliquer directement la méthode du bootstrap paramétrique que si on a une paramétrisation de la loi du bruit, car la loi du modèle $X_t^{(p)}$ sous la vraie valeur θ_0 du paramètre dépend de la loi de l'erreur du modèle ϵ. Par exemple, dans le cas où le bruit est gaussien, on a le résultat suivant :

Lemme 3 *Sous les hypothèses de la proposition 4, on a*

$$TL_n^* = n(V_n(\tilde{\theta}_n^{q*}) - V_n(\tilde{\theta}_n^{s*})) \xrightarrow{C.L} \chi^2_{s-q}.$$

Ce qui induit aussi la consistance du bootstrap paramétrique dans le cas de l'utilisation de la méthode du maximum de vraisemblance.

3.11 Conclusion

Nous avons traité dans ce chapitre l'application du bootstrap paramétrique au test de différence de contrastes. Dans cette application, nous avons utilisé des perceptrons multicouches avec une sortie unidimensionnelle. Les simulations traitées montrent l'efficacité et la puissance du test de la méthode. Nous avons montré aussi la consistance du bootstrap paramétrique appliqué aux modèles auto-régressifs non linéaires dont la variable de sortie est de dimension 1.

Dans le cas où la dimension de la variable de sortie est supérieure à 1, et si on connaît la loi invariante du modèle $ARF_d(p)$, l'application est évidente. Dans le cas contraire, il est nécessaire d'étudier les variables d'entrées et de sorties du modèle. Dans ce cas précis, l'application du bootstrap paramétrique au modèle $ARF_d(p)$ reste un problème ouvert.

Il est à noter, que malgré les performances du bootstrap (paramétrique ou non), cette méthode nécessite un temps de calcul important dans son application aux perceptrons multicouches. Ce temps est dû aux algorithmes d'apprentissage.

Deuxième partie

Localisation d'un véhicule en mouvement sur une voie routière

Chapitre 4

Application du Filtre de Kalman étendu à la localisation d'un véhicule

4.1 Introduction

L'étude menée en collaboration avec le LIVIC[1] (Laboratoire sur les Interactions Véhicules-Infrastructure-Conducteurs) a pour objectif la *localisation*, en coordonnées géographiques, d'un véhicule en mouvement sur une voie autoroutière à partir de capteurs embarqués. On souhaite avec ces moyens développer un système d'aide à la conduite.

La fiabilité mécanique des automobiles est parvenue à un degré très important et satisfaisant. Le problème actuel n'est plus seulement d'augmenter cette fiabilité, mais de pallier les éventuelles déficiences du conducteur. Ainsi, la voiture doit désormais être capable de "détecter" une situation dangereuse pour le conducteur, et éventuellement de "prendre le contrôle" pour éviter l'accident[2]. Ce cahier des charges, très simple apparemment, nécessite en fait toute une série d'opérations extrêmement complexes à réaliser.

Le souci grandissant des constructeurs automobiles d'améliorer la sécurité à bord de leurs automobiles les a donc amenés à réfléchir sur la possibilité d'exploiter les données issues de caméras ou de GPS, traitées informatiquement.

[1] implanté à Satory, Versailles

[2] Cette approche marque une rupture avec la démarche des années 80, qui se concentrait avant tout sur le conducteur. Citons à titre d'exemple le projet *Prometheus* (1986), le projet *CASSIE* (*CA*ractérisation *S*ymbolique de *SI*tuations de conduit*E*)(1999).

Dans ce chapitre, nous allons traiter de la localisation d'un véhicule à partir de données peu précises (fournis par un système GPS) et de données plus précises mais relatives (fournis par un odomètrie et un gyroscope). Le GPS (*Global Positionning System*), appelé DGPS (*Differential Global Positionning System*) en fonctionnement différentiel, est un capteur recevant des signaux provenant de satellites et qui informe sur la position ; l'*odomètre* informe sur la vitesse *linéaire* ou la distance parcourue ; le *gyroscope* informe sur la vitesse *angulaire* ou la variation de l'angle de rotation. Les principales difficultés de cette étude viennent de

① la faible précision du GPS,
② du *calibrage* de l'odomètre et du gyroscope,
③ et de la *synchronisation* entre les différents capteurs.

Le niveau de précision requis pour la localisation ne peut en effet être atteint qu'au prix d'un *recalibrage préliminaire* de l'odomètre et du gyroscope, bien qu'ils aient été présentés par leurs constructeurs comme parfaitement calibrés. Un problème de synchronisation a par ailleurs été mis en évidence (cf. Figure 4.7).

La localisation du véhicule consiste ici en une opération d'intégration des données collectées par ces trois capteurs en vue d'extraire une *nouvelle information géographique*, plus représentative de l'ensemble des données initiales. Elle doit permettre de compenser les limites d'un capteur par les observations d'un ou de plusieurs autres. Elle doit aussi permettre d'exploiter à la fois les effets de redondance et de complémentarité de ces informations.

Dans un premier temps, nous expliquons le fonctionnement et les caractéristiques du GPS et de la centrale inertielle. Nous décrirons ensuite la méthode du Filtre de Kalman qui permet le suivi de la localisation grâce à l'odométrie et enfin nous présenterons les résultats obtenus.

4.2 Description des capteurs

Les essais expérimentaux sont réalisés à partir d'une voiture `Renault Mégane Scenic` par une équipe technique du LIVIC. Une piste d'essai, dont l'allure est dessinée à l'échelle (en mètre) sur la figure 4.1, est mise à la disposition du laboratoire pour faire ces essais. Les capteurs embarqués pour la localisation sont de deux sortes :

- Un *capteur extéroceptif* : GPS *(Global Positionnig System)*, qui permet la détermination instantanée de la position et de la vitesse "partout" dans un référentiel absolu. Cependant, il présente deux inconvénients : la position n'est disponible qu'à une cadence d'une seconde et la précision de cette position est dégradée par de nombreuses perturbations.

- Les *capteurs inertiels* (INS abrégé de *Inertial Navigation System*) : un odomètre et un gyroscope qui mesurent les variables dynamiques du véhicule, à partir desquelles il est possible de reconstruire un modèle de trajectoire du véhicule. Les principaux inconvénients de ces capteurs sont les dérives des mesures : ils sont précis à court terme et inexacts à long terme.

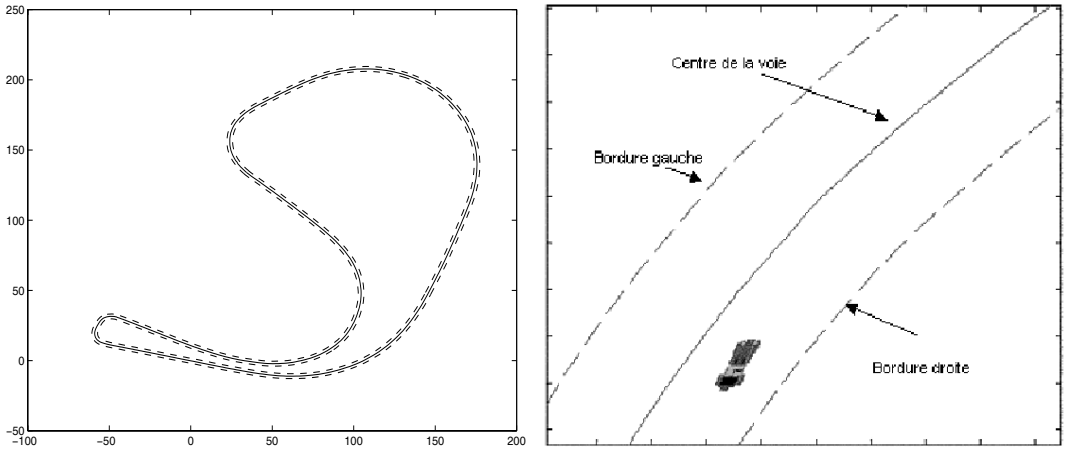


FIG. 4.1 – Allure de la piste d'essai du LIVIC.

4.2.1 GPS et DGPS

Principe de fonctionnement

Le système GPS est un système de radio-navigation constitué de 24 satellites et 5 bases au sol tels qu'à tout instant au moins 3 d'entre eux soient "visibles" de n'importe quel point de la surface du globe. Chaque satellite émet un signal sinusoïdal codé contenant sa propre position et la date précise d'émission de l'appel, de façon à ce que le récepteur GPS recevant le code, puisse reconstruire la position exacte à

partir du temps de vol de l'onde. Ce système fournit par *triangulation* à l'utilisateur sa position en longitude, en latitude et en altitude. Si la distance qui sépare un satellite du récepteur GPS est d_1 à un instant donné, nous savons que nous nous trouvons, à ce moment, sur une sphère de rayon d_1 dont le centre est le satellite. Si la distance au second satellite est d_2, nous pouvons affirmer que nous sommes aussi sur une sphère de rayon d_2. L'intersection de ces deux sphères est un cercle, dont tous les points ne sont pas physiquement possibles (certains de ces points ne sont pas sur la surface terrestre). Avec une troisième mesure d_3, l'intersection se réduit à deux points (Figure 4.2). Une quatrième mesure serait théoriquement nécessaire pour lever l'indétermination en donnant l'altitude, mais généralement, un des deux points est physiquement impossible (il n'est pas sur la surface terrestre).

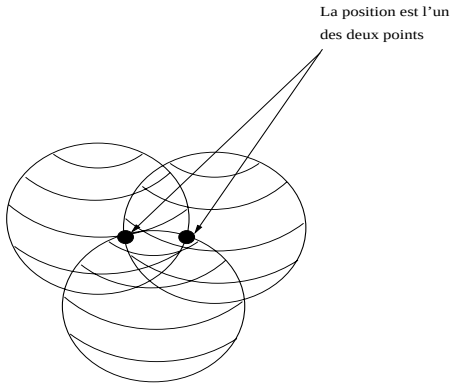


FIG. 4.2 – Mesure avec trois satellites.

L'altitude obtenue avec le système GPS est mesurée au dessus de l'ellipsoïde de référence et non par rapport au niveau moyen des mers. Dans le cadre de ce travail, on néglige les effets de l'altitude.

On obtient ainsi une précision de quelques mètres. Comme la longueur d'onde du signal varie de 1 à 10 mètres selon le code utilisé, le système permet en principe un positionnement entre 1 mètre et 10 mètres près. Avec des formes avancées d'utilisation du GPS, la précision peut atteindre le centimètre. En fait, il existe encore de grandes différences entre la précision théorique et les résultats obtenus, que nous étudions dans la section suivante.

Nature du signal

Le véhicule est équipé d'un GPS `Trimble Lassen SKII` qui enregistre le code de positionnement standard (x, y) (*Coarse/Acquisition*) à une cadence de 1 Hz. Il envoie les coordonnées géographiques dans le système. La précision du GPS reste assez compliquée à évaluer du fait de la protection industrielle qui couvre le système. En général, on parle d'une erreur de 10 mètres sans avoir une idée claire de la signification de cette incertitude. D'ailleurs le système GPS est un système de navigation qui n'était pas conçu à l'origine pour des applications de précision en temps réel (du moins pour les utilisateurs civils). Les récents développements des processeurs permettent néanmoins d'obtenir un positionnement plus précis en corrigeant les effets perturbateurs. On peut augmenter la précision par *fonctionnement différentiel* (DGPS), en utilisant un récepteur fixe qui définit le centre d'un repère dit par abus de langage *repère* DGPS. La figure 4.3 montre une partie de la piste reconstruite par le GPS et le DGPS.

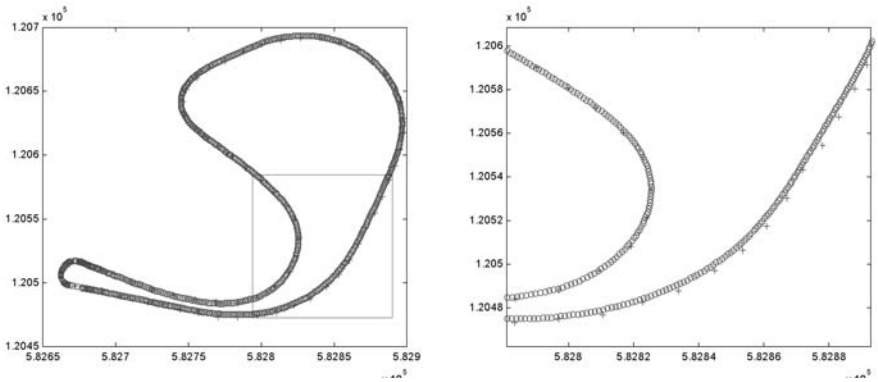


FIG. 4.3 – Trajectoire du véhicule mesurée par le DGPS(∘) et le GPS(+).

Le DGPS est un capteur plus sophistiqué qui permet d'augmenter la cadence et la précision des données. Malheureusement son coût (de l'ordre de 45 kilo euros) ne permet pas de l'intégrer pour le moment comme *capteur embarqué*.

Précisons par ailleurs que les conditions expérimentales de nos essais sont plutôt favorables au sens où, dans toutes nos expériences :

- ➤ il était possible de détecter au moins 7 satellites, garantissant ainsi une bonne configuration géométrique,
- ➤ la piste d'essai bénéficiait d'une grande visibilité du point de vue satellitaire.

Test de calibration GPS

Nous avons procédé à une phase de calibration du GPS afin de mettre en évidence la nature de ses signaux. Ces mesures mettent en évidence la nature de l'erreur. On voit immédiatement que l'on ne peut pas considérer cette erreur comme blanche. En effet, il existe une très forte corrélation entre deux mesures consécutives, comme le montre la figure 4.4 qui représente en ordonnée la valeur de $x_k - x_r$, où x_k est la position fournie par le GPS à l'instant k et x_r est la position réelle. L'abscisse correspond au temps. Ces mesures sont prises pour un véhicule à l'*arrêt*.

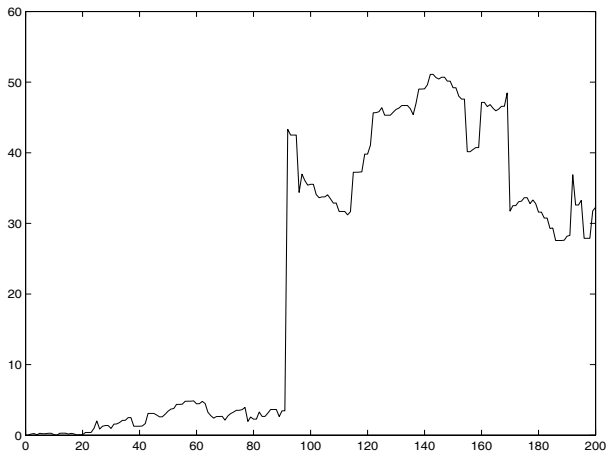


FIG. 4.4 – Mesure de la position d'un véhicule à l'arrêt en fonction du temps.

Cette dérive de la position provient des dégradations des performances du GPS. Nous allons vérifier plus rigoureusement les propriétés de *normalité* et de *blancheur* du signal. Dans ce but, une expérience a été menée à une fréquence d'échantillonnage de 1 Hz.

Mesure de l'autocorrélation La mesure d'autocorrélation permet de vérifier la *blancheur* du signal. Un estimateur empirique biaisé de la fonction d'autocorrélation d'un signal X, défini par la série d'observation $x_1, x_2, \ldots, x_N$, est calculée par

$$\hat{R}_X(m) = \frac{1}{N} \sum_{i=1}^{N-m} x_{i+m} x_i. \tag{4.1}$$

On notera que $\hat{R}_X(m) \leq \hat{R}_X(0)$. La figure 4.5 représente l'autocorrélation normalisée $\frac{\hat{R}_X(m)}{\hat{R}_X(0)}$ en fonction du temps en minutes. On peut en déduire qu'en dessous de 5 minutes, les données sont très corrélées.

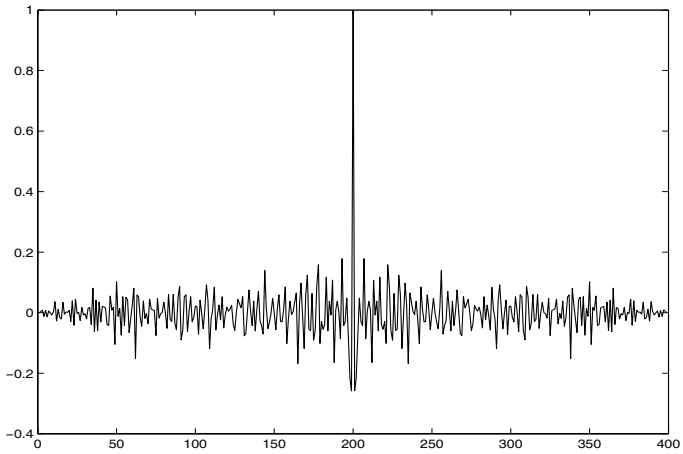


FIG. 4.5 – Autocorrélation sur une demi-journée.

Etude de la normalité Pour obtenir une estimation de la distribution de l'erreur de positionnement, on trace l'histogramme des valeurs mesurées (FIG. 4.6). On peut aussi vérifier le caractère gaussien de notre distribution en utilisant un *test de normalité* sur les moments d'ordre supérieur.

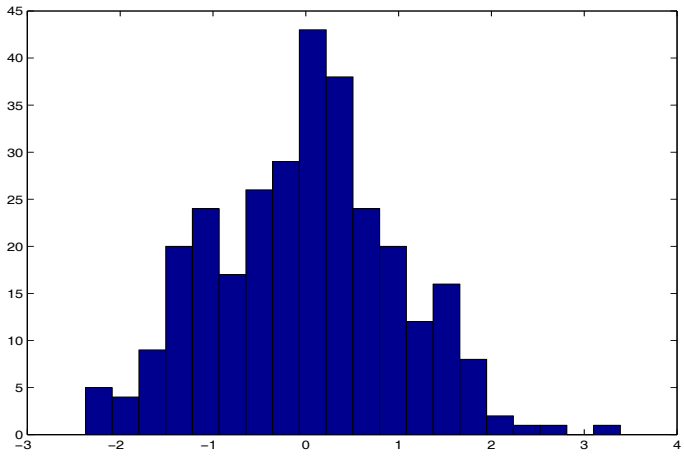


FIG. 4.6 – Distribution de probabilité.

On regarde plus précisemment les coefficients d'asymétrie γ_1 et d'applatissement γ_2

$$\gamma_1 = \frac{E[(X - E[X])^3]}{\mathrm{Var}[X]^{\frac{3}{2}}} \tag{4.2}$$

et

$$\gamma_2 = \frac{E[(X - E[X])^4]}{\mathrm{Var}[X]^2}. \tag{4.3}$$

Chacun de ces coefficients devrait être nul, puisque le moment d'ordre 3 d'une gaussienne est nul et que le moment d'ordre 4 vaut $3\sigma^2$. En utilisant les estimations empiriques suivantes

$$\hat{\gamma}_1 = \frac{1}{N}\sum_{i=1}^{N} x_i^3, \tag{4.4}$$

et

$$\hat{\gamma}_2 = \frac{1}{N}\sum_{i=1}^{N} x_i^4 - 3 \tag{4.5}$$

les x_i étant les mesures de position centrées réduites et N le nombre d'échantillons. L'hypothèse de normalité est rejetée avec une certitude de 95% si $T_2 = \frac{|\hat{\gamma}_1 - m_1|}{2\sqrt{v_1}} > 1$ ou si $T_2 = \frac{|\hat{\gamma}_2 - m_2|}{2\sqrt{v_2}} > 1$, m_1, m_2, v_1, v_2 étant définies comme les moyennes et les variances des estimateurs de γ_1 et de γ_2. Les valeurs de ces coefficients sont

$$m_1 = 0, \tag{4.6}$$

$$m_2 = -\frac{6}{N+1}, \tag{4.7}$$

$$v_1 = \frac{6(N-2)}{(N+1)(N+3)}, \tag{4.8}$$

$$v_2 = \frac{24N(N-2)(N-3)}{(N+1)^2(N+3)(N+5)}. \tag{4.9}$$

Dans notre cas, on trouve $\hat{\gamma}_1 = -0.026672$ et $\hat{\gamma}_2 = -0.44192$, donc $T_1 = 5.789$ et $T_2 = 13.61248$. On en conclut que les données ne sont pas gaussiennes, au niveau de confiance 95%. Pour obtenir des données respectant les hypothèses du filtre de Kalman (blancheur et normalité), certains auteurs sous-échantillonnent les mesures pour faire disparaître les corrélations dans le signal [68]. Parfois même, dans la littérature, ces données sont présentées comme blanches et gaussiennes [14]. En fait, étudier la localisation par GPS doit commencer par la recherche de la nature des signaux et leur calibrage, avant d'utiliser le filtre de Kalman.

4.2.2 Odomètre

Un odomètre consiste à mesurer les déplacements à partir d'une information sur la vitesse de rotation des roues. Elle combine un modèle de prédiction de la configuration avec des mesures de l'entrée de ce modèle et d'une partie de l'état. Les capteurs `DM5E` utilisés sont ceux installés par le système d'assistance au freinage ABS (*Anti Blocking System*). Chaque capteur fournit un signal carré à deux états dont la fréquence dépend de la vitesse de rotation de la roue. Son fonctionnement consiste à coder l'information du nombre de tours de la boîte de vitesse du véhicule, donnant ainsi la vitesse linéaire du véhicule *et* la distance parcourue. Il délivre une impulsion à pas constant de 100 Hz. Cette période a été calibrée préalablement par les essais.

4.2.3 Gyroscope

Il existe plusieurs gammes de gyroscope : du gyroscope monoaxial aux centrales inertielles, selon le principe physique qu'ils mettent en œuvre. Le gyroscope utilisé dans ce travail est un gyroscope monoaxial `VSG 29965x-0100` installé dans le véhicule pour détecter la vitesse de lacet. Le principe du gyroscope est celui d'un capteur vibrant à variation de direction : sachant qu'un élément vibrant tend à se maintenir dans son plan de rotation, il s'agit de mesurer l'accélération de Coriolis due aux oscillations provoquées par un capteur tournant autour de l'axe perpendiculaire à la direction de cette accélération[3]. Les accélérations de Coriolis agissent sur un élément céramique vibrant du gyroscope, ce qui induit des voltages proportionnels aux rotations. Les caractéristiques du gyroscope utilisé sont les suivantes :

Caractéristiques du gyroscope	
Fréquence :	8341 Hz
Plage de mesure :	100 deg/sec
Biais :	0.004 deg/sec
Facteur d'échelle :	19.73 mv/deg/sec

Le gyroscope dispose de deux modes de fonctionnement. Le premier délivre directement la vitesse angulaire ω par rapport à l'axe sensible, *i.e.* la perpendiculaire à la base du véhicule. Le second mode de fonctionnement consiste à intégrer la vitesse angulaire pendant un laps de temps. Cette intégration nous donne une variation angulaire $\Delta\theta$. L'inconvénient de ce mode de fonctionnement tient à ce qu'on perd le caractère absolu par rapport à un référentiel fixe. Pour nos essais, nous avons choisi

[3] On sait que l'accélération de Coriolis est donnée par $a = 2\omega v \sin\theta$, où ω est la vitesse angulaire, v la vitesse oscillatoire, θ l'angle entre les vitesses angulaire et oscillatoire.

le premier mode avec une fréquence comprise entre 20 Hz et 2000 Hz.

On notera que les deux capteurs *proprioceptifs* sont synchronisés pendant la collecte de données, l'impulsion délivrée par l'odomètre pilotant l'acquisition du gyroscope. Ceci permet d'obtenir une fréquence d'acquisition des données synchrone pour les deux capteurs.

Test de calibration sur l'odomètre et le gyroscope

Pour calibrer les deux capteurs odomètre et gyroscope, nous avons été amenés à simuler la vitesse linéaire et la vitesse angulaire à partir du DGPS. On obtient ainsi la même fréquence que le DGPS pour les deux vitesses, à savoir 10 Hz.

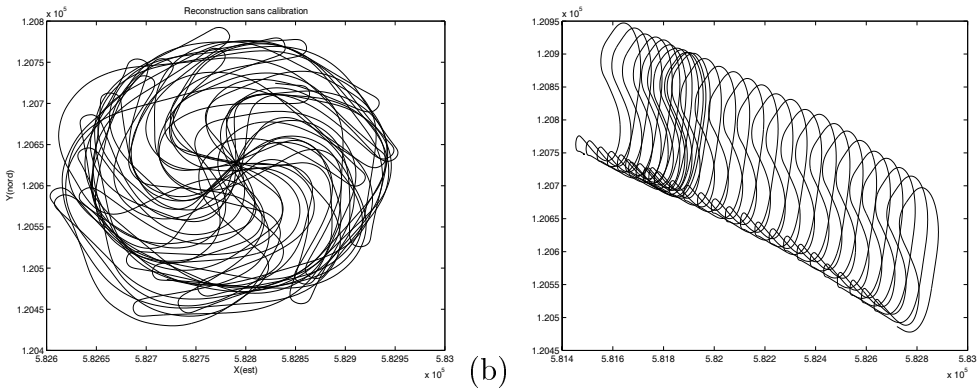


FIG. 4.7 – (a) Reconstruction de 30 tours de piste avec les données gyroscopique et odomètrique non calibrées. (b) Reconstruction de 30 tours de piste avec les données calibrées de ces deux capteurs.

La figure 4.7 compare l'utilisation des données brutes (Fig. 4.7.a) et l'utilisation des données calibrées (Fig. 4.7.b) dans la reconstruction de la piste.

➤ Sur la figure 4.7.a, la reconstruction de 30 tours de piste avec les données gyroscopique et odométrique *avant calibrage* met clairement en évidence la dérive des deux capteurs au cours du temps.

➤ La figure 4.7.b présente la reconstruction des 30 tours de piste avec les données gyroscopique et odométrique *après calibrage*.

Nous remarquons que, malgré ce calibrage, il n'est toujours pas possible de reconstruire exactement la piste, principalement au niveau des virages.

4.2.4 Modèle de trajectoire

Les systèmes inertiels de navigation discrétisent la trajectoire du véhicule. La position du véhicule est déterminée à partir des coordonnées (x, y) de sa position antérieure fournie par le GPS et des valeurs des variables fournies par les capteurs : la vitesse linéaire v, la vitesse angulaire ω qui permet de calculer la variation angulaire $\Delta\theta$. Les coordonnées (x_k, y_k) du point M_k à l'instant k sont alors exprimées dans le repère du DGPS. La reconstruction de la trajectoire (Fig.4.8) est le résultat de la boucle suivante :

$$\begin{cases} x_{k+1} &= x_k + v_k \cos(\theta_k) \\ y_{k+1} &= y_k + v_k \sin(\theta_k) \\ \theta_{k+1} &= \theta_k + \omega_k \end{cases} \tag{4.10}$$

L'orientation *initiale* du véhicule (aussi appelée *cap*) est fondamentale dans notre algorithme de localisation, car elle conditionne la qualité de la solution obtenue : on utilise donc une carte embarquée qui fournit l'angle de départ dans le référentiel absolu (voir Figure 4.8) et qui marque le début de l'algorithme itératif. Le cap est calculé à partir des premières données GPS (cf. éq.4.11) :

$$\theta_0 = \arctan(\frac{y_1 - y_0}{x_1 - x_0}), \tag{4.11}$$

où $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ et $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ désignent les coordonnées des positions du véhicule à l'instant initial et l'instant suivant. La figure 4.8 illustre cette équation. Dans cette figure, $(\vec{x}, \vec{y})$ désigne le repère absolu du véhicule, $(\vec{x}_v, \vec{y}_v)$ le repère relatif associé au véhicule.

Puis, à chaque instant k, l'angle θ_k est recalculé à partir de sa valeur précédente par l'équation (4.12) :

$$\theta_k = \theta_0 + \sum_{j=1}^{k} \Delta\theta_j = \theta_0 + \sum_{j=1}^{k} \omega_j. \tag{4.12}$$

4.3 Filtrage de Kalman

Les applications visées nécessitent un positionnement précis par rapport à la voie de circulation. Le filtre de Kalman a pour rôle d'extraire, à partir des données que

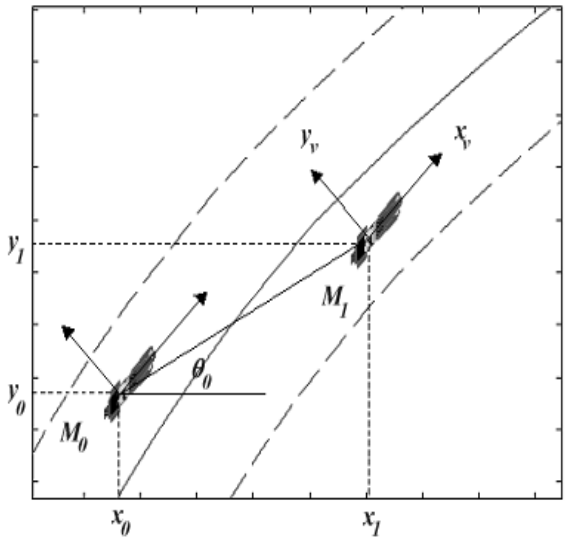


FIG. 4.8 – Variables utilisées dans le modèle d'état.

les capteurs fournissent, une estimation de l'état du véhicule (*i.e.* position, vitesse, ...) qui a généré ces données. Il permet donc de "*suivre*" ce véhicule puisque cette estimation sera utilisée à l'itération suivante pour prédire le nouvel état du véhicule. Dans cette section, nous développons les idées du filtrage de Kalman en tant qu'outil aussi bien pour l'*estimation de l'état* que pour l'*estimation des paramètres*.

Kalman [53, 52] a développé cette aproche en transformant les équations décrivant le filtre de Wiener dans le cas continu en équation différentielle. On trouvera un exposé sur le filtre de Wiener dans l'ouvrage de Lee [57] qui fut un étudiant de Wiener. D'autres approches ont été proposées pour développer le filtre de Kalman. Sage et Masters [72] ont utilisé une technique basée sur la méthode des moindres carrés. Ses principales caractéristiques sont :

- ➤ il donne l'erreur quadratique minimale,
- ➤ il est récursif, donc ne nécessite que peu de mémoire,
- ➤ il ne nécessite pas d'hypothèse de stationnarité,
- ➤ il nécessite une modélisation détaillée.

Nous nous limiterons ici à un exposé élémentaire en mettant l'accent sur les

possibilités du filtre. Dans le cas général, un système est représenté dans l'espace d'état par deux équations :

$$\left\{ \begin{array}{lcl} X_{k+1} & = & \phi_k(X_k) + U_k + G_k \\ Y_k & = & h_k(X_k) + W_k \end{array} \right. \tag{4.13}$$

Dans la première équation (l'équation d'état), X_k représente le vecteur d'état à l'instant k, ϕ_k est la fonction d'état, U_k représente la commande à l'instant k et G_k représente le bruit sur le processus. Dans la seconde équation (l'équation d'observation), Y_k est l'observation à l'instant k, h_k est la fonction d'observation et W_k est le bruit sur cette observation. Les bruits G_k et W_k sont des bruits blancs qui peuvent être gaussiens[4] ou non, indépendants de X_k.

4.3.1 Filtre de Kalman linéaire

Nous nous limitons ici au cas linéaire, c'est-à-dire en prenant $\phi_k = A_k$ et $h_k = C_k$, où A_k et C_k sont les matrices d'état et d'observation. Le cas non-linéaire est l'objet de la section 4.3.2. Les équations d'état et d'observation deviennent alors :

$$\left\{ \begin{array}{lcl} X_{k+1} & = & A_k X_k + U_k + G_k \\ Y_k & = & C_k X_k + W_k \end{array} \right. \tag{4.14}$$

Les bruits d'état G_k et de mesure W_k sont supposés blancs, centrés et de matrice de variance-covariance Γ et Ω :

$$\left\{ \begin{array}{lcll} \Gamma_k & = & E[G_k G_k^T], & E[G_k] = 0, \\ \Omega_k & = & E[W_k W_k^T], & E[W_k] = 0, \\ E[G_k W_k^T] & = & 0. & \end{array} \right. \tag{4.15}$$

Équations du filtre

Supposons qu'à l'instant k on dispose d'une mesure Y_k et de l'estimée $\widehat{X}_{k/k}$ (cette estimée est le résultat de l'application du filtre de Kalman à l'instant $k-1$). Le bruit d'état G_k étant centré et indépendant de X_k, sa meilleure prédiction sera nulle. La meilleure prédiction à l'instant $k+1$ sera donc obtenue en appliquant l'équation d'état sans le bruit :

$$\widehat{X}_{k+1/k} = A_k \widehat{X}_{k/k} + U_k. \tag{4.16}$$

De l'équation précédente, on déduit une prédiction de la mesure à l'instant $k+1$ par :

$$\widehat{Y}_{k+1/k} = C_{k+1} \widehat{X}_{k+1/k}. \tag{4.17}$$

[4]Si le bruit est gaussien, alors on peut prouver que le résultat du filtre est optimal. Dans le cas contraire, le résultat est sous-optimal (Candy [15], Tsai et Kurtz [75]).

Une estimation de $X_{k+1/k+1}$ (aux instants d'observations) est alors obtenue en utilisant un terme correctif :

$$\widehat{X}_{k+1/k+1} = \widehat{X}_{k+1/k} + K_{k+1}(Y_{k+1} - C_{k+1}\widehat{X}_{k+1/k}). \tag{4.18}$$

K_k est appelé *gain du filtre.*

Des équations précédentes, on déduit les équations des filtres

➤ prédicteur (figure 4.9)

$$\widehat{X}_{k+1/k} = A_k\widehat{X}_{k/k} + U_k + A_kK_k(Y_k - C_k\widehat{X}_{k/k}). \tag{4.19}$$

➤ estimateur

$$\widehat{X}_{k+1/k+1} = A_{k+1}\widehat{X}_{k+1/k} + U_{k+1} + K_{k+1}(Y_{k+1} - C_{k+1}\widehat{X}_{k+1/k}). \tag{4.20}$$

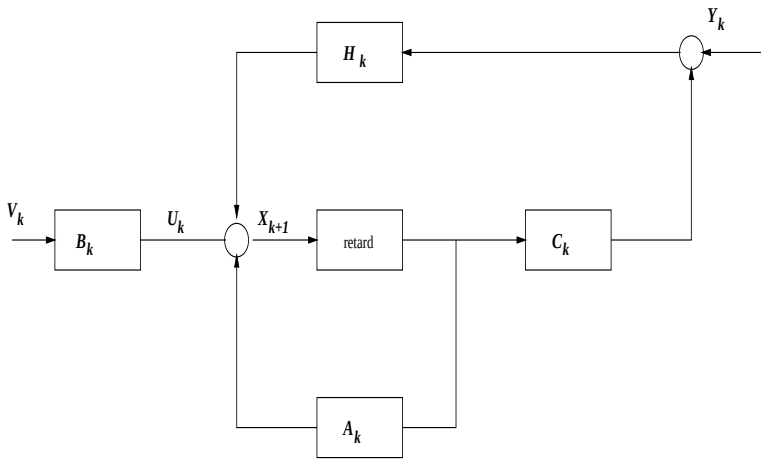


FIG. 4.9 – Structure du filtre de Kalman.

Dans l'équation 4.20, la quantité $I_{k+1} = Y_{k+1} - C_{k+1}\widehat{X}_{k+1/k}$ s'appelle l'*innovation.* Cette notion est importante. En effet, cette suite est blanche et centrée. C'est un des moyens de vérifier la convergence du filtre qui n'est pas garantie, en particulier lorsque les bruits n'ont pas les qualités de normalité qui permettent aux résultats du filtre d'être optimaux (Candy [15], Tsai et Kurtz [75]. Pour pallier ce problème, certains auteurs ont proposé une version "robuste" fondée sur une approximation polynômiale au premier ordre, qui répond au problème au prix d'un formalisme lourd et d'un coût calculatoire élevé [75].

Détermination du gain du filtre

Définissons les erreurs de prédiction et d'estimation :

$$\chi_{k+1/k} = X_{k+1} - \widehat{X}_{k+1/k} \tag{4.21}$$

$$\chi_{k+1/k+1} = X_{k+1} - \widehat{X}_{k+1/k+1} \tag{4.22}$$

Ecrivons alors que l'estimateur doit être sans biais et à variance minimale :

➤ biais nul (filtre prédicteur et filtre estimateur)

$$E[\chi_{k+1/k}] = 0 \tag{4.23}$$

$$E[\chi_{k+1/k+1}] = 0 \tag{4.24}$$

➤ variance minimale (filtre prédicteur et filtre estimateur), on doit minimiser

$$\Sigma_{k+1/k} = E[\chi_{k+1/k}\chi_{k+1/k}^T] \tag{4.25}$$

$$\Sigma_{k+1/k+1} = E[\chi_{k+1/k+1}\chi_{k+1/k+1}^T] \tag{4.26}$$

En utilisant les équations des filtres et les deux équations précédentes, on aboutit aux équations de récurrence :

$$\Sigma_{k+1/k} = A_k \Sigma_{k/k} A_k^T + \Gamma_k$$

$$\Sigma_{k+1/k+1} = [I_{k+1} - K_{k+1}C_{k+1}]\Sigma_{k+1/k}[I_{k+1} - K_{k+1}C_{k+1}]^T + K_{k+1}\Omega_{k+1}K_{k+1}^T$$

On remarquera que la première équation ne dépend pas directement de K_k. C'est donc la deuxième équation qui servira à calculer la valeur de K_{k+1} la rendant minimale. Après quelques lignes de calculs, on obtient

$$K_{k+1} = \Sigma_{k+1/k}C_{k+1}^T(\Omega_{k+1} + C_{k+1}\Sigma_{k+1/k}C_{k+1}^T)^{-1}. \tag{4.27}$$

La décroissance de sa trace au cours du temps est un critère de convergence possible. La valeur du minimum vaut alors :

$$\Sigma_{k+1/k+1} = [I_{k+1} - K_{k+1}C_{k+1}]\Sigma_{k+1/k}. \tag{4.28}$$

On obtient finalement la matrice de variance-covariance de l'erreur de prédiction :

$$\Sigma_{k+1/k} = A_k[I_{k+1} - K_kC_k]\Sigma_{k/k-1}A_k^T + \Gamma_k. \tag{4.29}$$

Mise en œuvre du filtre de Kalman

Pour utiliser l'ensemble des équations récurrentes constituant le filtre de Kalman, on doit choisir les conditions initiales, $\hat{X}_0 = E[X_0]$ et $\Sigma_{0/0} = \Gamma_0 = E[(X_0 - \hat{X}_0)(X_0 - \hat{X}_0)^T]$. Le tableau 4.3.1 résume la récurrence temporelle sur l'ensemble des équations du filtre.

Initialisation :			
	$\hat{X}_{0\|0}$	$=$	$E[X_0]$
	$\Sigma_{0/0}$	$=$	$E[(X_0 - \hat{X}_0)(X_0 - \hat{X}_0)^T]$
Récurrence au pas $k+1$ *:*			
	$\Sigma_{k+1/k}$	$=$	$A_k \Sigma_{k/k} A_k^T + \Gamma_k$
	K_{k+1}	$=$	$\Sigma_{k+1/k} C_{k+1}^T (\Omega_{k+1} + C_{k+1} \Sigma_{k+1/k} C_{k+1}^T)^{-1}$
	$\widehat{X}_{k+1/k}$	$=$	$A_k \widehat{X}_{k/k} + U_k$
	$\widehat{X}_{k+1/k+1}$	$=$	$\widehat{X}_{k+1/k} + K_{k+1}(Y_{k+1} - C_{k+1} X_{k+1/k})$
	$\Sigma_{k+1/k+1}$	$=$	$[I_{k+1} - K_{k+1} C_{k+1}] \Sigma_{k+1/k}$.

TAB. 4.1 – Algorithme du filtre de Kalman.

Les expressions des matrices de covariance d'erreur $\Sigma_{k+1/k}$ (appelée aussi erreur a priori) et $\Sigma_{k/k}$ (erreur a posteriori) d'une part et du gain K_k d'autre part peuvent être déterminées indépendamment de toute mesure et avant même que celle-ci ne soit traitée par le filtre. Les paramètres qui interviennent dans ces expressions dépendent soit des paramètres du modèle, soit des statistiques a priori du processus générateur et du bruit de mesure.

Dans l'équation de l'estimateur récursif 4.18, on retrouve le terme d'*innovation* I_k. Si la prédiction est parfaite, cette quantité est nulle. On peut montrer que si le processus I_k est un bruit blanc gaussien, alors le filtre est optimal, i.e. I_k ne contient plus d'information pouvant enrichir la mise à jour de l'estimation de l'état. Ainsi, en testant la "blancheur" de l'innovation, on peut apprécier les performances du filtre. Le filtre risque de diverger si la dynamique du bruit n'est pas gaussienne [75, 77].

Il y a lieu de noter également que :

➤ La vitesse est essentiellement liée à la bonne initialisation des paramètres X_0 et $P_{0/0}$. Lorsqu'on ne dispose pas d'information *a priori* sur la valeur de X_0,

il est nécessaire de faire un compromis afin d'obtenir une convergence garantie (covariance suffisamment forte) dans un temps acceptable (covariance pas "*trop*" forte).

➤ La qualité de la modélisation de ce processus influe également sur l'erreur d'estimation. Cependant, l'évolution du système peut changer au cours du temps (par exemple, un véhicule supposé à vitesse constante peut brusquement accélérer ou décélérer). Bar-Shalom et Birmiwal dans [3] ont proposé une méthode permettant d'évaluer à chaque instant parmi plusieurs modèles lequel était le plus approprié pour obtenir une qualité optimale.

➤ La taille du vecteur d'état influence également sur la vitesse de convergence. En effet, la convergence est obtenue globalement sur tous les paramètres. Il est clair que ce résultat sera d'autant plus rapide que le nombre de paramètres est peu élevé. Cependant, ce nombre doit être suffisamment important pour pouvoir décrire totalement l'évolution du système à l'aide de la fonction de transfert. Un compromis doit donc aussi être trouvé en fonction des contraintes sur les performances attendues. Afin d'améliorer la vitesse de convergence, quelques techniques sont apparues pour découpler le calcul des gains le long de chaque axe [2] ou pour calculer en régime permanent les différents éléments des matrices de gain et de covariance [35, 16].

➤ Enfin, il est à signaler que l'un des inconvénients de ce type de filtrage est d'inverser une matrice dans le calcul du gain de Kalman (Eq.4.27), ce qui peut provoquer des instabilités numériques. Afin de limiter ces risques, il est prudent de choisir une technique d'inversion matricielle la plus stable possible, comme par exemple une approche fondée sur l'extraction des valeurs singulières [55].

4.3.2 Filtre de Kalman étendu : EKF

Le filtrage de Kalman présenté à la section précédente supposait une linéarité des équations d'état et d'observation. En pratique, l'une ou l'autre (ou les deux) peuvent ne pas présenter cette propriété. Elles s'écrivent alors comme indiqué dans l'Eq.4.13.

Dans ce cas, on utilise une version modifiée du filtre de Kalman, nommé filtre de Kalman étendu[5](on pourra se référer au livre de Chui [20]). Cette version sous-

[5]Dans l'appellation anglo-saxonne *Extended Kalman Filter* (EKF).

optimale au sens de la minimisation de l'erreur quadratique, consiste à linéariser le modèle (*i.e.* les fonctions ϕ_k et h_k), autour de l'estimation de l'état.

Comme pour le filtre de Kalman classique, le filtre étendu est divisé en deux parties : *prédiction* et *estimation*. La prédiction consiste à estimer à partir de l'état précédent l'évolution du système, à partir de mesures extéroceptives. Elle effectue une comparaison entre une mesure obtenue par un capteur (l'observation) et une mesure simulée à partir d'une position prédite, puis elle utilise l'écart entre les deux afin d'estimer l'état du système.

Prédiction à l'instant k

L'état du système n'est pas directement mesurable et nous devons l'estimer à chaque instant. Soit $\widehat{X}_{k/k}$ la meilleure estimée de X_k à partir des observations obtenues jusqu'à l'instant k. Il est possible de prédire l'état du système à l'instant $k+1$ à l'aide de l'équation d'état, en considérant le bruit G_k comme nul :

$$\widehat{X}_{k+1/k} = \phi_k(X_{k/k}). \tag{4.30}$$

La matrice de variance-covariance de l'erreur d'estimation $\Sigma_{k+1/k}$ associée à l'estimateur $\widehat{X}_{k+1/k}$ est donnée par :

$$\begin{array}{rcl} \Sigma_{k+1/k} & = & E[(X_{k+1/k} - \widehat{X}_{k+1/k})(X_{k+1/k} - \widehat{X}_{k+1/k})^T] \\ & = & F_k \Sigma_{k/k} F_k^T + \Gamma_k \end{array} \tag{4.31}$$

où $\Sigma_{k/k}$ correspond à la matrice de variance-covariance de l'erreur associée à l'estimateur $\widehat{X}_{k/k}$, Γ_k est la matrice de variance-covariance du bruit G_k, et F_k est le jacobien de ϕ_k en X_k obtenu par la linéarisation de l'équation d'état au voisinage de $\widehat{X}_{k/k}$, i.e. :

$$F_k = \left(\frac{\partial \phi_k}{\partial X_k} \right)_{X_k = \widehat{X}_{k/k}}.$$

De même qu'il est possible de prédire l'état du système à l'instant $k+1$, il est possible de prédire la mesure qui sera observée à cet instant, à l'aide de l'équation de mesure, en considérant le bruit W_k nul :

$$\widehat{Y}_{k+1/k} = h_k(\widehat{X}_{k+1/k}). \tag{4.32}$$

La matrice de variance-covariance de l'erreur d'estimation sur les observations :

$$S_{k+1/k} = E[(Y_{k+1} - \widehat{Y}_{k+1/k})(Y_{k+1} - \widehat{Y}_{k+1/k})^T],$$

associée à l'estimateur $\widehat{Y}_{k+1}$ est donnée par :

$$S_{k+1/k} = J_{k+1}\Sigma_{k+1/k}J_{k+1}^T + \Omega_{k+1} \tag{4.33}$$

où Ω_{k+1} est la matrice de variance-covariance du bruit W_{k+1} et J_{k+1} est le jacobien de h_k en X_{k+1} obtenu lors de la linéarisation de l'équation d'observation au voisinage de $\widehat{X}_{k+1/k}$, i.e. :

$$J_{k+1} = \left(\frac{\partial h_k}{X_{k+1}}\right)_{X_{k+1}=\widehat{X}_{k+1/k}}.$$

L'équation (4.33) indique que la matrice de variance-covariance $S_{k+1/k}$ est composée de l'erreur d'observation Ω_{k+1} à laquelle s'ajoute l'erreur sur la position du véhicule $\Sigma_{k+1/k}$ projetée dans le référentiel de la mesure, par l'intermédiaire du jacobien J_{k+1}.

Estimation à l'instant $k+1$

Dans le cas de l'utilisation unique de capteurs proprioceptifs, les erreurs de localisation augmentent rapidement. Aussi, il est nécessaire de corriger l'estimation de la localisation en utilisant des capteurs extéroceptifs. Chaque observation Y_{k+1} va être comparée, par différence, avec la valeur estimée $\widehat{Y}_{k+1}$ prédite à l'étape précédente (Eq.4.32) :

$$I_{k+1} = Y_{k+1} - \widehat{Y}_{k+1/k}.$$

Cette différence, appelée ainsi qu'au chapitre précédent *innovation*, conduit à une correction de l'état prédit du système à l'étape précédente du filtre. Cette estimation permet à la fois l'ajout d'information nouvelle, et le retrait d'information périmée. Cette opération empêche le modèle interne de croître indéfiniment. L'estimation d'état à l'instant $k+1$ est donnée par :

$$\widehat{X}_{k+1/k+1} = \widehat{X}_{k+1/k} + K_{k+1}(Y_{k+1} - \widehat{Y}_{k+1/k}) \tag{4.34}$$

où Y_{k+1} correspond à la mesure observée, et $\widehat{Y}_{k+1/k}$ à la mesure prédite. Le *gain de Kalman*, K_{k+1} est donnée par :

$$K_{k+1} = \Sigma_{k+1/k}J_{k+1}^T S_{k+1/k}^{-1}. \tag{4.35}$$

La matrice de variance-covariance de l'erreur d'estimation $\Sigma_{k+1/k+1}$, associée à l'estimation $\widehat{X}_{k+1/k+1}$ est donnée par :

$$\begin{aligned} \Sigma_{k+1/k+1} &= \mathrm{E}((X_{k+1} - \widehat{X}_{k+1/k+1})(X_{k+1} - \widehat{X}_{k+1/k+1})^T) \\ &= (I_{k+1} - K_{k+1}J_{k+1})\Sigma_{k+1/k}. \end{aligned} \tag{4.36}$$

La matrice de variance-covariance de l'erreur d'estimation de l'état et la matrice de variance-covariance de l'erreur d'estimation de l'observation permettent au filtre de calculer le gain de Kalman et de l'appliquer à chaque nouvelle mesure. De ce gain dépendront les modifications (et la précision de cette modification) sur la localisation du véhicule.

Le filtre EKF a fait l'objet de nombreux développements, tant en automatique qu'en traitement du signal dans des domaines extrèmement variés. Nous citerons les domaines suivants : décodage de signaux modulés [17, 8], communications [4, 67], sysmique [22, 66], robotique et véhicule intelligent [10, 36, 39, 63]. Dans la dernière partie de ce chapitre, nous donnons une nouvelle application du filtre EKF.

4.4 Application à la localisation d'un véhicule

L'équation d'état : L'utilisation d'un filtre de Kalman pour la localisation d'un véhicule consiste à prendre comme équation d'état l'équation de progression du mouvement. Leonard et Durrant-White [59] proposent d'écrire :

$$X_{k+1} = f(X_k, U_k), \tag{4.37}$$

avec pour vecteur d'état à l'instant k, $X_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}$, $U_k = \begin{pmatrix} v_k \\ \omega_k \end{pmatrix}$ et

$$f(X_k, U_k) = \begin{pmatrix} x_k + v_k \cos(\theta_k) \\ y_k + v_k \sin(\theta_k) \\ \theta_k + \omega_k \end{pmatrix}.$$

Alors la matrice jacobienne de $f(X_k, U_k)$ est

$$F_k = \begin{pmatrix} 1 & 0 & -v_k \sin(\theta_k) \\ 0 & 1 & v_k \cos(\theta_k) \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.38}$$

U_k étant donné par l'odométrie. Les variables x_k et y_k représentent la position du véhicule le long des axes déterminés par le GPS et θ_k l'orientation du véhicule.

L'équation d'observation : Contrairement à l'équation d'état, l'équation de mesure est linéaire et donnée par :

$$Y_k = h_k X_k + W_k, \tag{4.39}$$

où $Y_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ est obtenu à partir du GPS, et $h_k = C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \forall k \in \mathbb{Z}^+$. Le bruit de mesure est évalué par caractérisation, par exemple telle qu'elle est détaillée dans la section 4.2.1.

Mise en oeuvre : La non-linéarité de l'équation d'état nous impose de choisir un filtrage de type EKF. Le processus générateur introduit un bruit de modélisation de variance :

$$Q_k = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix}, \tag{4.40}$$

avec $\sigma_x = \sigma_y$ fixés arbitrairement par l'ingénieur à 0.1 pour cette application, et $\sigma_\theta = 0$. On suppose également que les bruits dont sont entachés les mesures, sont décorrelés. On se donne :

$$\Omega_k = \begin{pmatrix} 2^2 & 0 \\ 0 & 2^2 \end{pmatrix}. \tag{4.41}$$

Conditions initiales : Les valeurs des paramètres à l'état initial sont regroupées dans la table 4.2. Elles correspondent aux informations proprioceptives fournies par les capteurs avant le début de l'expérience.

Elément	Valeur initiale X_0	Ecart max. possible
x	$x_{gps} = 0$ m	2 m
y	$y_{gps} = 0$ m	2 m
θ	θ_0	–

TAB. 4.2 – Choix des conditions initiales.

Principe de la simulation Les données gyroscopiques et odométriques ont été calculées dans un premier temps à partir des mesures DGPS *réelles*. En effet, le gyroscope n'étant pas calibré, les vitesses linéaire et angulaire ont été donc *simulées* à partir des mesures de distances parcourues entre deux instants t et $t-1$. Couramment, dans les commandes de procédés, on estime la vitesse par la dérivée de la position. Dans notre cas, si $d(t)$ et $v(t)$ désignent la distance pacourue et la vitesse linéaire du véhicule, on a

$$v(t) = \frac{d(t) - d(t-1)}{T}$$

où T est le pas d'échantillonnage temporel et

$$d(t) = \sqrt{\|x_t - x_{t-1}\|^2 + \|y_t - y_{t-1}\|^2}.$$

Cette méthode n'est valable que si la distance $d(t)$ n'est pas bruitée. En effet, si σ_d^2 désigne la variance de la position, celle de la vitesse devient $\sigma_v^2 = \frac{2}{T^2}\sigma_d^2 \approx 200\ \sigma_d^2$ si $T = 0.1s$. Si la position n'est pas bruitée, l'approximation par la dérivée reste acceptable. C'est le cas ici.
Les mesures de position (x_t, y_t) fournies par le GPS sont, quant à elles, directement utilisées dans le calcul. On peut ainsi localiser le véhicule sur son parcours. Les résultats sont présentés FIG. 4.12. Ils visualisent comment la trajectoire du véhicule (en "o") restitue l'allure de la piste d'essai (en ".") grâce à l'application du filtre EKF. La piste d'essai est la donnée-référence du DGPS (sa précision pour le positionnement du véhicule sur la piste est *centimétrique*).

Discussion des résultats Nous avons mis en évidence plusieurs points :

① La distribution de l'erreur entre les observations dans le repère absolu et l'estimateur de Kalman est *normale et biaisée* comme le montrent les histogrammes des FIG. 4.10, de biais $\mu_{\text{GPS},x} = -0.31$ pour les abscisses et de biais $\mu_{\text{GPS},y} = -1.21$ pour les ordonnées (voir TAB. 4.3). Cette moyenne est due à l'incertitude "*constructeur*" sur le GPS (de l'ordre de 2 mètres). Pour comparer les résultats du GPS avec ceux du DGPS, nous avons *sous-échantillonné* les données du DGPS, *i.e.* en ne retenant qu'un point sur 10 mesures, soit une cadence de 0.1 Hz., et appliqué le filtre EKF à ces deux jeux de données. Les résultats sont résumés TAB. 4.3. On voit que pour les données DGPS simulées, la distribution de l'erreur entre les observations dans le repère absolu et l'estimateur de Kalman est aussi gaussienne ; l'estimateur est à nouveau biaisé, mais considérablement moins : $\mu_{\text{DGPS},x} = -0.05$ pour les abscisses et $\mu_{\text{GPS},y} = 0.08$ pour les ordonnées.

Erreur	μ_{GPS}	σ_{GPS}	μ_{DGPS}	σ_{DGPS}
$x - \widehat{x}$	-0.31	3.29	-0.05	3.24
$y - \widehat{y}$	-1.21	2.97	0.08	2.83

TAB. 4.3 – Moyenne et écart type de l'erreur de positionnement du véhicule dans le repère absolu, avec des des vitesses linéaire et angulaire simulées.

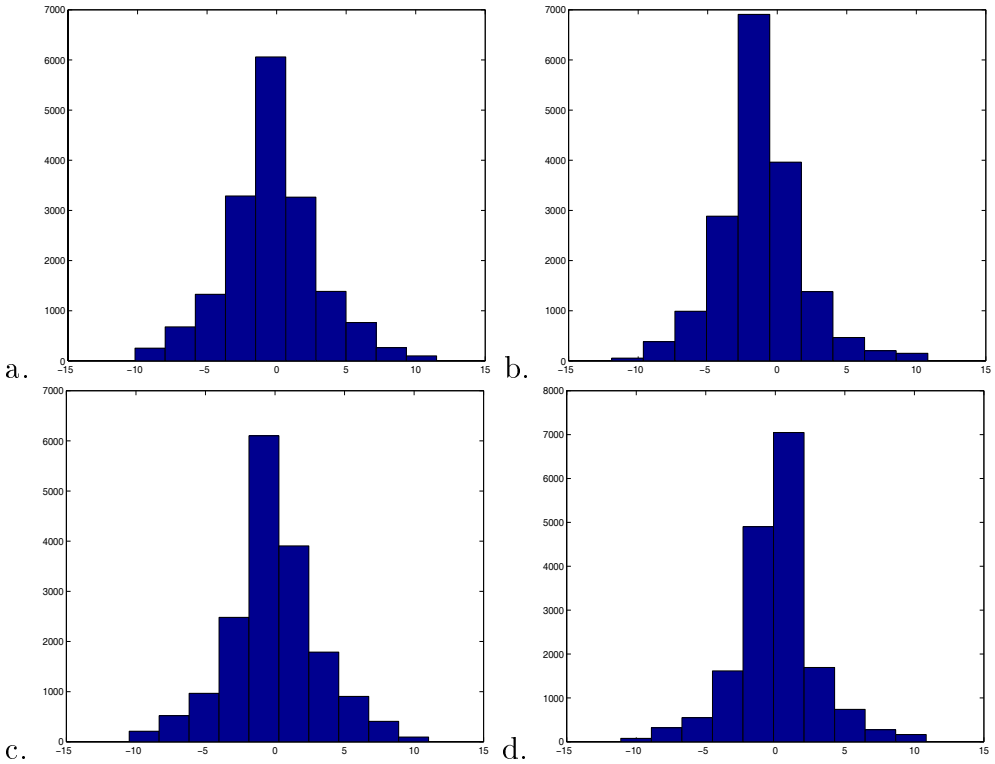


FIG. 4.10 – (a) et (b) : Erreurs d'estimation de la position GPS du véhicule $x - \widehat{x}$ et $y - \widehat{y}$. (c) et (d) : Erreurs d'estimation de la position DGPS du véhicule $x - \widehat{x}$ et $y - \widehat{y}$.

② La seconde série de résultats est obtenue à partir des données gyroscopiques et odométriques *calibrées* et *synchronisées* avec les données GPS, par la méthode du filtre de Kalman étendu. La distribution de l'erreur de positionnement GPS est à nouveau *normale* (FIG.4.11) et *biaisée*, mais la dispersion est supérieure à celle trouvée dans la première application (voir Tab.4.4). Cette différence dans la dispersion est due à l'insuffisance du calibrage de l'odomètre et du gyroscope. La moyenne des erreurs sur les ordonnées est non-nulle. Ce biais $\mu_{\text{GPS},y} = -1.437$ est dû à la précision du constructeur du GPS. Nous avons aussi appliqué la même méthode avec les données DGPS prises avec une période de $1s$, avec les données calibrées du gyroscope et de l'odomètre. La distribution des erreurs est bien légèrement biaisée ($\mu_{\text{DGPS},x} = 0.18$ et $\mu_{\text{DGPS},y} = -0.18$), mais la dispersion augmente par rapport au données simulées (Tab.4.4).

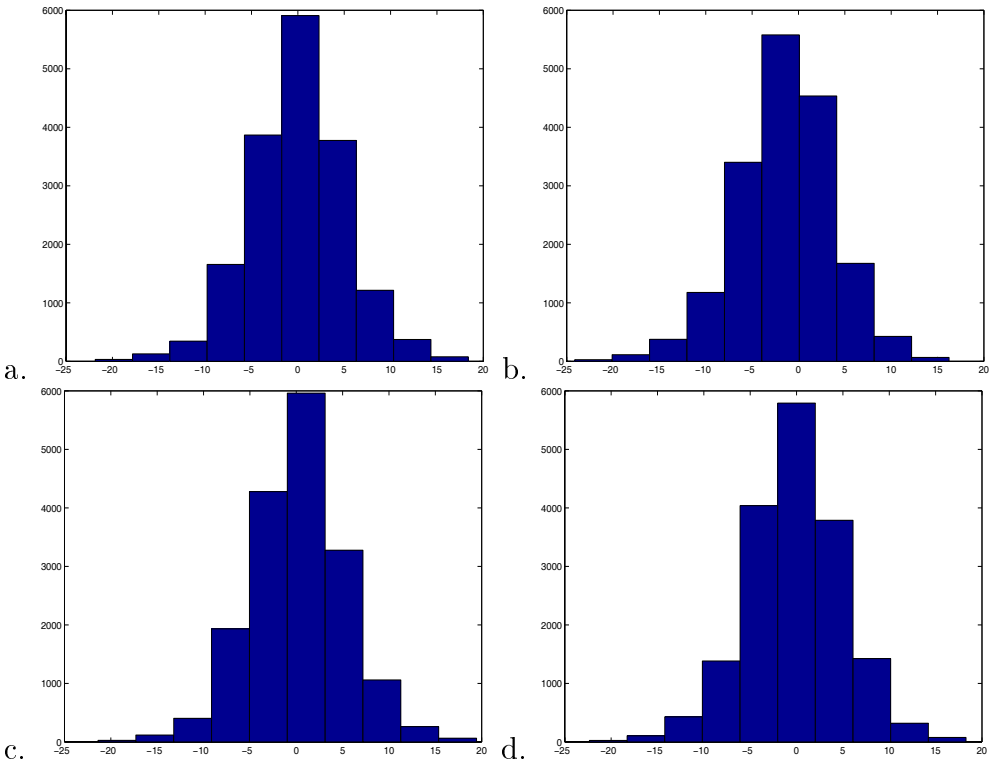


FIG. 4.11 – (a) et (b) : Erreurs d'estimation de la position GPS du véhicule $x - \widehat{x}$ et $y - \widehat{y}$ pour les vitesses linéaire et angulaire calibrées. (c) et (d) : Erreurs d'estimation de la position DGPS du véhicule $x - \widehat{x}$ et $y - \widehat{y}$ pour les données calibrées appliquées.

Erreur	μ_{GPS}	σ_{GPS}	μ_{DGPS}	σ_{DGPS}
$x - \widehat{x}$	-0.05	5.13	0.18	5.09
$y - \widehat{y}$	-1.47	5.18	-0.18	5.15

TAB. 4.4 – Moyenne et écart-type de l'erreur de positionnement du véhicule dans le *repère absolu* avec des vitesses linéaire et angulaire calibrées.

③ Plus généralement, ces résultats mettent en évidence que les erreurs commises selon x et y ne sont ni symétriques, ni centrées. On peut incriminer la géométrie de la piste, mais aussi le calibrage des capteurs inertiels.

④ Les résultats sont très sensibles vis-à-vis des conditions initiales et de bonnes connaissances *a priori* sont fortement recommandées.

La figure 4.12 représente les estimations des éléments du vecteur d'état, ainsi que les observations (de position). Les conclusions que l'on peut tirer sont les suivantes :

➤ La convergence est extrêmement rapide grâce à la qualité des observations de position : 3 à 5 itérations sont suffisantes.

➤ Le filtrage s'effectue de façon satisfaisante tant que $\theta < 12^o$ environ.

➤ Au delà de cette limite, on s'aperçoit que l'estimation de la position n'est plus centrée autour de l'observation. Cela peut être dû à deux phénomènes : avec cette courbure de la piste, la qualité de l'observation en angle est tellement meilleure que l'observation en position que le filtre ne prend en compte que la première. L'estimation de la position se fait alors par une simple intégration de la vitesse. En outre, l'écart entre observation et estimation ($x - \widehat{x}$ et $y - \widehat{y}$ dans les deux tableaux 4.3 et 4.4) traduit une dégradation des performances du GPS pour cette courbure, ce qui se traduit par l'apparition d'un biais. Ce biais peut provenir soit de la position (la caractérisation ne donne plus un résidu gaussien), soit de la vitesse.

Il faut souligner également que le filtre décrit ci-dessus a été testé sur des données réelles mesurées sur une piste sans irrégularités de terrain. D'autre part, les trajectoires ont été répétées plusieurs fois. Afin de ne pas toucher aux données, on a choisi de faire un filtre à pas d'échantillonnage fixe, défini comme l'intervalle entre deux observations séparées de $T = 0.1s$. De plus, les conclusions faites ici découlent de l'hypothèse selon laquelle la mesure de position est très fiable *quelle que soit la*

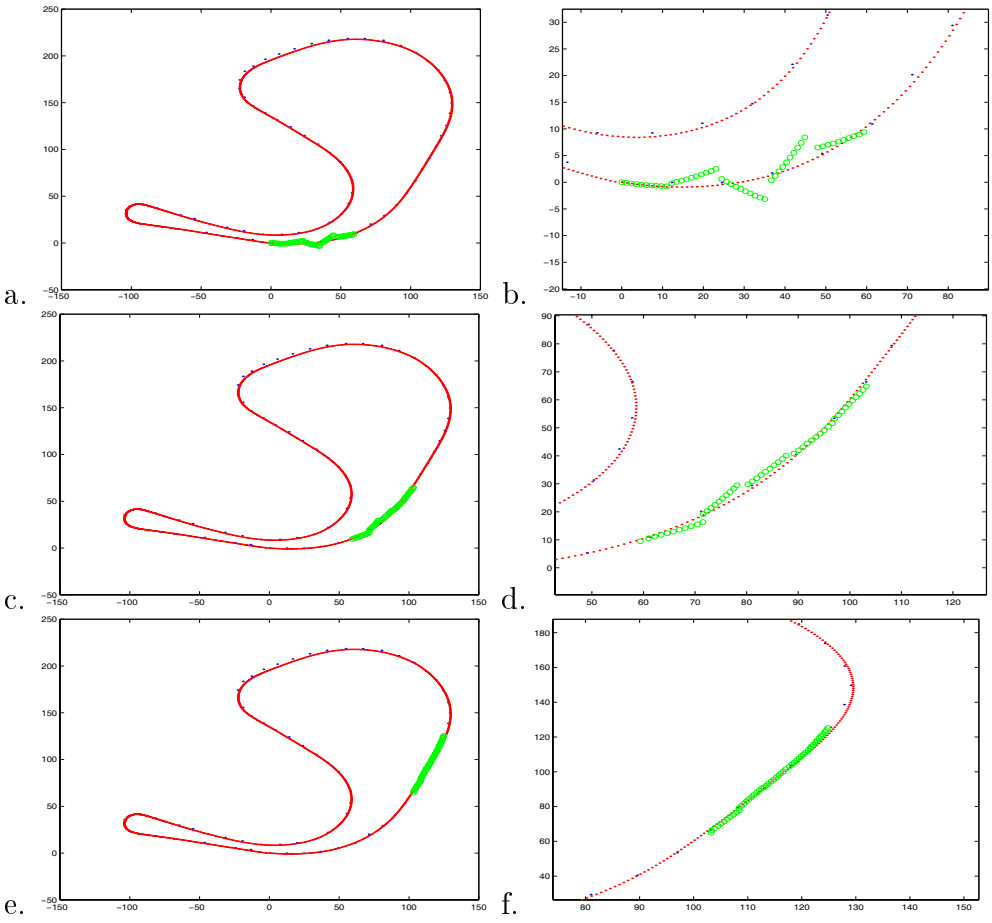


FIG. 4.12 – Convergence de l'algorithme de localisation. (a), (c), et (e) : Trajectoire du véhicule construite par le filtre EKF sur plusieurs tronçons de la piste d'essai. (b) ,(d) et (f) : Zoom sur le résultat du calcul EKF.

distance, c'est-à-dire que le bruit dont cette mesure est entachée, est constant et faible. Cependant, cette hypothèse n'est pas vérifiable, et donc rien ne permet de trancher sur l'origine du biais.

Erreur commise sur l'estimation d'état du véhicule Le système de positionnement du véhicule est le support du système de contrôle-commande du véhicule, il est donc indispensable que ce module soit fiable et procure des données en temps réel. Nos essais, réalisés sur une `Renault Mégane Scenic` mise à notre disposition par le LIVIC sur la piste d'essai du GIAT-SATORY nous ont permis de collecter des données pour trois capteurs odométrique, gyroscopique et GPS. L'utilisation de ces données brutes conduit à un "échec", du fait d'une dérive importante des deux premiers capteurs par rapport au GPS, et ce, malgré le calibrage de ces capteurs par leurs constructeurs. Nous n'avons donc pu utiliser les données collectées par ces capteurs qu'au prix d'un nouveau calibrage, détaillé dans la section précédente 4.4.

4.5 Propositions d'améliorations des résultats

Le problème posé de la localisation est d'ordre technique : en effet, un DGPS nécessite une liaison (souvent radio) avec une station de base. Cela implique un équipement lourd et coûteux. Par ailleurs, les signaux issus d'un récepteur GPS ne respectent pas les hypothèses de base du filtrage de Kalman. L'ajout de deux caméras, installées sur les rétroviseurs latéraux du véhicule, va nous permettre de calculer la distance du centre du véhicule perpendiculairement au centre de la voie. Nous pensons améliorer ainsi l'estimation de la position du véhicule.

Nous nous proposons, dans un premier temps, d'intégrer les données collectées par ces deux capteurs dans les équations d'état du véhicule. L'intégration des mesures données par les caméras permet de minimiser l'erreur de positionnement du véhicule par rapport au centre de la voie. Mais ces données ne permettent pas l'amélioration de la position longitudinale. La méthode proposée permet de tenir compte de la position du véhicule par rapport au centre de la route gérée par les caméras, mais aussi, de la position longitudinale gérée par l'odométrie. Cette méthode, dite méthode de *calcul par intervalles* , se résume en quatre étapes (voir l'exposé détaillé dans le chapitre 5) :

① Partant d'une position connue, le GPS fournira un domaine dans lequel le véhicule peut se trouver. Ce domaine sera approchée par un cercle $\mathcal{C}$ dont la taille dépendra des données constructeur.

② Les deux caméras donnent la distance d_c entre le véhicule et le centre de la voie avec une incertitude donnée par le constructeur. Le véhicule se trouve donc dans un domaine $\mathcal{D}$.

③ L'odomètre nous permet de calculer la distance parcourue entre deux instants. Avec l'incertitude donnée par le constructeur, on contraint le domaine des positions longitudinales du véhicule.

④ Le gyroscope donne l'angle de rotation muni d'une incertitude, limitant ainsi la variation angulaire du véhicule et par suite la zone positionant ce véhicule.

A l'issue d'une succession d'intersections des domaines donnés par les capteurs, on trouve un estimateur de la position. On utilisera le résultat de cette procédure pour prédire la position à l'instant suivant, jusqu'à ce qu'une nouvelle mesure soit recueillie. Cette nouvelle observation peut être la position donnée par le GPS, la distance donnée par l'odomètre, l'angle donné par le gyroscope ou la distance du véhicule par rapport au centre de la route donnée par les deux caméras.

Ces propositions font l'objet du chapitre 5 de ce mémoire de thèse. Malheureusement, il n'a pas été possible d'utiliser des données réelles non complètement disponibles. Nous avons donc utilisé des données simulées.

Chapitre 5

Localisation dynamique d'un véhicule par la méthode du calcul par intervalles

Le problème de la localisation d'un véhicule est un problème complexe comme on l'a vu dans le chapitre précédent. Sa résolution est basée sur une bonne connaissance et un bon calibrage des capteurs utilisés [13]. Nous avons vu que, malgré un calibrage des capteurs, le résultat obtenu par la méthode du filtre de Kalman n'est pas satisfaisant. Nous tentons donc de voir le problème de la localisation avec une autre approche qui intègre les erreurs de mesures dans les variables, au lieu de les présenter comme un bruit. Cette approche est appelée *calcul par intervalles*.

Reprenons par exemple l'équation d'observation du filtre de Kalman

$$Y_k = h(X_k) + W_k.$$

À un instant k, on fait une mesure. Cette mesure est entachée d'erreurs. Cette erreur est représentée par un bruit W_k dépendant des incertitudes des capteurs. Donc à l'instant k, Y_k se trouve dans un intervalle de centre la mesure observée et de demi-longueur l'incertitude des capteurs. La mesure Y_k sera donc représentée par un intervalle $[Y_k]$. De la même manière, la variable d'état X_k dans l'équation d'état du filtre de Kalman

$$X_{k+1} = \phi_k(X_k) + U_k + G_k$$

sera représentée par un intervalle $[X_k]$, et le vecteur de commande U_k par un intervalle $[U_k]$.

Dans ce chapitre, nous introduisons les principales notions de la méthode de l'analyse par intervalles, ainsi que les notations propres à cette méthode, et nous montrons que cette méthode peut résoudre des problèmes d'équations non-linéaires comprenant de nombreuses inconnues, en un temps très court et de manière garantie. Notre contribution consiste ici à appliquer le calcul par intervalles au problème de la localisation de véhicule en mouvement sur une voie routière. Nous tenons soigneusement compte de l'ensemble des données recueillis par les capteurs de toutes natures, et réalisons une fusion de données permettant d'obtenir des résultats tout à fait satisfaisants.

5.1 Introduction

L'approche probabiliste est la plus répandue dans les sciences de l'ingénieur. Dans cette approche, une variable réelle ou vectorielle de $\mathbb{R}^n$ est représentée par une densité de probabilité $p(x)$ telle que $\int_{\mathcal{D}} p(x)dx = 1$, où $\mathcal{D}$ est le *support* de la densité. Ce support contient à coup sûr la variable *incertaine* x. Dans l'approche ensembliste, la variable x est représentée par un ensemble noté $[x]$, supposé contenir x. Il est possible néanmoins que ce domaine contienne des zones où x ne se trouve pas. La représentation ensembliste est plus pauvre que la représentation probabiliste mais, en revanche, elle permet de manipuler plus facilement les variables aléatoires, car on ne fait pas d'hypothèses sur la manière dont la variable est distribuée.

Exemple : Représentation ensembliste et probabiliste.

Considérons trois variables aléatoires x, y et z réliées par la contrainte $z = x + y$. Cherchons à déduire les propriétés statistiques de z à partir de celles de x et y par les approches probabiliste puis ensembliste.

➤ Approche probabiliste. *Supposons que x et y sont munis de lois uniformes de densité $\pi(x) = 1/5$ sur $[0, 5]$ et $\pi(y) = 1/2$ sur $[1, 3]$. Pour calculer $\pi(z)$, il nous faut la loi conjointe de x et y. Si on considère x et y indépendantes, le calcul devient trivial la loi conjointe est le produit des deux lois,* i.e. *$\pi(x, y) = 1/10$ sur le pavé $[0, 5] \times [1, 3]$. On peut déduire ensuite facilement par un calcul d'intégrale la loi $\pi(z)$. Mais dans le cas général, le calcul est difficile, sauf lorsque le couple (x, y) est gaussien ou uniforme. Le calcul de $\pi(z)$ devient plus ardu si z est donné par exemple par $z = x^2.y$.*

➤ Approche ensembliste. *L'incertitude associée à x défini sur l'intervalle $[x] =$*

$[0,5]$ et à y défini sur l'intervalle $[y] = [0,5]$ est donnée par un ensemble (on reprend les mêmes ensembles que pour l'approche probabiliste). Alors z est dans l'intervalle $[z] = [0+1, 5+3] = [1,8]$

Sur cet exemple, le calcul ensembliste demande moins d'hypothèses et engendre des calculs plus simples. ■

Notons que par simplicité, il est souvent supposé que les variables sont indépendantes dans le but de connaître la densité jointe. Mais cette indépendance ne correspond pas à la réalité. La représentation de x par un ensemble $[x]$ ne signifie pas que $\pi(x)$ est uniforme mais revient juste à supposer que x est dans $[x]$, et rien de plus. En terme de probabilité, ceci revient à dire que x est à support borné et que ce support est inclus dans $[x]$. La possibilité de pouvoir raisonner en terme d'ensemble modifie radicalement les techniques de résolution de ces problèmes comme nous allons le voir dans la suite. Néanmoins, malgré tous ses avantages, le calcul ensembliste reste confiné à quelques applications dans les sciences de l'ingénieur. On peut le trouver par exemple en estimation où il est connu sous le nom d'approche à erreur bornée [46] et en commande robuste [5].

5.2 Calcul par intervalles

5.2.1 Définitions

Dans toute la suite, la notation x désigne un nombre, la notation $\boldsymbol{x}$ désigne un vecteur. *Un intervalle fermé borné* de $\mathbb{R}$ est noté $[x]$. Il peut toujours s'écrire sous la forme

$$[x] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}, \underline{x} \in \mathbb{R}, \overline{x} \in \mathbb{R}\}, \tag{5.1}$$

où $\underline{x}$ et $\overline{x}$ sont respectivement des bornes inférieures et supérieures de $[x]$. L'un des avantages de l'approche par intervalles réside dans la garantie donnée sur le contenu de l'intervalle. En effet, sur une machine, les réels ont une précision donnée selon le type de codage défini par une norme IEEE, le dernier chiffre étant donné par troncature ou par arrondi. Ainsi, le nombre π, donné avec deux chiffres après la virgule, prend la valeur $3,14$. Pour les intervalles, l'approche est différente : le nombre π appartient à l'intervalle [3,14 ; 3,15], et ce résultat est garanti. Pour que les valeurs solutions restent incluses dans l'intervalle d'étude, un arrondi extérieur peut être mis en place : la borne inférieure est alors arrondie au réel inférieur et la borne supérieure au réel supérieur de notre précision numérique.

Remarque 1

① *Par analogie, on peut définir aussi un intervalle ouvert de* $\mathbb{R}$ *et il sera noté* $]x[$. *Dans la suite de ce chapitre, on n'utilisera pour les définitions et le calcul que des intervalles fermés, bornés.*

② *Un réel* x *peut toujours être représenté comme un intervalle* dégénéré, *c'est-à-dire un intervalle dont les 2 bornes sont égales, i.e.* $\underline{x} = \overline{x} = x$.

On peut également caractériser un intervalle par

➤ sa longueur

$$w([x]) = \overline{x} - \underline{x}$$

et

➤ son centre

$$m([x]) = \frac{\overline{x} + \underline{x}}{2}$$

ou

➤ son rayon

$$r([x]) = \frac{\overline{x} - \underline{x}}{2}.$$

Ces intervalles sont des ensembles de $\mathbb{R}$. Nous rappelons dans la suite les notions que nous utilisons dans notre application.

5.2.2 Opérations sur les intervalles

Opérations ensemblistes pures

L'union et l'intersection d'ensembles sont deux notions très utilisées. Les opérateurs sont définies par

$$[x] \cap [y] = \{x \in [x] \text{ et } y \in [y]\}, \tag{5.2}$$

pour l'intersection, et

$$[x] \cup [y] = \{x \in [x] \text{ ou } y \in [y]\} \tag{5.3}$$

pour l'union.

L'union de deux intervalles ne donne pas toujours un intervalle. Par exemple, l'union des intervalles $[1, 2]$ et $[3, 4]$ a pour résultat un ensemble disjoint de deux intervalles, ce qui rend difficile la manipulation de cet opérateur. Une autre notion d'union a été introduite, qui renvoie le plus petit intervalle contenant l'ensemble des solutions de l'union :

$$[x] \sqcup [y] = [[x] \cup [y]] = [\min(\underline{x}, \underline{y}), \max(\overline{x}, \overline{y})] \tag{5.4}$$

L'intervalle solution est donc l'encadrement extérieur de l'union définie par l'équation (5.3). Cette union est appelée *union convexe*.

Opérations arithmétiques

Les opérateurs classiques peuvent aussi être définis pour les ensembles, généralisant les opérations sur les réels. Additionner, par exemple, deux intervalles, revient à chercher l'intervalle contenant toutes les valeurs des additions des deux variables appartenant à ces intervalles. Si $[-1; 3]$ est le domaine de la variable x et $[y] = [0; 2]$, l'addition de $[x]$ et $[y]$ donne l'intervalle $[-1; 5]$. Cet intervalle est déterminé par la somme des bornes inférieures qui donne -1 et celle des bornes supérieures qui donne 5. Nous définissons une opération quelconque $\circ$ entre deux sous-ensembles $\mathbb{X}$ et $\mathbb{Y}$ de $\mathbb{R}$ comme suit :

$$\mathbb{X} \circ \mathbb{Y} = \{x \circ y \mid x \in \mathbb{X} \text{ et } y \in \mathbb{Y}\}. \tag{5.5}$$

Pour $\circ \in \{+, -, \times, /\}$, le récapitulatif des opérations sur les intervalles est donné par :

$$[x] + [y] = \{x + y \mid x \in [x], y \in [y]\} = [\underline{x} + \underline{y}; \overline{x} + \overline{y}] \tag{5.6}$$

$$-[y] = \{-y \mid y \in [y]\} = [-\overline{y}; -\underline{y}] \tag{5.7}$$

$$[x] - [y] = \{x - y \mid x \in [x], y \in [y]\} = [x] + (-[y]) \tag{5.8}$$

$$\begin{aligned} [x] \times [y] &= \{x \times y \mid x \in [x], y \in [y]\} \\ &= [\min(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y},), \max(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y},)] \end{aligned} \tag{5.9}$$

$$\frac{1}{[y]} = \{\frac{1}{y} \mid y \in [y]\} = \begin{cases} [\frac{1}{\overline{y}}, \frac{1}{\underline{y}}] & \text{si } 0 \notin [y] \\ \text{non-défini} & \text{sinon} \end{cases} \tag{5.10}$$

La division telle qu'elle est définie dans l'équation 5.10 ne permet pas de prendre en compte les intervalles contenant 0. On résout ce problème en ajoutant à l'ensemble des réels $-\infty$ et$+\infty$. On complète ainsi la définition de la division pour un intervalle

contenant 0.

$$\frac{1}{[y]} = \begin{cases} \emptyset & \text{si } [y] = [0,0] \\ [\frac{1}{\overline{y}}, \frac{1}{\underline{y}}] & \text{si } 0 \notin [y] \\ [-\infty, \frac{1}{\underline{y}}] & \text{si } \underline{y} < \overline{y} = 0 \\ [-\infty, \frac{1}{\underline{y}}] \cup [\frac{1}{\overline{y}}, +\infty] & \text{si } \underline{y} < 0 < \overline{y} \\ [\frac{1}{\overline{y}}, +\infty] & \text{si } 0 = \underline{y} < \overline{y} \end{cases} \tag{5.11}$$

et

$$\frac{[x]}{[y]} = [x] \times \left(\frac{1}{[y]}\right). \tag{5.12}$$

Vecteurs d'intervalles ou *pavé*

Définition 1 *Un pavé ou* vecteur d'intervalles $[\boldsymbol{x}]$ *de* $\mathbb{R}^n$ *est le produit cartésien de* n *intervalles, noté* $[x_1] \times \ldots \times [x_n]$*, ou encore*

$$[\boldsymbol{x}] = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n] = \begin{pmatrix} [x_1] \\ \vdots \\ [x_n] \end{pmatrix}.$$

Dans cette structure, chaque composante du vecteur est un intervalle. Cette représentation présente l'inconvénient d'être néanmoins peu précise : en effet, si un ensemble $\mathbb{S}$ de forme quelconque peut être encadré par un pavé $[\boldsymbol{x}]$, l'encadrement réalisé contient tous les points de $\mathbb{S}$ mais aussi les points du pavé n'appartenant pas à $\mathbb{S}$. Ce pessimisme engendré par les pavés est appelé *wrapping effect* [64]. Par exemple si le GPS donne une position $M_k = (x_k, y_k)$ à un instant k, le véhicule se trouve dans un cercle de centre M_k et de rayon r_{GPS} (incertitude donnée par le constructeur). Dans le calcul ensembliste, le cercle sera encadré par un pavé $([x_k] \times [y_k])$ tel que $[x_k] = [x_k - r_{\text{GPS}}; x_k + r_{\text{GPS}}]$ et $[y_k] = [y_k - r_{\text{GPS}}; y_k + r_{\text{GPS}}]$ (voir figure 5.1).

Les notions ensemblistes que nous venons de rappeler dans le cas réel sont étendues composantes par composantes pour les pavés. Les définitions du centre et de la longueur sont aussi étendues composantes par composantes. Ainsi, la longueur du pavé $[\boldsymbol{x}]$ est la longueur du plus grand de ses côtés, *i.e.* $w([\boldsymbol{x}]) = \max_{i=1,\ldots,n} w([x_i])$. Le centre du pavé $[\boldsymbol{x}]$ est défini comme le vecteur $(\frac{\underline{x}_1+\overline{x}_1}{2}, \ldots, \frac{\underline{x}_n+\overline{x}_n}{2})$.

5.2.3 Fonctions d'inclusion

Soit f une fonction réelle définie sur un domaine $D \subset \mathbb{R}$. La définition de cette fonction peut être étendue aux variables intervalles $[x] \subset D$. Soit

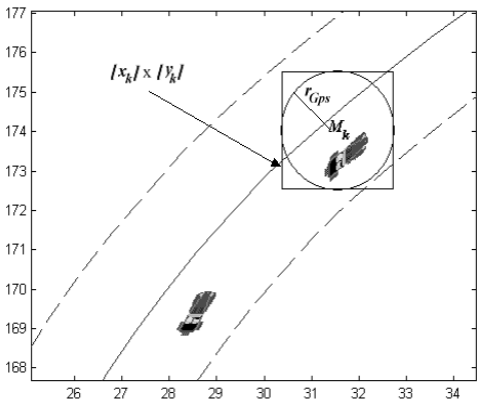


FIG. 5.1 – Représentation de l'ensemble des solutions dans un pavé.

$$f([x]) = \{f(x) | x \in [x]\}$$

l'image de $[x]$ par f. Ce n'est évidemment pas toujours un intervalle. Remarquons qu'on a toujours :

$$[x] \subseteq [y] \Longrightarrow f([x]) \subseteq f([y]).$$

Il est facile de déterminer l'image d'un intervalle par une fonction monotone, comme par exemple $\exp(x), \tan(x), \sqrt{x}, \ldots$. Ainsi, $\exp([x]) = [\exp(\underline{x}), \exp(\overline{x})]$.

Pour les fonctions non monotones , c'est plus délicat. Par exemples, les fonctions sinus et cosinus sont des fonctions périodiques, il faut donc considérer différents cas. Par exemple, pour la fonction cosinus, on pose $\cos([x]) = [\underline{\cos}, \overline{\cos}]$ avec

$$\underline{\cos} = \begin{cases} -1, & \text{si } \exists k \in \mathbb{Z} \mid 2k\pi + \pi \in [x] \\ \min(\cos(\underline{x}), \cos(\overline{x})) & \text{sinon.} \end{cases}$$

et

$$\overline{\cos} = \begin{cases} +1, & \text{si } \exists k \in \mathbb{Z} \mid 2k\pi \in [x] \\ \max(\cos(\underline{x}), \cos(\overline{x})) & \text{sinon.} \end{cases}$$

On note l'ensemble des intervalles réels par $\mathbb{IR}$. Plus généralement, si D un ensemble de réels, $\mathbb{I}D$ désigne l'ensemble des intervalles de D.

Définition 2 - Fonction d'inclusion
Soit $f : D \longrightarrow \mathbb{R}$, *une fonction* $[f] : \mathbb{I}D \longrightarrow \mathbb{IR}$ *est une* fonction d'inclusion *de* f *si*

pour tout $[x] \in \mathbb{I}D$ *elle vérifie*

$$f([x]) \subseteq [f]([x]). \tag{5.13}$$

Contrairement à l'image d'une fonction f, l'image d'une fonction d'inclusion $[f]$ est bien un intervalle. Quelle que soit la fonction f, on sait *toujours* trouver une fonction d'inclusion, mais elle n'est pas unique. Le problème est donc de trouver une fonction d'inclusion dont l'image soit un intervalle contenant l'image de f, et qui soit le plus petit possible

(figure 5.2). Dans ce cas, $[f]$ est dite *minimale*. On étend cette définition aux fonctions vectorielles, et les intervalles sont remplacés par des pavés.

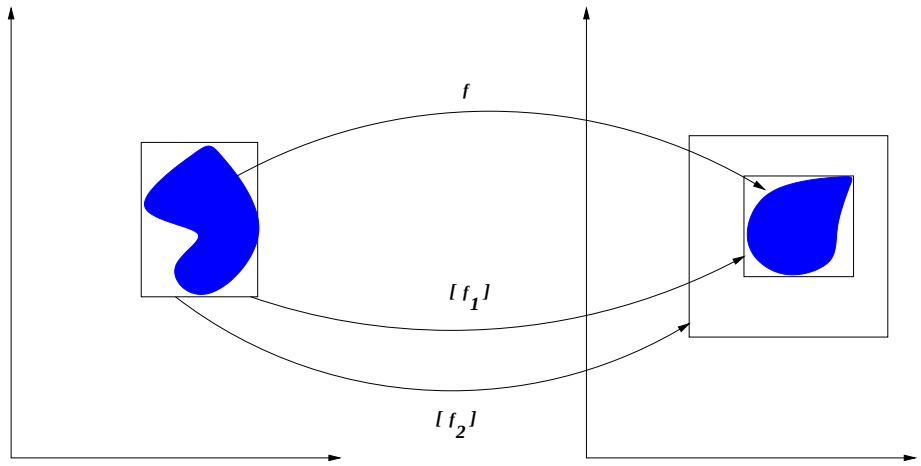


FIG. 5.2 – Image d'un ensemble par une fonction f et deux fonctions d'inclusions $[f_1]$ et $[f_2]$. La fonction d'inclusion $[f_1]$ est minimale.

Dans [44], L. Jaulin rappelle que la notion de fonction d'inclusion est la clef du calcul par intervalles : elle permet l'analyse par blocs d'un espace de recherche alors que classiquement, nous ne pouvions le parcourir que par un nuage de points de taille infinie, donc impossible à balayer en un temps fini. L'*analyse par intervalles* est généralement appelée par des algorithmes de type *branch-and-bound* qui découpent l'espace (*branch*) tout en réduisant les domaines solutions de certaines variables (*bound*). Elle propose des algorithmes permettant de trouver des fonctions d'inclusion $[f]$ *efficaces* (i.e. précises et rapides à évaluer) pour une classe énorme de fonctions (polynômiales[60], définies par des équations d'état [46] ou solutions d'une équation différentielle, etc...) et pourvu que cette fonction $[f]$ soit *convergente*. La

fonction d'inclusion $[f]$ pour f est convergente si pour toute suite $[x]_k$

$$\lim_{k\to\infty} w([x]_k) = 0 \Rightarrow \lim_{k\to\infty} w([f]([x]_k)) = 0. \tag{5.14}$$

Cette propriété est demandée pour la convergence de la plupart des algorithmes utilisant l'analyse par intervalles.

Calcul de fonctions d'inclusion : Il existe plusieurs méthodes pour obtenir une fonction d'inclusion d'une fonction donnée (le lecteur peut se rapporter à [47] pour un liste détaillée des traitements des fonctions non-linéaires). La méthode la plus simple consiste à remplacer les occurrences des variables scalaires $x, y, \ldots$ dans l'expression de f par les variables intervalles correspondantes $[x], [y], \ldots$, une fonction d'inclusion *naturelle* est alors obtenue. Cependant, cette fonction d'inclusion n'est pas nécessairement *minimale* au sens de l'inclusion. Elle ne l'est que lorsque chaque variable n'apparaît qu'une seule fois dans l'expression de f (voir Kieffer [54]). L'exemple suivant montre que le choix de l'écriture d'une fonction est important pour le calcul de la fonction d'inclusion.

Exemple : Performances de plusieurs fonctions d'inclusion.

Soit la fonction $f(x) = x^2 - x$ définie sur l'intervalle $[x] = [-1, 1]$ Etant donné quatre écritures de cette fonction

$$\begin{aligned} f_1(x) &= x(x-1) \\ f_2(x) &= xx - x \\ f_3(x) &= x^2 - x \\ f_4(x) &= (x - \frac{1}{2})^2 - \frac{1}{4} \end{aligned}$$

les évaluations intervalles sont :

$$\begin{aligned} [f_1]([x]) &= [x]([x]-1) &&= [-2, 2] \\ [f_2]([x]) &= [x][x] - [x] &&= [-2, 2] \\ [f_3]([x]) &= [x]^2 - [x] &&= [-1, 2] \\ [f_4]([x]) &= ([x] - \frac{1}{2})^2 - \frac{1}{4} &&= [-\frac{1}{4}, 2] \end{aligned}$$

Selon la fonction choisie, l'encadrement sera plus ou moins grossier (Voir Fig. 5.3).

Ces différences proviennent de ce que les expressions $[x]$ et $[x]^2$ ne sont pas équivalentes, chacune des occurences de x pouvant varier indépendemment. Ainsi, les

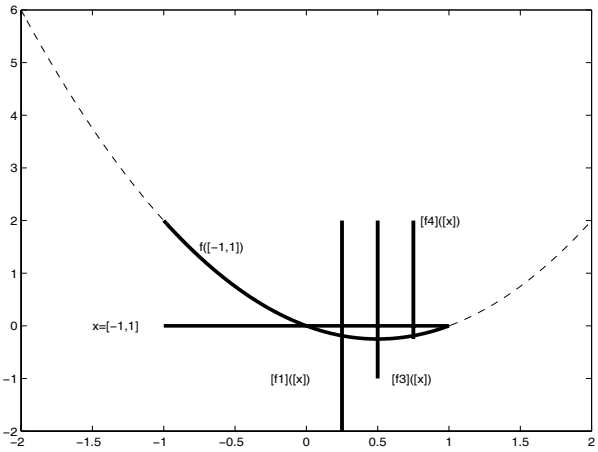


FIG. 5.3 – Comparaison de fonctions d'inclusion d'une même fonction.

incertitudes s'ajoutent dans les écritures $[f_1], [f_2]$ *et* $[f_3]$*, alors que ce n'est pas le cas si l'on considére l'écriture* $[f_4]$ *où* x *n'apparaît q'une seule fois.* ■

Dans l'exemple précédent, les fonctions d'inclusion $[f_1]$, $[f_2]$ et $[f_3]$ sont dites *pessimistes* car les intervalles images ne sont pas égaux aux plus petit intervalle contenant l'image de f. Il n'existe pas de méthode systématique permettant d'obtenir pour une fonction donnée une fonction d'inclusion minimale. Cependant, plus une variable apparaît fréquemment dans l'expression de f, plus le pessimisme sera accentué. Ce problème est connu dans la littérature sous le nom de *problème de dépendance* [47].

La complexité de la recherche d'une fonction d'inclusion augmente avec le nombre de variables. C'est la principale limitation de l'application de la méthode de l'analyse par intervalles. Cependant, il est possible de traiter de nombreux problèmes réalistes, comme par exemple la poursuite d'un robot mobile [47]. Dans la partie suivante, nous montrons que cette méthode peut être efficace pour déterminer la position d'un véhicule en mouvement sur une voie routière.

Dans la suite, pour simplifier l'écriture des équations, la notation $[y] = [f]([x])$ est équivalente à $y = f(x)$ avec $y \in [y]$ et $x \in [x]$.

5.3 Application à la localisation dynamique d'un véhicule

Nous avons vu dans le chapitre 4 les difficultés engendrées par le mauvais calibrage des capteurs pour le problème proposé par le LIVIC, de la localisation d'un véhicule en mouvement sur une voie. Ce mauvais calibrage influe directement sur l'application du filtre de Kalman étendu. Nous avons vu aussi que les performances de cette méthode sont améliorées après calibrage des capteurs. Le LIVIC a donc décidé de modifier le protocole de mesures. Le GPS a été remplacé par le DGPS qui est plus précis (précision centimétrique). L'odomètre et le gyroscope ont été remplacés par une centrale inertielle. Le LIVIC a également décidé d'ajouter deux capteurs (caméras fixées sur les rétroviseurs latéraux). Enfin, pour prendre en compte l'incertitude de toutes les mesures correspondantes, nous avons proposé d'essayer de résoudre le problème de localisation dynamique en utilisant la méthode du calcul par intervalles.

Dans la suite, on notera, à un instant k, un point M_k dans un repère $\mathcal{R}_h$ par $(x_{h,k}, y_{h,k)}$ et on confondra le repère absolu au repère lié au DGPS. Le véhicule sera donc équipé :

- d'une *carte de son environnement*, comportant les coordonnées du centre de la voie dans un repère absolu,

- d'un DGPS, tel qu'il est défini dans le chapitre 4. Le DGPS donne, à un instant k, un point $(x_{dgps,k}, y_{dgps,k}) = (x_{a,k}, y_{a,k})$ dans un repère absolu noté $\mathcal{R}_a$, avec une erreur de précision définie par le constructeur. Cette erreur se traduit par un cercle d'incertitude $\mathcal{C}$ de centre le point $(x_{a,k}, y_{a,k})$ et de rayon r (en général il s'agit d'une donnée constructeur). Le véhicule n'est pas exactement au point $(x_{a,k}, y_{a,k})$, mais il se trouve dans la limite fixée par le cercle $\mathcal{C}$. Dans la suite, et pour nos calculs, un point DGPS $(x_{a,k}, y_{a,k})$ sera un pavé $([x_{a,k}] \times [y_{a,k}])$ tel que $w([x_{a,k}]) = w([y_{a,k}]) = 2r$. Ce pavé contient le domaine $\mathcal{C}$ (figure 5.4), c'est-à-dire que le véhicule est sûrement dans ce pavé,

- d'une centrale inertielle qui donne dans un repère mobile lié au véhicule, noté $\mathcal{R}_{m,k}$ à l'instant k, les deux vitesses linéaire et angulaire avec une erreur définie par le constructeur. La vitesse linéaire $v_{m,k}$ (respectivement la vitesse angulaire $\omega_{m,k}$) est donc représentée par un intervalle $[v_{m,k}]$ (respectivement $[\omega_{m,k}]$) dans $\mathbb{R}$ de longueur égale à deux fois l'incertitude donnée par le constructeur (figure 5.5),

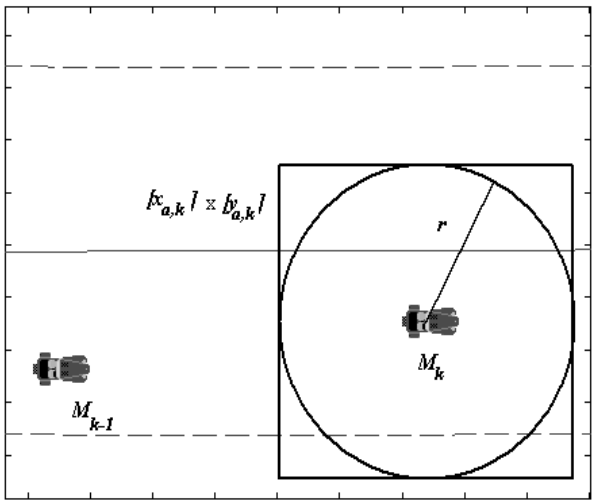


FIG. 5.4 – Position du véhicule donnée par le DGPS.

➤ de deux caméras latérales qui fournissent les distances aux bords droit $d^d_{m,k}$ et gauche $d^g_{m,k}$, ainsi que le cap. Les distances sont données dans le repère $\mathcal{R}_{m,k}$. On calcule alors la distance du véhicule au centre de la voie :

$$d_{a,k} = d_{m,k} = \frac{L}{2} - d^g_{m,k}\left(\frac{L}{d^g_{m,k} + d^d_{m,k}}\right)$$

où L la largeur de la voie. Comme pour les mesures du DGPS ou de la centrale inertielle, on tient compte de l'incertitude des mesures, en remplaçant les mesures ponctuelles par des pavés. La distance de la caméra gauche appartient au pavé $[d^g_{m,k}]$, celle de droite au pavé $[d^d_{m,k}]$. Par suite, la distance $d_{a,k}$ appartient au pavé (figure 5.6)

$$[d_{m,k}] = \frac{L}{2} - [d^g_{m,k}]\left(\frac{L}{[d^g_{m,k}] + [d^d_{m,k}]}\right)$$

On note $[\alpha_{m,k}]$ le pavé correspondant au cap.

Notons $\mathcal{R}_{c,k}$ le repère mobile lié au centre de la voie, ayant pour centre la projection du centre de $\mathcal{R}_{m,k}$ sur la tangente au centre de la voie à l'instant k. Un point

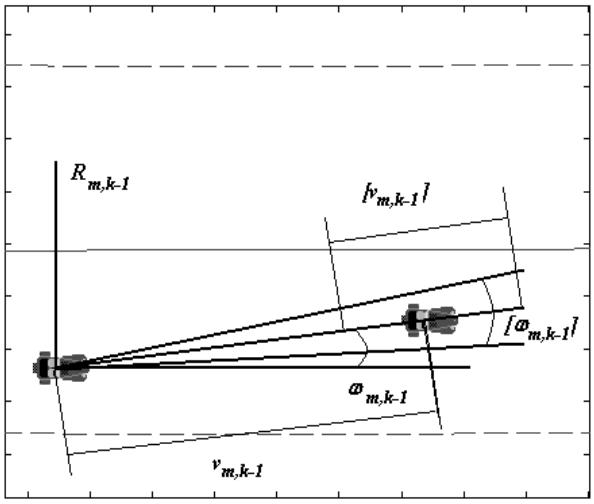


FIG. 5.5 – Incertitude sur les deux vitesses linéaire et angulaire donnée par la centrale inertielle.

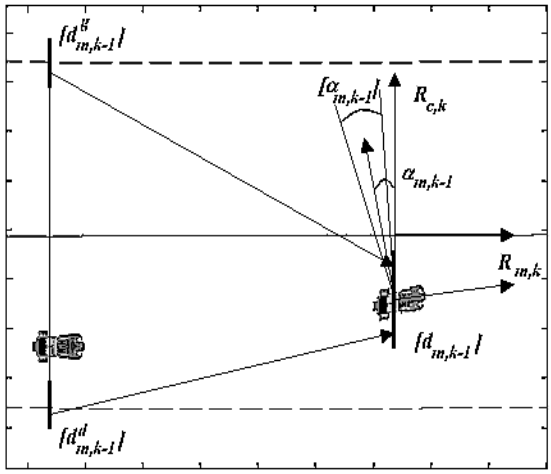


FIG. 5.6 – Incertitude donnée par les deux caméras.

M dans le repère $\mathcal{R}_{c,k}$ est représenté par $(x_{c,k}, y_{c,k})$. L'angle entre les deux repères $\mathcal{R}_{c,k}$ et $\mathcal{R}_a$ est noté $\theta_{a,k}$.

Nous récapitulons les informations données par les différents capteurs à un instant k dans la liste suivante :

$[d^g_{m,k}]$	distance du véhicule au bord gauche de la voie, donnée dans le repère $\mathcal{R}_{m,k}$,
$[d^d_{m,k}]$	distance du véhicule au bord droit de la voie dans le repère $\mathcal{R}_{m,k}$,
$[\alpha_{m,k}]$	cap du véhicule, donné dans le repère $\mathcal{R}_{m,k}$,
$[v_{m,k}]$	vitesse linéaire du véhicule, donnée dans le repère $\mathcal{R}_{m,k}$,
$[\omega_{m,k}]$	vitesse angulaire du véhicule, donnée dans le repère $\mathcal{R}_{m,k}$,
$([x_{a,k}], [y_{a,k}])$	position du véhicule donnée par le DGPS dans le repère $\mathcal{R}_a$.

Comme dans l'application du filtre de Kalman, pour résoudre le problème de la localisation du véhicule, on a besoin de deux équations formant un système : une équation dite *équation d'état* contrôlant l'évolution du véhicule, et une *équation de mesure*. Cette dernière permet de faire une correction ou un ajustement de la position lors de l'acquisition de nouvelles données. Cet ajustement se traduit par une intersection entre le résultat du calcul donné par l'équation d'état et les mesures obtenues.

On représente donc la position du véhicule à l'instant k par une variable d'état $[X_{a,k}] = ([x_{a,k}], [y_{a,k}], [d_{a,k}], [\alpha_{a,k}])$, dans le repère absolu $\mathcal{R}_a$. Les mesures, à l'instant k, sont regroupées dans une variable de mesure

$$[Y_{a,k}] = ([x_{a,k}], [y_{a,k}], [v_{m,k}], [\omega_{m,k}], [d^d_{m,k}], [d^g_{m,k}], [\alpha_{m,k}]).$$

On définit les équations d'état et de mesure par :

$$\begin{cases} [X_{a,k+1}] &= [F]([X_{a,k}]) \\ [Y_{a,k+1}] &= [H]([X_{a,k+1}]) \end{cases},$$

où $[F]$ et $[H]$ sont deux fonctions d'inclusion non-linéaires.

L'algorithme développé est le suivant : à l'instant initial $k = 0$, on dispose d'une donnée DGPS $([x_{a,0}], [y_{a,0}])$ qui permet de se positionner sur la carte et de déterminer le centre $([(Ox)_{a,0}], [(Oy)_{a,0}])$ du repère lié à la voie dans le repère absolu $\mathcal{R}_a$. Ce centre est calculé par la projection orthogonale du point DGPS $([x_{a,0}], [y_{a,0}])$ sur le segment qui relie les deux points de la carte embarquée les plus proches du point DGPS. Pour plus de précision, on pourra prendre les trois points (ou plus) de la carte embarquée les plus proches du point DGPS, et projeter sur cette courbe. Le calcul du centre $([(Ox)_{a,0}], [(Oy)_{a,0}])$ permet de calculer l'angle $[\theta_0]$ que font les deux repères entre eux. Au même instant, on a trois informations de plus provenant des deux caméras, telles que les deux distances $[d^g_{m,0}]$ et $[d^d_{m,0}]$ qui permettent de calculer la distance du véhicule par rapport au centre de la voie $[d_{m,0}]$

$$[d_{m,k}] = \frac{L}{2} - [d^g_{m,k}](\frac{L}{[d^g_{m,k}] + [d^d_{m,k}]}) \tag{5.15}$$

et le cap $[\alpha_{m,0}]$ qui donne l'orientation du véhicule. Les deux vitesses linéaire et angulaire sont nulles. La figure 5.7 montre une partie d'une voie routière (bordure gauche, centre de la voie et bordure droite), la trajectoire exacte simulée d'un véhicule, la position donnée par le DGPS (le point) ainsi que le pavé où se trouve le véhicule.

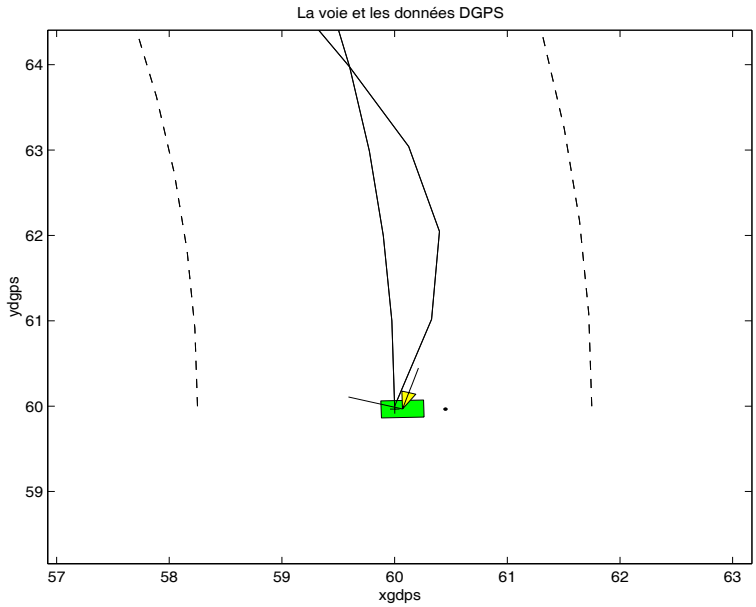


FIG. 5.7 – Initialisation : Position du véhicule sur la carte à l'instant $k = 0$

À l'instant suivant $k = 1$, on obtient les deux vitesses linéaire $[v_{m,1}]$ et angulaire $[\omega_{m,1}]$. On estime donc la position $[\hat{x}_{m,1}], [\hat{y}_{m,1}]$ du véhicule dans le repère $\mathcal{R}_{m,0}$,

$$\left\{ \begin{array}{rcl} [\hat{x}_{m,k+1}] & = & [v_{m,k}][\cos]([\omega_{m,k}]) \\ [\hat{y}_{m,k+1}] & = & [v_{m,k}][\sin]([\omega_{m,k}]) \\ [\hat{\alpha}_{m,k+1}] & = & [\hat{\alpha}_{m,k}] + [\omega_{m,k}] \end{array} \right. . \tag{5.16}$$

Par un changement de repère, on trouve la position $(\hat{x}_{c,1}, \hat{y}_{c,1})$ du véhicule et la distance $[\hat{d}_{c,1}]$ dans le repère lié à la voie $\mathcal{R}_{c,0}$:

$$\begin{array}{rcl} \left(\begin{array}{c} [\hat{x}_{c,k+1}] \\ [\hat{y}_{c,k+1}] \end{array} \right) & = & \left(\begin{array}{c} [0] \\ [\hat{d}_{m,k}] \end{array} \right) + \left(\begin{array}{cc} [\cos]([\hat{\alpha}_{m,k}]) & -[\sin]([\hat{\alpha}_{m,k}]) \\ [\sin]([\hat{\alpha}_{m,k}]) & [\cos]([\hat{\alpha}_{m,k}]) \end{array} \right) \left(\begin{array}{c} [\hat{x}_{m,k+1}] \\ [\hat{y}_{m,k+1}] \end{array} \right) \\ & = & \left(\begin{array}{c} [\cos]([\hat{\alpha}_{m,k}])[\hat{x}_{m,k+1}] - [\sin]([\hat{\alpha}_{m,k}])[\hat{y}_{m,k+1}] \\ [\hat{d}_{m,k}] + [\sin]([\hat{\alpha}_{m,k}])[\hat{x}_{m,k+1}] + [\cos]([\hat{\alpha}_{m,k}])[\hat{y}_{m,k+1}] \end{array} \right). \end{array} \tag{5.17}$$

On a alors la distance entre le véhicule et le centre de la voie :

$$\begin{array}{rcl} [\hat{d}_{c,k+1}] & = & [\hat{d}_{m,k}] + [\hat{x}_{m,k+1}][\sin]([\hat{\alpha}_{m,k}]) + [\hat{y}_{m,k+1}][\cos]([\hat{\alpha}_{m,k}]) \\ & = & [\hat{d}_{m,k}] + [\hat{v}_{m,k}][\cos]([\hat{\alpha}_{m,k}] + [\hat{w}_{m,k}]), \end{array}$$

et la position dans le repère absolu $\mathcal{R}_a$

$$\left(\begin{array}{c} [\hat{x}_{a,k+1}] \\ [\hat{y}_{a,k+1}] \end{array} \right) = \left(\begin{array}{c} [\hat{(Ox)}_{a,k}] \\ [\hat{(Oy)}_{a,k}] \end{array} \right) + \left(\begin{array}{cc} [\cos]([\hat{\theta}_{a,k}]) & -[\sin]([\hat{\theta}_{a,k}]) \\ [\sin]([\hat{\theta}_{a,k}]) & [\cos]([\hat{\theta}_{a,k}]) \end{array} \right) \left(\begin{array}{c} [\hat{x}_{c,k+1}] \\ [\hat{y}_{c,k+1}] \end{array} \right) =$$

$$\left(\begin{array}{l} [\hat{(Ox)}_{a,k}] + [\cos]([\hat{\theta}_{a,k}] + [\hat{\alpha}_{m,k}])[\hat{x}_{m,k+1}] - [\sin]([\hat{\theta}_{a,k}] + [\hat{\alpha}_{m,k}])[\hat{y}_{m,k+1}] - [\sin]([\hat{\theta}_{a,k}])[\hat{d}_k] \\ [\hat{(Oy)}_{a,k}] + [\sin]([\hat{\theta}_{a,k}] + [\hat{\alpha}_{m,k}])[\hat{x}_{m,k+1}] - [\cos]([\hat{\theta}_{a,k}] + [\hat{\alpha}_{m,k}])[\hat{y}_{m,k+1}] - [\cos]([\hat{\theta}_{a,k}])[\hat{d}_k] \end{array} \right), \tag{5.18}$$

Notons ici que la vitesse angulaire est toujours incluse dans le pavé $[-\pi, \pi[$, alors que, lors du calcul de l'angle $[\hat{\theta}_{a,k}]$ entre les deux repères $\mathcal{R}_a$ et $\mathcal{R}_{c,k}$, on peut trouver $[\hat{\theta}_{a,k}] \cap [-\pi, \pi[= \emptyset$. Par une simple transformation, et pour des raisons de calcul, on ramène le pavé $\hat{\theta}_{a,k}$ à un pavé inclus dans l'intervalle $[-\pi, \pi[$.

La position du véhicule est estimée dans le repère absolu $\mathcal{R}_a$. La figure 5.8 montre l'évolution de la position du véhicule après deux itérations ($k = 1$, $k = 2$). La position exacte est inconnue à l'instant k, mais on estime que le véhicule est dans un pavé. Les pavés trouvés par le calcul intersectent la trajectoire exacte simulée de la voiture, c'est-à-dire que le calcul par intervalles estime assez bien la position.

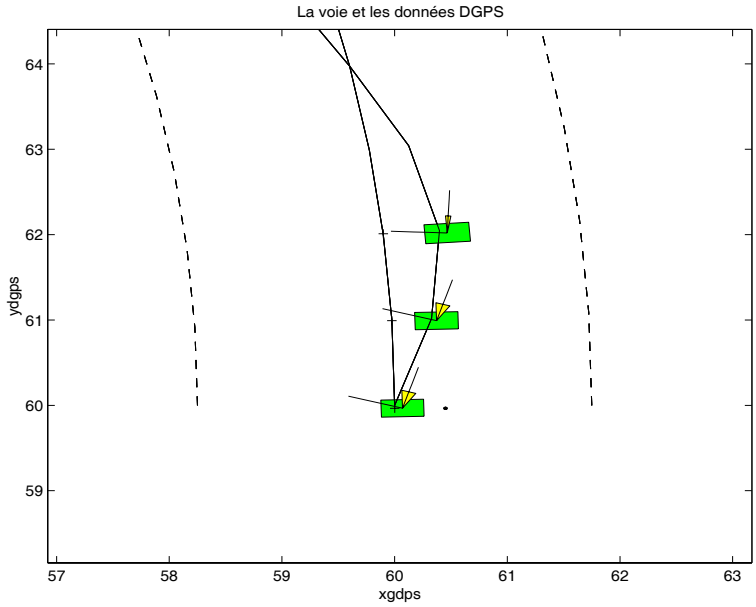


FIG. 5.8 – Position du véhicule à l'instant $k = 0, 1, 2$

On recommence la procédure jusqu'à l'arrivée de nouvelles mesures DGPS et/ou caméras. Dans notre application, les fréquences d'arrivée des données sont cinq données de vitesses linéaire et angulaire pour une donnée caméra, et dix données de vitesses linéaire et angulaire pour une donnée DGPS. Lors de l'acquisition des mesures caméras ($[d^g_{m,k}], [d^d_{m,k}], [\alpha_{m,k}]$), l'équation (5.17) corrige les calculs . Supposons, à l'instant k, que nous avons estimé la distance $[\hat{d}_{m,k}]$ et le cap $[\hat{\alpha}_{m,k}]$, et que nous ayons eu trois nouvelles mesures caméra. On calcule par l'équation 5.15 la distance $[d_{m,k}]$, et on obtient :

$$[\hat{d}_{m,k}] := [\hat{d}_{m,k}] \cap [d_{m,k}] \quad \text{et} \quad [\hat{\alpha}_{m,k}] := [\hat{\alpha}_{m,k}] \cap [\alpha_{m,k}], \tag{5.19}$$

la position du véhicule est alors donnée par l'équation (5.18).

Cet ajustement permet de recaler la position du véhicule latéralement et pas longitudinalement, c'est-à-dire de recaler sa position par rapport au centre de la voie. Les figures 5.9 et 5.10 montrent la position et le cap du véhicule avant et après le recalage par les deux caméras. Le pavé qui se trouve au centre de la figure 5.10 est l'intersection entre la position estimée et la position calculée par les deux caméras. On remarque aussi que les pavés de position du véhicule intersectent la trajectoire simulée du véhicule, c'est-à-dire que le recalage se fait assez bien avec la méthode de calcul par intervalles.

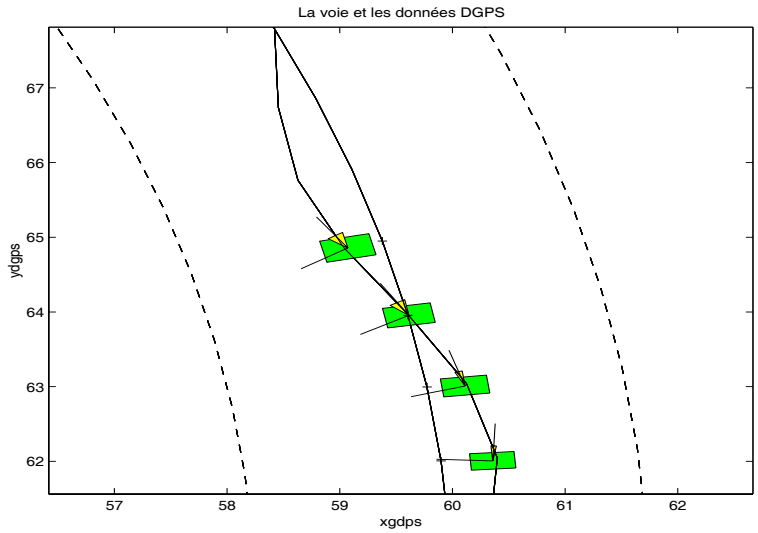


FIG. 5.9 – La position du véhicule dans le repère absolu avant le recalage par les caméras.

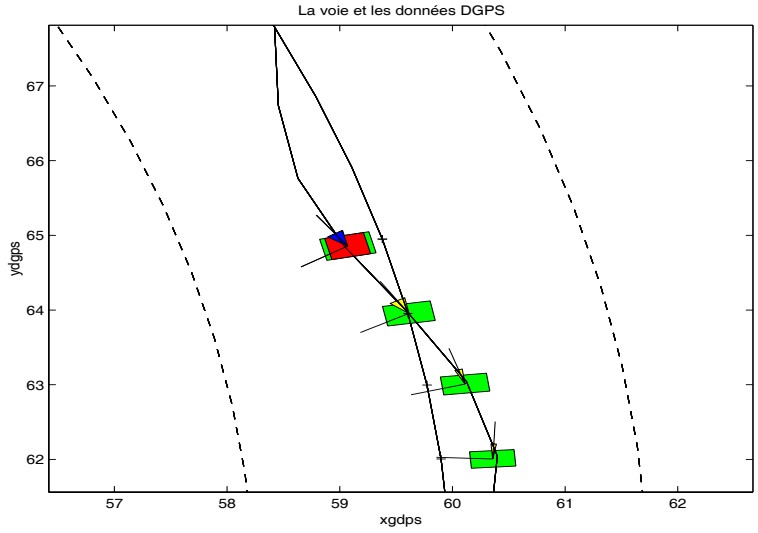


FIG. 5.10 – La position du véhicule dans le repère absolu après le recalage par les caméras.

Pour faire le recalage longitudinal, on doit attendre une nouvelle donnée Dgps $([x_{dgps}], [y_{dgps}])$. On obtient alors

$$([\hat{x}_{a,k}], [\hat{y}_{a,k}]) := ([\hat{x}_{a,k}], [\hat{y}_{a,k}]) \cap ([x_{a,k}], [y_{a,k}]). \tag{5.20}$$

La figure 5.11 montre le gain dans la position du véhicule. Le pavé du centre de la figure est l'intersection de trois pavés. Le premier est le résultat des estimations (le plus grand dans la figure), le deuxième est le résultat du calcul par les deux caméras (recalage latéral) et le dernier est le résultat des données DGPS (racalage longitudinal). L'intersection de ces trois pavés réduit la surface de position du véhicule, et pour estimer la position à l'instant suivante, on prend le pavé résultat de l'intersection, comme le montre la figure 5.12.

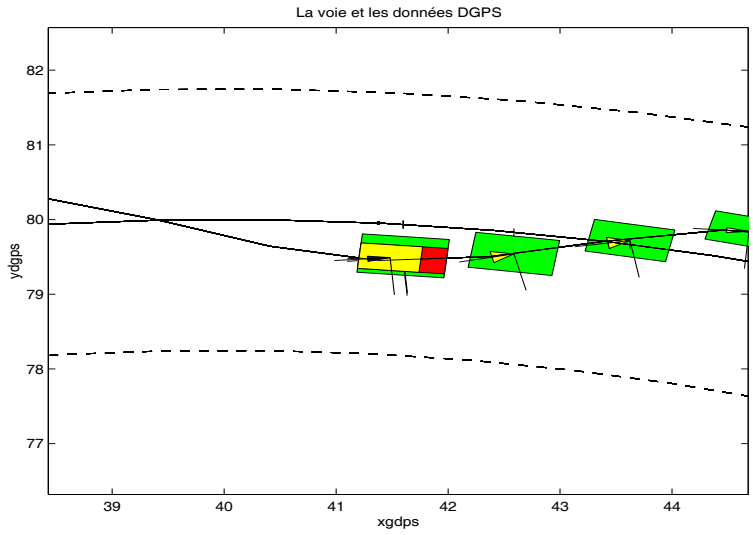


Fig. 5.11 – Recalage caméras et Dgps.

La figure 5.12 montre aussi une séquence de calcul de la position du véhicule. On peut remarquer que :

- l'erreur des capteurs se propage, c'est-à-dire que la taille des pavés augmente d'une itération au suivante,
- les recalages latéral et longitudinal se font assez bien par l'intersection,
- après chaque recalage, on recommence le calcul par le pavé résultat de l'intersection,
- tout les pavés intersectent la position exacte simulée.

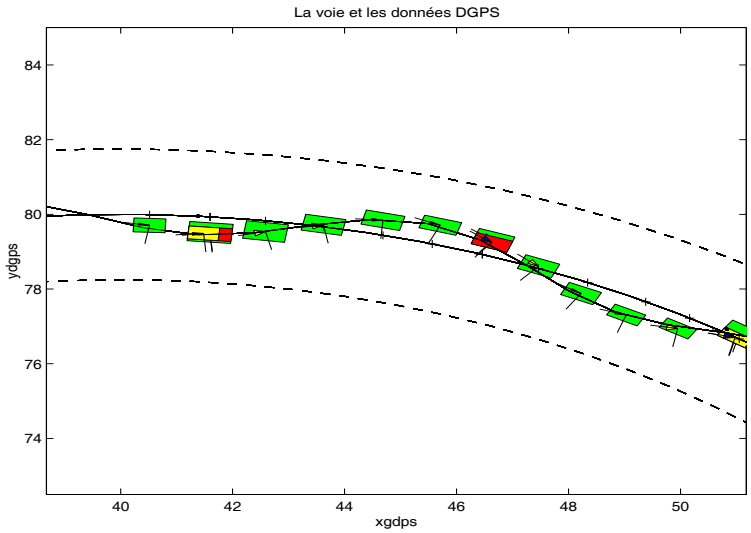


FIG. 5.12 – Position du véhicule après le recalage

5.4 Conclusion

Notre objectif dans ce chapitre était de localiser un véhicule au moyen d'un système d'équations non-linéaires par le calcul par intervalles. Le recours au calcul par intervalles était motivé par l'imprécision du calibrage des capteurs : en effet, la position cherchée du véhicule est d'autant plus entachée d'erreur que les capteurs sont imprécis. Or, il n'est plus nécessaire de tenir compte de la précision des capteurs, elle est *intégrée* dans la variable ensembliste, donc naturellement prise en compte dans l'algorithme *forward-backward* qui manipule ces variables. Le but est atteint : nous avons montré à travers la section 5.3 que le calcul par intervalles permet effectivement de localiser le véhicule sur la route à partir de données DGPS, inertielles et caméra (simulées). La position du véhicule n'est plus réduite à un point mais elle est définie par un domaine solution "garanti". Ce domaine intersecte bien la trajectoire exacte du véhicule comme les simulations page 112 le démontrent. Cependant, l'application reste à valider sur un ensemble de données réelles, qui ne sont pas encore disponibles.

Chapitre 6

Conclusion

Dans ce mémoire de thèse, on s'est intéressé à deux types de problèmes distincts. Le premier est la sélection de modèles dans la famille des perceptrons multicouches, et le second est la localisation d'un véhicule en mouvement sur une voie routière.

On a introduit dans la première partie les définitions, les propriétés ainsi que les méthodes utilisées pour résoudre le problème de la sélection de modèles non-paramétriques dans le cas de la famille des perceptrons multicouches. Les propriétés qu'on a rappelées sont basées sur des résultats asymptotiques, c'est-à-dire qu'elles sont justifiées à condition que le nombre d'observations soit assez important. Cette condition n'est pas toujours vérifiable pour des applications réelles. Pour résoudre ce problème, on a proposé l'utilisation d'une méthode de ré-échantillonnage le bootstrap.

Le bootstrap est une méthode largement utilisée dans le cas de l'estimation de paramètre, c'est-à-dire pour les modèles paramétriques comme les modèles linéaires, mais plus rarement dans le cas non-paramétrique. On a donc rappelé la méthodologie du bootstrap, et on a étendu cette méthode aux modèles non-paramétriques. Cette extension est étudiée dans le cas particulier de la sélection de modèles pour des perceptrons multicouches. On a montré, sur des exemples, que les résultats obtenus par le bootstrap sont très satisfaisants.

À la fin de la première partie, on s'est intéressé au test asymptotique de différence de contrastes appliqué aux modèles auto régressifs non-linéaires. On a montré sur un exemple simulé la « limite » de l'utilisation de ce test lorsque le nombre d'observations est trop petit. On a montré aussi que l'application du bootstrap au test permet de bien choisir le vrai modèle, et que ce test est consistant.

Dans la seconde partie, on a étudié le problème de la localisation d'un véhicule en mouvement sur une voie routière. Pour résoudre ce problème, le véhicule a été équipé de différents capteurs. Ces capteurs collectent des informations avec différentes fréquences. Le problème devient alors un problème de fusion de données pour un modèle non-linéaire. La méthode la plus utilisée dans ce cas de figure est le Filtre de Kalman étendu.

Le Filtre de Kalman étendu est une méthode robuste pour la fusion de données. Cette robustesse est obtenue sous une condition sur les erreurs des différents capteurs. Dans un premier temps, on a étudié les capteurs embarqués dans le véhicule, et on les a calibré pour pouvoir appliquer la méthode du filtre de Kalman étendu. On a, par la suite, appliqué cette méthode à la localisation du véhicule, mais les résultats obtenus n'ont pas été satisfaisants malgré le calibrage des capteurs. On a donc proposé l'ajout de deux capteurs sur le véhicule pour mieux contrôler sa position sur la route, et l'application d'une méthode qui tient compte des erreurs des capteurs sans regarder leur nature, c'est la méthode de l'analyse par intervalles.

La méthode de l'analyse par intervalles permet de mieux prendre en compte la non-linéarité du problème de la localisation, et de ne pas modéliser la nature de l'erreur. Le calibrage des capteurs est donc non-nécessaire pour l'application de cette méthode. Après une brève présentation de la méthode, on a montré que son application est très satisfaisante sur des simulations de données capteurs.

Bibliographie

[1] J.G Attali et G. Pagès (1995a) Approximation of finction by perceptrons : a new approach. *Neural Processing Letters*, **V. 22**.

[2] R.S. Baheti (1983) A sub-optimal Kalman filter design for target tracking applications. *IEEE conference on decision and control*, p. 552-556.

[3] Y. Bar-Shalom et K. Birmiwal (1978) Variable dimension filter for ma neuvering target tracking. *IEEE transactions on aerospace and electronic s ystems*, **V. AC-23(4)**, p. 618-626.

[4] Y. Bar-Shalom et T.E. Fortmann (1988) *Tracking and Data Asso ciation.* Academic Press.

[5] B.R. Barmish (1988) New tools for robustness analysis. *Proceedings of the IEEE Conference on Decision and Control*, p. 399-408.

[6] A. Barron (1993) Universal Approximation Bounds for Superposition of Sigmoidal Fonction. *IEEE Transaction on Information Theory*, **V. 39**.

[7] R. Battiti (1992) *First and Second Order Methods for Learning : Between steepest Descent and Newton's Method.* Neural Computation, **4**, p 141-166.

[8] J. Bayliss et E. Brigham (1970) Application of the Kalman filter to continuous signal restoration. *Geophysics*, **V. 35**, p. 2-23.

[9] S. Bayomog, X. Guyon, C. Hardouin, J . Yao (1996) Test de difference de contarstes et somme pendérée de khi-deux. *Canadian Jornal Statistic*, **V. 24**.

[10] J.R. Bergen, P. Anandan, K.J. Hanna and R. Hingorani (1992) Hierarchical Model-Based Motion Estimation. *Proc. ECCV'92*, p. 237-252.

[11] P.J Bickel and D.A. Freedman Same asymptotic theory for the bootstrapp. *Annals of Statistics*, **V. 9**, p. 1196-1217.

[12] D.S. Borowiak (1990) *Model discrimination for nonlinear regression models*, Marcel Dekker, New York.

[13] P. Bouron *Méthodes ensemblistes pour le diagnostic, l'estimation d'état et la fusion de données temporelles*, Université de Technologie de Compiègne, 2002

[14] N.W. Campbell, M.R.Pout, M.D.J. Priestley, E.L. Dagless, et B.T. Thomas (1994) Autonomous road vehicle navigation. *Engineering applications on Artificial Intelligence*, **V. 7(2)**,p. 177-190.

[15] J.V. Candy (1988) *Signal Processing. The modern approach.* McGraw-Hill International editions, Electrical Engineering Editions.

[16] F.R. Castella et F.G. Dunnebacke (1974) Analytic results for the x, y Kalman tracking filter. *IEEE transactions on aerospace and electronic s ystems*, **V. 10(6)**, p. 891-895.

[17] D. E. Catlin (1989) Estimation, Control, and the Discrete Kalman Filter. *Applied Mathematical Sciences*, **V. 71**, Springer-Verlag, New York.

[18] P.G. Ciarlet (1982) *Introduction à l'analyse numérique et à l'optimisation*, Masson.

[19] C. Chentouf and C. Jutten (1996) Combining sigmoids and radial basic functions in evolutive neural architectures. *ESANN'96.*

[20] C.K Chui (1990) *Kalman filtring with real-time application* Springer Verlag.

[21] M. Cottrell, B. Girard, Y. Girard, M. Mongeas, and C. Muller (1995) Neural modeling for time series : a statistical stepwise method for weignt elimination. *IEEE Transactions on Neural Networks*, **V. 6**, p. 1355-1364.

[22] N. Crump (1974) A Kalman filter approach to the deconvolution of seismic signals. *Geophysics* **V. 39**, p. 1-13.

[23] Mc Culloch, P. et Nelder, J. A. (1988) Generalized Linear Models, Chapman & Hall, seconde édition, Monographs. *Statistics and Applied Probability*, **V. 37**.

[24] G. Cybenko (1989) Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, **V. 2**, p 303-314.

[25] L. Davis (1991) *Hanbook of Genetic Algorithmes.* Van Nostrand Reinhold.

[26] P.Doukhan, M. Ghindes (1992) *Étude des processus* $X_n = f(X_{n-1}) + \epsilon_n$. Thèse, Université Paris 11.

[27] P.Doukhan, A. Tsybakov (1993) Nonparametric recursive estimation in nonlinear ARX-models. *Problems of Information Transmission*, **V. 29(4)**, p 318-327.

[28] D. Dubois and H. Prade (1988) *Possibility theory : an approch to computerized processing of incertainty*, Plenum Press, New York.

[29] M. Duflo (1990) *Méthodes récursives aléatoires.* Masson, Paris.

[30] B. Efron (1979) The convex hull of a random set of points. *Biometrika*, **V. 52**, p 331-342.

[31] B. Efron, R Tibshirani (1993) *An Introduction to the Bootstrap*. Chapman & Hall, New Yok

[32] R. A. Fisher (1952) *Contributions to Mathematical Statistics*. J. Wiley, New York.

[33] M. Frean (1990) The Upstrat Algorithm : A Method for Constructing and Trining Feedforward Neural Networks. *Neural Computation*, **V. 2**, p 147-152

[34] D.A Freedman (1981) Bootstrapping Regression model. *Annals of Statistics*, **V. 9**, p 1218-1228.

[35] B. Friedland (1973) Optimum steady-state position and velocity estimation using noisy sampled position data. *IEEE Transactions on Aerospace and Electronic Systems*, **V. 9, N° 6**, p. 906-911.

[36] T. Gandhi *et al.* (2000) Detection of Obstacles in the Flight Path of an Aircraft". *Proc. CVPR*, **V. 2**, p. 304-311.

[37] F. Gaudier (1998) *Optimisation et réseaux de neurones pour le repositionnement des barres de combustible nucléaire*. Thèse de doctorat de l'université Paris VI, ENS Cachan.

[38] D. Goldberg (1989) *Genetic Algorithms in Search, Optimisation and Machine learning*. Addison-Wesley publishing campany.

[39] N.J. Gordon, D.J. Salmond and A.F.M. Smith (1993) Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *IEEE Proceedings-F*, **V. 140, No. 2**, p. 107-113.

[40] X. Guyon (1995) *Random Fields on a Network-Modeling*. Statistic and Application, Springer-Verlag, Berlin.

[41] B. Hassibi and D. Stork (1993) Second Order Derivatives for Network Pruning : Optimal Brain Surgeon. *Neural Information Processing Systems*, **V. 5**.

[42] J. Hertz, A. Krogh et R. Palmer (1991) *Introduction to the theory of neural computation*. Addison-Wesley, Redwood City, CA.

[43] K. Hornik, M. Stinchcombr & H. White (1989) Multilayer feedforward networks are universal approximators. *Neural networks*, **V. 2**, p 395-366.

[44] L. Jaulin (2000) *Le calcul ensembliste par analyse par intervalles*. Habilitation à diriger les recherches. Université Paris-Sud, Orsay, France.

[45] L. Jaulin (1994) *Solution globale et garantie de problèmes ensemblistes, application à l'estimation non-linéaire et à la commande robuste*. Thèse de l'université Paris-Sud, Orsay, France.

[46] L. Jaulin et E. Walter (1999) Guaranteed bounded-error parameter estimation for nonlinear models with uncertain experimental factors. *Automatica*, **V 35**, p. 849-856.

[47] L. Jaulin, M. Kieffer, O. Didrit et E. Walter (2001) *Applied interval analysis.* Springer, Paris.

[48] A. J. Jones (1993) Genetic Algorithms and Their Application to the Disign of Neural Networks. *Neural Computing and Applications*, **V. 1**, p 32-45.

[49] R. Kallel, M. Cottrell, V. Vigneron (2000) Bootstrap for neural model selection. *ESANN'00*, Bruges - Belgique.

[50] R. Kallel, J. Rynkiewicz (2002) Parametric bootstrap for test of contrast difference in neural networks. *ESANN'02*, Bruges - Belgique.

[51] R. Kallel, M. Cottrell, V. Vigneron (2002) Bootstrap for neural model selection. *Neurocomputing*, **V. 48**, p 175-183.

[52] R.E Kalman (1960) A new approach to linear filtring and prediction problems. *Trans. ASME, Series D, Journal of Basic Eng.*, **V. 82**.

[53] R.E. Kalman et R.S. Bucy (1960) New results in linear filtering and prediction theory. *Trans. ASME, Series D, Journal of Basic Eng.*, **V. 38**, p. 95-101.

[54] M. Kieffer (1999) *Estimation ensembliste par analyse par intervalles, application à la localisation d'un véhicule.* Thèse de doctorat. Université Paris-Sud, Orsay.

[55] A.J. Laub (1985) Numerical linear algebra aspects on control design computations. *IEEE conference on Automatic Control*, **V. 30(2)**, p. 97-108.

[56] Y. Le Cun, J.S. Demker and S. Solla (1989) *Optimal Brain Damage.* In Proceedings of the NIPS, **V. 2**.

[57] Y.W. Lee (1960) *Statistical theory of communication.* John Wiley.

[58] R. Lengellé and T. Denoeux (1996) Training MLPs Layer by Layer using an Objective Function For Internal Representations. **V. 9**, p 83-97.

[59] J.J. Leonard et H.F. Durrant-White (1989) Navigation by correlating geometric sensor data. *Robotic Research Group, Oxford.*

[60] S.A. Malan, M. Milanese, M. Taragna et J. Garloff (1992) B^3 algorithm for robust performances analysis in the presence of mixed parametric and dynamic perturbations. *Proceedings of the 31st IEEE Conferences on Decision and Control*, p. 128-133.

[61] M. Mangeas (1984) *Propriétés statistiques des modèles paramétriques nonlinéaires de prévision de séries temporelles : Application aux réseaux de neurones à propagation directe.* Thèse, Université Paris1.

[62] M. Mangeas, J. Yao (1900) On least squares estimation for nonlinear autoregressive precesses.

[63] M. Maurer, R. Behringer, D. Dickmanns, T. Hildebrandt, F. Thomanek, J. Schielen. Vamors-p (1995) An advanced autonomous road vehicule guidance. *SPIE Mobile Robots IX, 2352*, p. 239-248.

[64] R.E. Moore (1966) *Intervalle Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

[65] M. Mézard and J. P. Nadal (1989) Learning in feedforward layered networks : the tiling algorithm. *J. Phys. A : Math. Gen.*, **V. 22**, p 2191-2203.

[66] D.T. Pham (1996) *A singular evolutive kalman filter for data assimilation in oceanography*. Rapport technique RT 163-LMC/IMAG.

[67] G. Rigoll (1986) A new algorithm for estimation of formant trajectories directly from the speech signal based on the extended Kalman filter. *ICAASP'86*, Tokio, Japan.

[68] K. Ritanen, H. Mäkelä, K. Koseine, J. Puputti, M. Sampo, M. Ojafa (1995) Dévelopment of an autonomous navigation systm for an outdoor vehicule. *IAV'95*, p. 220-225.

[69] C. Robert (1992) *Analyse Bayésienne*. Economica, Paris.

[70] D. E. Rumelhart, G.E. Hinton, R.J. Wiliams (1986) Learning internal representations by error propagation. *Parallel distributed processing*, **18**, Cambridge, MIT Press.

[71] J. Rynkiewicz (2000) *Modèles hybrides intégrant les réseaux de neurones artificiels à des modèles de chaînes de Markov cachées : Application à des prédictions de séries temporelles*. Thèse, Université Paris 1.

[72] A.P. Sage et G.W. Masters (1967) Least squares curve fitting and discrete optimum filtering. *IEEE Trans. on Education*, **V. 10, N° 1**, p. 29-36.

[73] P. Siarry et G. Dreyfus (1988) *La méthode du recuit simulé*. I.D.S.E.T. Paris.

[74] R. Tibshirani (1988) Variance stabilization and the bootstrap. *Biometrika*, **V. 75**, p 433-444.

[75] C. Tsai et L. Kurtz (1983) An adaptative robustizing approach to Kalman filtering. *Automatica* , **V. 19(3)**, p. 278-288.

[76] V. Vapnik (1996) *Learning theory*. Springer-Verlag, New-York.

[77] W.R. Wu (1996) Maneuvering target tracking with coloured noise. *IEEE Transactions on Aerospace and Electronic Systems*, **V. 32(4)**, p. 1311-1320.

[78] J. Yao (2000) On least square estimation for stable nonlinear AR process. *The Annals of institut of Matimatical Statistics*, **V. 52**, p. 316-331.

[79] A. D. Zapranis and A. P. Refenes (1999) *Principles of Neural Model Identification, Selection and Adequacy*. Springer, New York.