Apprentissage d'estimateurs sans modèle avec peu de mesures - Application à la mécanique des fluides

Kévin Kasper

NNT : 2016SACLN029

Thèse de doctorat de l'Université Paris-Saclay préparée à l'Ecole Normale Supérieure de Cachan

Ecole doctorale n°580
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Traitement du Signal

par

M. Kévin KASPER

Apprentissage d'estimateurs sans modèle avec peu de mesures - Application à la mécanique des fluides

Thèse présentée et soutenue à Cachan le 12 Octobre 2016.

Composition du Jury :

M.	Luc DUGARD	Directeur de recherche CNRS, HDR GIPSA-Lab	(Président du Jury)
M.	Michel BERGMANN	Chargé de recherche INRIA, HDR IMB	(Rapporteur)
M.	Jean-François GIOVANNELLI	Professeur des universités, HDR IMS	(Rapporteur)
M.	Régis DUVIGNEAU	Chargé de recherche INRIA, HDR Sophia Antipolis Méditerranée Center	(Examinateur)
M.	Alain RICHARD	Professeur des universités, HDR CRAN	(Examinateur)
M.	Denis SIPP	Directeur de recherche ONERA, HDR DAFE	(Examinateur)
M.	Hisham ABOU-KANDIL	Professeur des universités, HDR SATIE	(Directeur de thèse)
M.	Lionel MATHELIN	Chargé de recherche CNRS LIMSI	(Co-encadrant)

Table des matières

Table des figures

Liste des tableaux

Remerciements

Par ces quelques lignes, je tiens à remercier toutes les personnes qui ont participé de près ou de loin, de manière fortuite ou non, au bon déroulement de cette thèse. Au crépuscule de la candeur étudiante, ce sont ces personnes que j'aimerais mettre en avant dans ces remerciements.

En premier lieu, je tiens à remercier M. Pascal LARZABAL de m'avoir accepté comme thésard au SATIE et de m'avoir toujours soutenu pendant cette période de ma vie.

Au terme de ce parcours, je tiens à adresser mes plus vifs remerciements à mon directeur de thèse M. Hisham ABOU-KANDIL pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour m'avoir aidé dans les moments difficiles ainsi que pour tout le temps qu'il a consacré à faire aboutir cette thèse.

Je n'aurais jamais débuté une telle aventure sans M. Mohamed ABBAS-TURKI qui m'a donné envie de découvrir et de parcourir les méandres de la recherche scientifique.

J'adresse mes plus chaleureux remerciements à M. Lionel MATHELIN qui m'a supporté lors de mes pires moments et qui m'a aidé avec une grande efficacité à franchir les nombreux obstacles scientifiques rencontrés.

Je souhaite remercier du fond du cœur Mme Cécile DURIEU qui m'a donné envie de consacrer mon avenir à l'enseignement, qui m'a soutenu dans les moments ardus et qui m'a conseillé sur ma recherche et sur mon projet professionnel.

Je souhaite également exprimer ma gratitude à M. Eric VOURC'H qui m'a soutenu, supporté et réconforté au cours de ces trois années de thèse.

J'adresse mes plus sincères remerciements à tous les membres du Jury, à toutes celles et à tous ceux qui se sont déplacés pour assister à la soutenance et qui sont venus à la "réunion de clôture". Ils ont contribué à faire, de cette épreuve universitaire, une journée inoubliable.

Mes remerciements vont également à l'équipe administrative formée par Mmes Sophie ABRIET, Béatrice BACQUET et Aurore GRACIA qui m'ont toujours aidé avec un grand sourire même lorsque je me présentais à l'imprévu. Un grand merci à M. Dominique BACH qui a toujours résolu mes problèmes informatiques avec enthousiasme et humour.

Que MM. Yazid RIZI, Gérard CHAPLIER, Jean-Pierre BARBOT, Adrien MERCIER, Baptiste CHAREYRON, Victor MORIN, Benjamin GAUSSENS, Nidal BOUCENNA et Nomane BOULMERDJ sachent que je les remercie pour leur amité sincère et leur aide spontanée.

Enfin, merci à mes parents, mon frère et ma compagne Renuka TAYADE pour les encouragements, parfois nécessaires, que seule une famille peut offrir.

CHAPITRE 1
Contexte Scientifique

Chapitre 1

Contexte Scientifique

Contenu

Ce chapitre a pour but principal de présenter rapidement le contexte scientifique de ce travail de thèse ainsi que les objectifs à atteindre.

Chaque notion présentée rapidement dans ce premier chapitre le sera à nouveau dans les chapitres qui suivent. Ces derniers seront bien **plus détaillés** et les **références** appropriées aux travaux connexes seront données.

1.1 Introduction

Le **problème inverse** consiste à déterminer **les causes** à partir d'**observations expérimentales** [1, 2]. De tels problèmes se rencontrent dans des domaines scientifiques variés comme la géologie, l'imagerie médicale, la cosmologie, le génie électrique et la mécanique des fluides par exemple. A partir de mesures, les paramètres caractérisant les causes (les sources) sont recherchés. Malheureusement, ces derniers sont bien souvent plus nombreux que les mesures et y sont reliés de manière complexe.

Par exemple, un problème inverse linéaire peut se mettre sous la forme :

$$\boldsymbol{s} = O\boldsymbol{y} \tag{1.1}$$

où

— $\boldsymbol{s} \in \mathbb{R}^{n_S}$ représente les observations expérimentales, **connues**, $n_S \in \mathbb{N}^*$,
— $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ représente l'état du système (les causes), **inconnu**, $n_{\boldsymbol{x}} \in \mathbb{N}^*$,
— $O \in \mathbb{R}^{n_S \times n_{\boldsymbol{x}}}$ représente l'opérateur d'observation, **connu**

L'objectif est d'obtenir $\boldsymbol{y}$ à partir des observations $\boldsymbol{s}$.

Pour tenter de résoudre ce problème, il est nécessaire d'avoir un *a priori* sur le système étudié (représenté par la matrice O dans l'exemple précédent). Il faut le plus souvent un **modèle** du système concerné qui permet de faire le lien entre les grandeurs recherchées et les mesures. De ce fait, une approche purement physique est menée, dans un premier temps, afin de relier les paramètres caractérisant le système aux données mesurées.

De nombreux systèmes nécessitent, lors de leur première description, un très grand nombre de variables d'état pour permettre une modélisation satisfaisante pour l'utilisateur. De ce fait, à partir de très peu d'information mesurée, il faut récupérer de nombreux paramètres (composantes du vecteur d'état par exemple) du système étudié. En général, sans modification, reformulation préalable, un tel problème est mal posé : il y a beaucoup plus d'inconnues que de mesures. Ceci se traduit en pratique par une infinité de solutions possibles permettant d'aboutir aux données mesurées. Une démarche plus complexe doit être alors menée pour réduire le champ des vecteurs d'état possibles tout en augmentant les chances de choisir une solution convenable.

Cette thèse est consacrée à un problème inverse particulier. Tout d'abord, l'objet source à déterminer est de dimension bien plus grande que celles des mesures. Par exemple, 3015 inconnues doivent être obtenues à partir de 3 mesures seulement. Ensuite, aucun modèle physique n'est disponible pour élaborer les solutions de ce problème inverse. Seules des images du champ d'intérêt (à être reconstruit) et du champ des mesures (observé) sont disponibles. Ce choix est fait pour assurer la généralité des méthodes proposées et leur facilité d'utilisation.

Heureusement, de nombreux travaux ont été effectués sur une telle problématique, comme le montrera le chapitre 2 [3, 4, 5].

1.2 Estimation et Apprentissage

Les deux grands domaines auxquels est rattaché ce travail de thèse sont : l'**estimation** et l'**apprentissage**.

L'estimation est une branche du domaine des statistiques qui consiste à estimer la valeur des paramètres recherchés à partir de données mesurées entachées de composantes aléatoires [6, 7]. Ces dernières peuvent représenter le bruit de mesures ou bien des phénomènes plus abstraits tels que les erreurs de modélisations.

Dans ce manuscrit, l'objectif principal va être **d'obtenir un estimateur** de la quantité d'intérêt. Bien que des approches linéaires (chapitre 2) permettent d'obtenir de bons résultats, l'estimateur retenu sera une application **non-linéaire** (chapitre 3) afin de pouvoir se détacher facilement des contraintes classiques de ce type de problème (*wellposedness*). Les approches d'estimation à base de **parcimonie**, ***sparsity***, sont considérées [8, 9, 10, 11]. Les articles [12, 13] ont été le point de départ des travaux de cette thèse.

Cependant, la prise en compte de données statistiques ne sera pas faite de manière conventionnelle (approche probabiliste). L'idée principale étant de produire l'approche la plus **générale** possible, ne nécessitant aucune intégration *complexe* d'informations de nature probabiliste sur le bruit pour fonctionner. Après une étude du cas non-bruité, le bruit sera intégré grâce aux capacités **d'apprentissage** de l'algorithme recherché (section 3.4).

En effet, la nature même des données disponibles, des *images* de la grandeur d'intérêt (un champ de pression par exemple), donne à ce problème des affinités fortes avec le domaine de l'apprentissage. Dans le contexte présent (traitement de données numériques), ce domaine est plus connu sous le nom de ***Machine Learning*** puisque l'apprentissage sera fait de manière algorithmique, par des machines [14, 15]. Ces approches ont pour objectif de traiter, de manière supervisée ou non, des jeux de données afin d'extraire différentes caractéristiques connues ou non. Ce domaine est ainsi très large de par sa nature et re-

groupe de nombreux problèmes (tout comme les problèmes inverses). Plus d'information sur ce vaste domaine se trouve dans le chapitre 4.

Pour conclure, l'estimateur recherché devra être produit **uniquement** à partir des jeux de données initiaux. Il doit aussi être capable, de par sa nature, d'**inclure des informations sur le bruit de mesure**. Sa **simplicité d'utilisation** doit lui permettre d'être **applicable sur de nombreux problèmes** sans modifications majeures.

1.3 Position du problème

Recherche d'un estimateur

De nombreux domaines tels que la mécanique des fluides, l'étude des machines électriques (branche du génie électrique) et l'étude des déformations de structures massives (branche du génie civil) peuvent voir leurs grandeurs d'intérêts prédites avec grande précision. En effet, la puissance de calcul des ordinateurs de bureau actuels fait qu'il est possible d'exploiter pleinement les équations différentielles de la physique (équations de Maxwell, de Navier-Stokes, ...) à l'aide d'algorithmes exploitant leurs formes discrétisées (spatialement et temporellement). Ces simulations (types éléments finis) produisent ainsi des jeux de données très proches des données observées. Cependant, de telles simulations sont très lourdes et ne peuvent pas être utilisées en **temps réel** pour obtenir des informations sur le système (en vue d'une commande par exemple).

C'est à cette échelle que l'approche recherchée intervient. L'estimateur est produit à partir de ces jeux de données simulés et peut ensuite être utilisé en temps réel (en d'autres termes, l'estimation doit être simple à évaluer numériquement). Cet estimateur exploite les données mesurées (faible dimension) afin d'obtenir une estimée des données sources, des données d'intérêt (grande dimension).

Suivant la nature des données simulées, différents estimateurs seront proposés. Ces derniers seront de plus en plus complexes à synthétiser mais pourront être appliqués à des problèmes plus complexes.

Les estimateurs recherchés doivent être capables de répondre aux contraintes suivantes :
- ils sont produits uniquement à partir des jeux de données initiaux sous format matriciel
- ils n'exploitent aucun *a priori* physique, aucun modèle d'évolution dynamique
- ils peuvent inclure des données statistiques sur le bruit de mesure
- ils peuvent être utilisés en temps réel
- ils présentent de meilleures performances que les méthodes actuelles.

Choix des mesures

Vu la nature du problème considéré, la position des capteurs joue un rôle **primordial**. Les capteurs sont ici des sous-systèmes qui permettent d'obtenir une information sur le système principal étudié.

Dans un premier temps, les capteurs donneront une information locale sur la grandeur à estimer. Le jeu de données mesuré sera une restriction du jeu de données à estimer. Ceci correspond tout simplement au cas où les mesures sont une *restriction* du champ à

reconstruire.

Ensuite, un cas plus général où les mesures sont faites sur un objet de nature différente, mais tout de même reliées à la grandeur principale à estimer, est considéré (voir les chapitres 3 et 4). Cette différence peut prendre pour origine le fait que les mesures sont bruitées ou qu'elles sont modifiées par la caractéristique entrée-sortie des capteurs utilisés.

Étant donné qu'un nombre limité de mesures est disponible, la question "qu'est-ce qui doit être mesuré ?" doit être traitée avec soin. De ce fait, le second objectif de cette thèse est de produire un algorithme permettant d'effectuer un **placement des capteurs** adapté à l'estimateur recherché (voir le chapitre 5). Outre la production d'un placement adapté à l'estimation effectuée, cet algorithme devra être capable de prendre en compte efficacement des *contraintes de placement*. Ceci traduit le fait que les capteurs ne peuvent pas être placés librement à cause de leur extension spatiale non-nulle et/ou des contraintes de fixation des capteurs. Enfin, des informations statistiques sur le bruit de mesure devront aussi être exploitées pour produire des positions robustes.

1.4 Avant-propos

Satisfaisant à toutes ces restrictions, une méthode capable de s'adapter à n'importe quel domaine produisant des données numériques discrétisées sera élaborée. Cette méthode sera générale et ne sera pas restreinte à l'application (de nature fluide) utilisée tout au long de ce manuscrit. Elle inclura une approche de **placement des capteurs** ainsi que la génération d'**estimateurs** simples à utiliser.

Afin de synthétiser le contenu des différents chapitres et ainsi permettre une lecture plus ciblée, un résumé de chaque chapitre de ce manuscrit est disponible ci-dessous.

Le chapitre 2 a pour objectif d'**illustrer le problème** retenu. Ce dernier y sera clairement formalisé. Le contexte, les méthodes existantes, le système servant de *benchmark*, les notations et les diverses hypothèses formulées sont tous disponibles dans ce chapitre.

Le chapitre 3 expose comment la notion de **parcimonie** a été exploitée pour produire un premier estimateur. Ces pages contiennent toutes les informations qui ont permis de concevoir cet estimateur. Ce dernier est facilement reproductible à l'aide des divers Pseudo-Codes présents et des notes techniques permettant de surmonter quelques problèmes mineurs de convergence ou des cas particuliers. Cet estimateur est ensuite généralisé pour pouvoir être applicable sur plus de problèmes.

Le chapitre 4 exploite la notion de **classification** (sans oublier la parcimonie) afin d'améliorer l'estimateur général proposé dans le chapitre précédent. Ce chapitre n'est qu'une ébauche de ce qu'il est possible de faire en combinant la parcimonie et la classification. De nombreuses améliorations peuvent être faites sur chacune des étapes proposées.

Le chapitre 5 est dédié à la résolution d'un problème de **placement de capteurs**. La nécessité de produire un algorithme dédié est démontrée. Un algorithme flexible capable de s'adapter au contexte de l'étude a été proposé. Ce dernier présente de nombreux avantages qui sont détaillés dans les pages de ce chapitre.

Le chapitre 6 est un recueil **d'applications variées**. De nouveaux problèmes sont soulevés puis résolus sans difficulté par les algorithmes produits lors de ce travail de thèse.

Ce chapitre a pour objectif d'illustrer le **potentiel** des méthodes produites. Pour donner un avant goût de ce chapitre, dans un système fluide classique, une mémorisation des commandes appliquées et des mesures a été effectuée pour estimer avec grande précision le champ de pression et la traînée. De plus l'estimation du champ des vitesses a été réalisée à partir de mesures de pressions en surface de l'obstacle seulement.

Le chapitre 7 est la **conclusion générale** de ce travail. Il contient quelques propositions de **travaux de développement**, permettant d'améliorer les résultats obtenus.

Bibliographie

[1] Mario Bertero, Christine De Mol, and Edward R Pike. Linear inverse problems with discrete data . I : General formulation and singular system analysis. *Inverse Problems*, 1 :301–330, 1985.

[2] Mike Cullen, Meila A. Freitag, Stefan Kindermann, and Robert Scheichl. *Large Scale Inverse Problems : Computational Methods and Applications in the Earth Sciences.* Walter de Gruyter & Co, 2013.

[3] Laurent Cordier and Michel Bergmann. Proper Orthogonal Decomposition : An Overview. In *Lecture series 2002-04 and 2003-04 on Post-Processing of Experimental and Numerical Data.* Von Karman Institute for Fluid Dynamics, 2003.

[4] Y.C. Liang, H.P. Lee, S.P. Lim, W.Z. Lin, K.H. Lee, and C.G. Wu. Proper Orthogonal Decomposition and its appilcations—Part I : Theory. *Journal of Sound and Vibration*, 252(3) :527–544, May 2002.

[5] Muruhan Rathinam and Linda R. Petzold. A New Look at Proper Orthogonal Decomposition. *SIAM Journal on Numerical Analysis*, 41(5) :1893–1925, January 2003.

[6] Jerry M. Mendel. *Lessons in Estimation Theory for Signal Processing and Control.* Prentilce Hall, 2nd edition, 1995.

[7] Harry L. Van Trees, Kristine L. Bell, and Zhi Tian. *Detection Estimation and Modulation Theory, Part I : Detection, Estimation and Filtering Theory.* Wiley, 2nd edition, 2013.

[8] Emmanuel J. Candès. Compressive sampling. In *Proceedings of the Internation Congress of Mathematiciens*, Madrid, 2006. European Mathematical Society.

[9] Emmanuel J. Candès and Michael B. Wakin. An Introduction To Compressive Sampling. *IEEE Signal Processing Magazine*, 25(2) :21–30, March 2008.

[10] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4) :1289–1306, April 2006.

[11] Yonina C. Eldar and Gitta Kutyniok, editors. *Compressed Sensing.* Cambridge University Press, Cambridge, 2012.

[12] Ido Bright, Guang Lin, and J Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(12) :127102, 2013.

[13] Steven L. Brunton, Jonathan H. Tu, Ido Bright, and J. Nathan Kutz. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. page 16, December 2013.

[14] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. 2005.

[15] Arnaud Martin. Analyse des données. *Revue des études Arméniennes*, 28(1) :527–529, 2004.

CHAPITRE 2
Illustration du problème

Chapitre 2

Illustration du problème

Contenu

Ce chapitre a pour objectif de **formaliser le problème étudié**.

Les problématiques explicitées au chapitre précédent sont tout d'abord mises en équation. Ceci permet d'inscrire ce travail de thèse dans un cadre scientifique rigoureux. Les hypothèses faites sont illustrées et justifiées.

Afin de pouvoir comparer les différentes solutions retenues, un exemple pratique est utilisé : l'écoulement d'un fluide autour d'un obstacle. L'**objectif principal** est l'estimation de l'état du fluide à partir d'un petit nombre de mesures de pression ponctuelles effectuées en surface de l'obstacle. Une motivation supplémentaire est de réduire (dans une certaine mesure) la traînée subie par l'obstacle en agissant sur l'écoulement par le biais d'actionneurs. La physique du problème et la méthode de résolution numérique sont présentées.

Ensuite, l'approche Proper Orthogonal Decomposition (POD, [1]) sera exposée puisqu'elle est très couramment rencontrée pour effectuer une telle estimation. Les performances de cette méthode linéaire sur le système test constitueront une référence pour illustrer l'intérêt des nouvelles approches conçues. Les limitations associées à cette méthode sont clairement illustrées et des pistes de résolution seront proposées puis suivies dans le chapitre 3.

2.1 Mise en équation

Soit

$$f(\boldsymbol{x}_i, t) \in \mathbb{R}^M, M \in \mathbb{N}^* \tag{2.1}$$

le champ étudié, avec $\boldsymbol{x}_i \in \mathbb{R}^{M_x}$, $M_x \in \mathbb{N}^*$, et $t \in \mathbb{R}$ les coordonnées spatiales et temporelle respectivement. $\boldsymbol{x}_i$ repère un point de l'espace indexé par i. Les coordonnées spatiales présentent $M_x = 2$ pour une étude classique plane et $M_x = 3$ pour une étude dans l'espace à 3 dimensions. La formulation du problème peut faire apparaître plus de coordonnées que la dimension intrinsèque de l'espace et il n'est pas interdit d'avoir $M_x > 3$ selon la paramétrisation choisie. Cette formalisation permet d'englober de nombreux domaines de la physique tels que :

— la thermodynamique avec le champ scalaire de température ($f = T$, $M = 1$),
— l'électromagnétisme avec le champ vectoriel électrique ($f = \boldsymbol{E}$, $M = 3$) et magnétique ($f = \boldsymbol{B}$, $M = 3$),
— la mécanique des fluides avec le champ scalaire de pression ($f = P$, $M = 1$) et le champ vectoriel des vitesses ($f = \boldsymbol{v}$, $M = 3$) et
— la mécanique quantique avec la fonction d'onde ($f = \psi$, $M = 1$).

Cette formulation ne limite pas l'utilisateur qu'au seul domaine de la physique fondamentale. Une image numérisée peut être représentée de cette manière si f renvoie le code RVB du pixel repéré par $\boldsymbol{x}_i$. Ainsi, les problèmes de classification et d'extraction de formes, très prisées en traitement des images, peuvent aussi être décrits avec cette formulation. Sauf mention du contraire, seuls des champs scalaires ($M = 1$) seront considérés dans la suite du manuscrit. Cette restriction est faite pour alléger les notations uniquement. Un remaniement de la formulation (section 6.3.3) permettra d'utiliser les méthodes proposées sur des champs de *dimension finies quelconques*.

La **restriction majeure** imposée lors de cette étude est le fait que l'estimation recherchée doit être faite en **temps réel**. Ainsi, il est très délicat d'exploiter directement un modèle décrivant l'évolution dynamique de f en fonction du temps et des sources extérieures afin de résoudre les problèmes posés. Par exemple, il est déconseillé de résoudre directement les équations de Maxwell pour un problème d'électrodynamique ou l'équation de Fourier pour un problème de diffusion de la chaleur si une solution doit être obtenue rapidement à partir des mesures. En d'autres termes, les algorithmes recherchés ne doivent pas exploiter directement de tels modèles ou relations. Par contre, ces modèles pourraient être utilisés pour générer des données qui seront utilisées pour élaborer ces algorithmes.

Une autre raison qui motive le choix de ne pas exploiter directement un modèle est le fait que les méthodes obtenues seront **générales**. Elles pourront être appliquées à des domaines variés. Bien qu'exploiter un tel modèle permettra, *a priori*, d'obtenir de meilleures performances d'estimation, l'algorithme devra être modifié pour chaque nouvelle situation. Ceci n'est pas souhaité et la flexibilité des algorithmes conçus est une priorité. Ensuite, un modèle dynamique n'est pas toujours accessible ni facilement exploitable pour répondre au problème posé. Si le champ f représente une collection d'images distinctes, un modèle dynamique reliant ces champs n'aura pas forcément de sens. Enfin, certains phénomènes physiques sont difficilement modélisables (écoulements turbulents en mécanique des fluides par exemple). Par contre, une série de mesures sur un système réel peut permettre d'extraire des informations difficiles à obtenir en simulation.

Heureusement, des informations sur f sont disponibles : une collection de $f(\boldsymbol{x}_i, t)$ pour un nombre fini de points de l'espace et un nombre fini d'instants. Cet ensemble de valeurs, à un instant donné, est qualifié de *snapshot* : une photo capturant l'état du champ dans l'espace. La nature discrète d'une telle collection résulte plus communément des méthodes

numériques de type éléments finis ou volumes finis afin de déterminer l'état physique (f) du système et son évolution en fonction des conditions initiales et des perturbations appliquées. De nombreuses équations régissant l'évolution des systèmes physiques se trouvent sous forme d'équations différentielles et ceci incite à utiliser ces méthodes de résolution. De plus, la puissance de calcul des ordinateurs actuels et les possibilités de parallélisation ont largement contribué à la démocratisation de ces approches numériques.

L'espace est discrétisé selon un maillage approprié contenant $n_{\boldsymbol{x}} \in \mathbb{N}^*$ points. L'ensemble des points constituant le maillage est

$$\mathcal{T} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_{\boldsymbol{x}}}\}. \tag{2.2}$$

Les index repérant chaque point du maillage sont récupérés dans un seul vecteur $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$. Le fait que $n_{\boldsymbol{x}}$ puisse être *important*, confère à ce problème le qualificatif de *grande dimension*. Le champ étudié, après discrétisation spatiale, est représenté par

$$\boldsymbol{y}_t^T = (f(\boldsymbol{x}_1, t) \cdots f(\boldsymbol{x}_{n_{\boldsymbol{x}}}, t)) \in \mathbb{R}^{n_{\boldsymbol{x}}}. \tag{2.3}$$

Ce dernier vecteur correspond à l'état du champ. Par abus, vu la définition du champ f, la notation suivante est aussi utilisée :

$$\boldsymbol{y}_t = f(\boldsymbol{x}, t). \tag{2.4}$$

Seuls certains instants du temps sont retenus. Soient

$$\sqcup = \{t_1, \cdots, t_{n_{\text{snap}}}\} \tag{2.5}$$

les $n_{\text{snap}} \in \mathbb{N}^*$ instants considérés .

Les données peuvent être regroupées dans une unique matrice, la **matrice des *snapshots***. Cette matrice est notée Y et correspond à

$$Y = \left(\boldsymbol{y}_{t_1} \cdots \boldsymbol{y}_{t_{n_{\text{snap}}}}\right) \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\text{snap}}}. \tag{2.6}$$

Elle est aussi appelée la **séquence d'apprentissage** puisque qu'elle est à la base des différentes méthodes développées.

Les performances des méthodes traitant de ce problème dépendent principalement du contenu de la séquence d'apprentissage. Comment doivent être choisis ces différents *snapshots* de f et combien doivent être retenus ? Ces points doivent être traités au cas par cas suivant l'application et l'objectif fixés. Ceci est l'inconvénient principal des approches basées sur des séquences d'entraînement. La flexibilité obtenue en basant les algorithmes sur Y se fait au détriment de la possibilité de garantir des performances d'estimation absolue. Cependant, une telle limitation serait apparue en utilisant des modèles si ces derniers étaient exploités hors de leur domaine de validité.

Des informations concernant le choix de la séquence d'apprentissage sont données dans la suite de ce manuscrit (voir 2.4.2). Ce choix est en effet crucial en tant que pièce maîtresse des méthodes développées. Afin de se fixer les idées, pour un problème d'estimation, les *snapshots* pourront contenir les états du champ les plus probables d'être rencontrés. Ainsi, si le système réel se trouve proche d'un tel état, la méthode développée devra *a priori* donner de bons résultats.

Le problème principal traité dans ce manuscrit, présenté dans le chapitre précédent, va maintenant être mis en équation.

Problème pincipal : *Pb.Estimation*

Dans ce premier problème, nommé *Pb.Estimation*, l'objectif est de concevoir un estimateur **statique sans mémoire** du champ f discrétisé, soit $\boldsymbol{y}_{t_i}$, qui n'utilisera qu'un nombre très limité de mesures effectuées sur le même champ, à l'instant t_i. Les données requises pour poser le problème sont :

— la position des capteurs,
— la structure de l'estimateur utilisé,
— un critère pour quantifier les performances de l'estimation.

Les mesures sont obtenues à partir de capteurs qui sont supposés renvoyer la valeur du champ discrétisé en un point du maillage. Dans un premier temps, les capteurs ont une extension spatiale nulle. Ceci les autorise à être placés sur n'importe quel nœud du maillage. Le cas plus général où un capteur occupe plusieurs points de maillage est traité dans le chapitre 5. Soit $n_S \in \mathbb{N}^*$ le nombre de capteurs utilisés avec $n_S \ll n_{\boldsymbol{x}}$. Ces derniers sont localisés sur les points du maillage

$$\{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_{n_S}\} \subset \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_{\boldsymbol{x}}}\}. \tag{2.7}$$

A l'instant t_i, ils délivrent l'information

$$\boldsymbol{s}_{t_i}^T = (f(\boldsymbol{s}_1, t_i) \cdots f(\boldsymbol{s}_{n_S}, t_i)) \in \mathbb{R}^{n_S}. \tag{2.8}$$

Afin de relier les mesures $\boldsymbol{s}_{t_i}$ au champ discrétisé $\boldsymbol{y}_{t_i}$, une matrice $C \in \mathbb{R}^{n_S \times n_{\boldsymbol{x}}}$ est utilisée. Elle contient un seul 1 par ligne et les termes restants sont nuls. C est qualifiée de matrice creuse car elle possède très *peu* de termes non nuls. Toutes les matrices creuses satisfaisant à cette contrainte forment l'espace matriciel $\mathcal{M}_C$. La position des 1 indique la position des capteurs sur la grille du maillage (contenant $n_{\boldsymbol{x}}$ éléments). La relation suivante est obtenue :

$$\boldsymbol{s}_{t_i} = C\, \boldsymbol{y}_{t_i}. \tag{2.9}$$

La matrice $S \in \mathbb{R}^{n_S \times n_{\text{snap}}}$ est obtenue en concaténant les mesures associées aux différents *snapshots*, ainsi

$$S = \left(\boldsymbol{s}_{t_1} \ldots \boldsymbol{s}_{t_{n_{\text{snap}}}}\right). \tag{2.10}$$

Tout d'abord, l'estimateur utilisera uniquement les mesures à un instant donné, t_i, pour obtenir $\widehat{\boldsymbol{y}}_{t_i}$, une estimée de $\boldsymbol{y}_{t_i}$. Dans la suite (6.1), une collection de mesures passées sera utilisée (mise en mémoire), réalisant ainsi ce qui est plus communément appelé du filtrage. De la prédiction, un cas d'estimation où un état futur est estimé à partir de mesures passées, sera aussi présentée (6.3.1). En notant $\Psi(.)$ l'application de $\mathbb{R}^{n_S} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ représentant l'estimateur, la relation suivante permet de définir $\widehat{\boldsymbol{y}}_{t_i}, \forall t_i \in \sqcup$:

$$\widehat{\boldsymbol{y}}_{t_i} = \Psi(\boldsymbol{s}_{t_i}). \tag{2.11}$$

Dans le contexte de cette étude, l'estimateur doit être conçu à partir de la séquence d'apprentissage Y et de la position des capteurs C . L'application $\Psi(.)$ n'est pas nécessairement linéaire. En regroupant les vecteurs estimés associés aux différents *snapshots*, la matrice $\widehat{Y} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\text{snap}}}$ est formée selon

$$\widehat{Y} = \left(\widehat{\boldsymbol{y}}_{t_1} \ldots \widehat{\boldsymbol{y}}_{t_{n_{\text{snap}}}}\right) = \left(\Psi(\boldsymbol{s}_{t_i}) \ldots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}\right)\right). \tag{2.12}$$

Deux normes sont utilisées pour quantifier les performances de l'estimation, la norme ℓ_2 et la norme de Frobenius. La norme ℓ_2 d'un vecteur, définie sur un espace euclidien muni d'un repère orthonormal, consiste à prendre la racine carrée de la somme des composantes

du vecteur en question, elles-mêmes mises au carré. Soient y_{j,t_i} la $j^{\text{ème}}$ composante du vecteur $\boldsymbol{y}_{t_i}$, avec $1 \leq j \leq n_{\boldsymbol{x}}$. La norme ℓ_2 de $\boldsymbol{y}_{t_i}$ est

$$||\boldsymbol{y}_{t_i}||_2 = \sqrt{\sum_{j=1}^{n_{\boldsymbol{x}}} y_{j,t_i}^2}. \tag{2.13}$$

La norme de Frobenius est une norme matricielle qui peut être vue comme une extension de la norme ℓ_2. Soient $y_{j,i}$ les composantes de la matrice Y, avec $1 \leq j \leq n_{\boldsymbol{x}}$ et $1 \leq i \leq n_{\text{snap}}$. j correspond ici à l'indice de la ligne et i à celle de la colonne de Y. La norme de Frobenius de Y est

$$||Y||_F = \sqrt{\sum_{i=1}^{n_{\text{snap}}} \sum_{j=1}^{n_{\boldsymbol{x}}} y_{j,i}^2} = \sqrt{\sum_{i=1}^{n_{\text{snap}}} ||\boldsymbol{y}_{t_i}||_2^2}. \tag{2.14}$$

Pour un instant donné t_i, la qualité de l'estimation est obtenue via la norme 2 de l'erreur de reconstruction ϵ_{t_i}, soit

$$\epsilon_{t_i} = ||\boldsymbol{y}_{t_i} - \widehat{\boldsymbol{y}}_{t_i}||_2 = ||\boldsymbol{y}_{t_i} - \Psi\left(\boldsymbol{s}_{t_i}\right)||_2. \tag{2.15}$$

L'erreur relative associée est définie par :

$$\epsilon_{rt_i} = ||\boldsymbol{y}_{t_i} - \widehat{\boldsymbol{y}}_{t_i}||_2 / ||\boldsymbol{y}_{t_i}||_2. \tag{2.16}$$

Afin de concevoir un estimateur performant sur l'ensemble de la séquence d'entraînement, l'erreur moyenne sur chaque *snapshot* est calculée via ϵ

$$\epsilon = ||Y - \widehat{Y}||_F. \tag{2.17}$$

Avec cette norme matricielle, chaque instant possède la même importance ainsi que chaque point possible du maillage. Une pondération des différents termes est possible afin de traduire l'importance de certaines zones du maillage et/ou de certains instants.

L'erreur relative effectuée sur la totalité de la séquence est aussi définie via

$$\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F. \tag{2.18}$$

Le problème *Pb.Estimation* est énoncé ci-dessous :

<u>Données</u> : Y, C
<u>Objectif</u> : Trouver une application $\Psi\left(.\right)$ définie sur $\mathbb{R}^{n_S} \rightarrow \mathbb{R}^{n_{\boldsymbol{x}}}$ qui minimise
$$\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F$$
où $\widehat{Y} = \left(\Psi\left(\boldsymbol{s}_{t_1}\right) \ldots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}\right)\right)$ et $S = CY$.

Cette formulation est directe et simple. La version vectorielle associée à cette dernière est utilisée dans de nombreux travaux de recherche (par exemple [2, 3, 4, 5, 6]). Le reproche principal qui peut être fait à la formulation ici retenue est la non-prise en compte de données statistiques sur le bruit de mesure. De telles informations sur **le bruit** seront intégrées dans le chapitre suivant (voir la partie 3.4).

2.2 Système Test

Cette section a pour objectif de décrire un système pratique qui servira de référence tout au long du manuscrit. Un tel système permettra principalement d'illustrer la flexibilité de la formulation précédente et la méthode couramment utilisée pour résoudre ce problème. L'expérience retenue est l'écoulement d'un fluide autour d'un obstacle solide, un cylindre. Avant d'expliciter chaque paramètre, un rappel des concepts de base de la mécanique des fluides est donné.

2.2.1 Mécanique des Fluides

La mécanique des fluides traite de l'étude du *comportement des fluides* et des *forces internes* associées. On appelle plus communément fluide les liquides et les gaz. Un écoulement de fluide peut être décrit par la connaissance du mouvement de chacune des entités élémentaires le constituant, en général des atomes et/ou des molécules. Cependant, une telle approche fait intervenir un bien trop grand nombre de particules. Par exemple, un volume de 1m^3 d'eau liquide contient environ 3.10^{30} molécules d'eau. Une approche exploitant directement les lois de la mécanique du point sur un système à plusieurs corps est alors déconseillée à cause du nombre d'inconnues (position et vitesse de chaque molécule, d'un point de vue classique). Une description plus *macroscopique* du fluide est nécessaire pour contourner cet obstacle dû à la dimension. Les paragraphes qui suivent ont pour but de présenter succinctement les notions essentielles de la mécanique des fluides afin de mieux cerner les spécificités du système test.

Particule Fluide

Le concept de particule fluide est indispensable dans les études de ce domaine. Au lieu de considérer chacune des entités élémentaires, un **groupement** de ces dernières est réalisé afin de former une particule fluide. L'intérêt d'une telle description est de pouvoir y associer des grandeurs macroscopiques, facilitant grandement la description du fluide. Chaque particule fluide possède par construction une masse élémentaire constante lors de l'écoulement. La vitesse d'une particule fluide est la vitesse d'ensemble (vitesse barycentrique) des molécules qu'elle contient. La pression et la température d'un tel objet peuvent aussi être définies.

La validité de cette description dépend de la taille de la particule fluide. Cette taille doit être petite au niveau macroscopique, où les grandeurs d'intérêts sont continues, afin de décrire précisément l'écoulement. Elle doit aussi être grande au niveau microscopique afin de pouvoir négliger les fluctuations associées principalement à l'agitation thermique (par exemple, la température d'une particule élémentaire n'a pas de sens). L'échelle d'étude est qualifiée de **mésoscopique**.

Pour se faire une idée plus concrète, un écoulement d'eau liquide dans un conduit de 10cm de diamètre est considéré. La longueur caractéristique de cet écoulement est $L = 0.1\text{m}$. La distance moyenne entre deux molécules d'eau est de l'ordre de grandeur du diamètre de cette molécule, soit $l = 3.10^{-10}\text{m}$. La particule fluide est modélisée par un cube d'arête de longueur a. Afin de travailler à l'échelle mésoscopique, a doit vérifier : $l \ll a \ll L$. Dans ce cas, avec $a = 10^{-6}\text{m} = 1\mu\text{m}$, la particule fluide a une masse $\text{d}m = 10^{-15}\text{kg}$, occupe un volume $\text{d}\tau = 10^{-18}\text{m}^3$ et contient environ 3.10^{10} particules.

Description Eulérienne et Lagrangienne d'un fluide

Deux descriptions sont couramment utilisées pour décrire le comportement d'un fluide, la description Lagrangienne et Eulérienne.

Dans la *description Lagrangienne* d'un fluide, les caractéristiques du fluide sont obtenues en **suivant chaque particule fluide lors de son mouvement**. Ce dernier est décrit par la connaissance des trajectoires $\boldsymbol{R}_i(t) = \boldsymbol{OM_i}(t)$, où, O est l'origine du repère euclidien utilisée et $M_i(t)$ repère la $i^{\text{ème}}$ particule fluide à la date t. La vitesse de ces particules est donnée par $\boldsymbol{V}_i(t) = \frac{d\boldsymbol{R}_i(t)}{dt}$. Cette description correspond à celle communément utilisée en mécanique du point.

Dans le cadre d'une *approche Eulérienne*, les caractéristiques du fluide sont déterminées en **chaque point fixe de l'espace**. Le mouvement du fluide est donné par la connaissance des vitesses des particules fluides passant par tout point M donné de l'espace d'étude à la date t : $\boldsymbol{v}(M,t)$. Ce dernier est un champ vectoriel. Au lieu de suivre une particule fluide dans son mouvement, un seul point M de l'espace est étudié. Il y a ainsi plusieurs particules fluides qui "passent" par M, définissant ainsi la vitesse à l'instant étudié. Ces deux approches sont reliées par : $\boldsymbol{V}_i(t) = \boldsymbol{v}(\boldsymbol{R}_i(t), t)$. Cette description est particulièrement adaptée pour décrire un espace discrétisé où la valeur des champs en chaque point du maillage est étudiée.

Dans la suite, le champ vectoriel des vitesses et le champ scalaire de pression sont principalement utilisés. Dans le cadre Eulérien, la valeur de ces champs au point M à l'instant t est noté $\boldsymbol{v}(M,t)$ et $P(M,t)$.

Lignes de courants

Le pendant de la notion de trajectoire en description Lagrangienne correspond à la notion de ligne de courant en description Eulérienne. Pour un instant donné, une ligne de courant est une courbe telle que le vecteur vitesse d'une particule fluide se situant sur cette dernière y est **tangent**. La méthode d'obtention de telles courbes est similaires à celle utilisée pour déterminer des lignes de champ électrique ou magnétique. Soit $d\boldsymbol{M}(M,t)$ un élément vectoriel tangent au point M d'une ligne de courant tracé à l'instant t. Cet élément est colinéaire au vecteur vitesse $\boldsymbol{v}(M,t)$. Ceci ce traduit par la relation :

$$d\boldsymbol{M}(M,t) \wedge \boldsymbol{v}(M,t) = \boldsymbol{0}, \tag{2.19}$$

où $\boldsymbol{0}$ est le vecteur nul, de dimension appropriée.

Dérivée particulaire

Bien que la description Eulérienne soit très élégante pour décrire le comportement d'un fluide, certaines relations ne peuvent être obtenues qu'en suivant une particule fluide lors de son mouvement. La dérivée particulaire d'une grandeur scalaire ou vectorielle associée à une particule fluide correspond à la variation de cette quantité dans le temps **lorsque la particule en question est suivie dans son mouvement**. La dérivée particulaire manipule des champs en description Eulérienne et elle est utilisée pour obtenir des relations locales décrivant la dynamique du fluide. Elle est aussi connue sous le nom de dérivée totale ou de dérivée Lagrangienne. En prenant l'exemple de la pression $P(M,t)$, sa dérivée particulaire en M à la date t est définie par :

$$\frac{\mathrm{D}P}{\mathrm{D}t}(M,t) = \lim_{\mathrm{d}t \to 0} \frac{P(M + \boldsymbol{v}(M,t)\,\mathrm{d}t, t + \mathrm{d}t) - P(M,t)}{\mathrm{d}t}. \tag{2.20}$$

L'opérateur dérivée particulaire, agissant sur une grandeur scalaire ou vectoriel associée à une particule fluide située en M à la date t, est définie par :

$$\frac{\mathrm{D}}{\mathrm{D}t} = \left(\frac{\partial}{\partial t} + \boldsymbol{v}\left(M,t\right).\boldsymbol{\nabla}\right), \tag{2.21}$$

où $\boldsymbol{\nabla}$ est l'opérateur nabla.

En analyse vectorielle, $\boldsymbol{\nabla}$ permet à lui seul de manipuler les opérateurs gradient, divergence et rotationnel sans avoir besoin de définir un système de coordonnées. Dans la suite, la notation (M,t) est omise afin de faciliter la lecture. Ces opérateurs sont ainsi illustrés par :

— $\boldsymbol{\nabla} P$, le gradient de P,
— $\boldsymbol{\nabla} \cdot \boldsymbol{v}$, la divergence de $\boldsymbol{v}$ et
— $\boldsymbol{\nabla} \wedge \boldsymbol{v}$, le rotationnel de $\boldsymbol{v}$.

La dérivée particulaire peut être décomposée en deux termes avec :

— $\frac{\partial .}{\partial t}$, la **dérivée locale** qui représente un comportement non permanent de la grandeur étudiée et
— $\boldsymbol{v} \cdot \boldsymbol{\nabla}\left(.\right)$, la **dérivée convective** (aussi appelé opérateur d'advection) qui traduit le caractère non uniforme de la grandeur étudiée.

Incompressibilité

Par construction, une particule fluide possède une masse fixe. Cependant, ce n'est pas le cas pour le volume occupé par cette dernière. Ceci conduit à une masse volumique potentiellement variable. Soit ρ la masse volumique de la particule fluide (située en M à la date t). Un fluide est qualifié d'incompressible si **son volume est constant** quelles que soient les conditions extérieures. En réalité, chaque fluide est compressible. Cependant, un liquide est bien moins compressible qu'un gaz. Pour cette raison et par abus de langage, un liquide est qualifié de fluide incompressible et un gaz est qualifié de fluide compressible.

Ensuite, un écoulement est dit incompressible si la masse volumique de chaque particule fluide est constante lors de son évolution. Un tel écoulement vérifie

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = 0, \forall\left(M,t\right). \tag{2.22}$$

Un écoulement peut ainsi être incompressible même si le fluide utilisée ne l'est pas. Pour caractériser cet écoulement, un bilan local de conservation de la masse dans un milieu sans sources est effectué (pas d'apport/retrait de fluide). L'équation suivante est obtenue :

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{v}\right) = 0. \tag{2.23}$$

En faisant intervenir la dérivée particulaire, cette dernière équation peut s'écrire sous la forme :

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho \boldsymbol{\nabla} \cdot \boldsymbol{v} = 0. \tag{2.24}$$

Ainsi, selon 2.22, un écoulement incompressible vérifie la relation

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0, \tag{2.25}$$

En *pratique*, si la vitesse du fluide est largement inférieure à la vitesse du son dans le fluide en question, l'écoulement est qualifié d'incompressible. Plus précisément, il faut que

le nombre de Mach $M_a \in \mathbb{R}_+^*$ vérifie : $M_a < 0.3$.

Dans le cadre d'un écoulement incompressible, la fonction découlement $\boldsymbol{\psi}(M,t)$ peut être définie. Ce champ vectoriel vérifie la relation :

$$\boldsymbol{v} = \boldsymbol{\nabla} \wedge \boldsymbol{\psi}. \tag{2.26}$$

D'après l'analyse vectorielle, un champ de divergence nulle peut être exprimé en fonction du rotationnel d'un autre champ. $\boldsymbol{\psi}$ est connue sous le nom de fonction de courant et permet de simplifier la résolution de certains problèmes fluide en réduisant le nombre d'inconnues.

Vorticité

Un outil communément utilisé pour décrire un écoulement est la vorticité. Elle permet de représenter le caractère tourbillonnaire local de l'écoulement. La vorticité en un point M à la date t est noté $\boldsymbol{\omega}(M,t)$. Elle est définie via le vecteur vitesse par :

$$\boldsymbol{\omega} = \frac{1}{2}\boldsymbol{\nabla} \wedge \boldsymbol{v}. \tag{2.27}$$

Le champ de vorticité est donc un champ scalaire et permet une visualisation simplifiée de l'écoulement par rapport au champ des vitesses. Son signe permet de déterminer dans quel sens le fluide **tourne localement**.

Pression et Viscosité

Un élément de surface situé à l'intérieur du fluide va subir une force provenant des molécules voisines. Cette force se décompose en une composante normale, nommée force de pression, et en une composante tangentielle, nommée force de viscosité (ou de cisaillement). La force de viscosité tend à homogénéiser la vitesse à l'intérieur d'un fluide. Par exemple, en considérant une veine de fluide lente au contact d'une veine de fluide plus rapide, la veine lente sera accélérée et la veine rapide sera ralentie. Cette force est la principale responsable pour expliquer que le miel ne s'écoule pas de la même manière que l'eau. Dans un écoulement incompressible d'un fluide homogène, les forces de cisaillement au point M à la date t sont équivalentes à une force volumique dont l'expression est

$$\boldsymbol{f}_{\text{cis}}(M,t) = \eta \boldsymbol{\nabla}^2 \boldsymbol{v}(M,t) = \eta \Delta \boldsymbol{v}(M,t), \tag{2.28}$$

où Δ est l'opérateur Laplacien et η est le coefficient de viscosité dynamique. L'unité dans le système international de ce dernier est le poiseuille (Pl), tel que 1Pl = 1Pa.s.

Traînée

La traînée est la résultante des forces exercées sur un solide immergé dans un fluide, projeté sur la direction de l'écoulement incident (l'écoulement à l'infini, non perturbé par l'obstacle). Cette dernière est notée $\gamma \in \mathbb{R}$. Elle ne prend pas en compte les effets de la pesanteur mais seulement du fluide. Dans l'air, elle est qualifiée de force de traînée aérodynamique tandis que dans l'eau elle devient la force de traînée hydrodynamique. Elle est parallèle à la direction de l'écoulement du fluide. De plus, cette force s'oppose à ce que l'objet remonte l'écoulement.

Afin de travailler avec un objet adimensionné, le coefficient de traînée est utilisé. Ce dernier est défini par :

$$C = \frac{\gamma}{\frac{1}{2}\rho S V^2} \tag{2.29}$$

où ρ est la masse volumique du fluide, V est la vitesse du fluide par rapport à l'obstacle et S est la surface de référence de l'obstacle.

Équation de Navier-Stokes

Les outils précédents permettent de décrire l'équation locale de la dynamique, incontournable en mécanique des fluides. Connue sous le nom d'équation de Navier-Stokes, elle s'exprime dans le cas d'un écoulement incompressible d'un fluide homogène ($\forall\,(M,t)$, $\rho\,(M,t) = \rho$, une constante) par

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\boldsymbol{\nabla})\boldsymbol{v} = \frac{1}{\rho}(-\boldsymbol{\nabla}P) + \nu\Delta\boldsymbol{v}, \tag{2.30}$$

où ν est la viscosité cinématique, définie par $\nu = \frac{\eta}{\rho}$ et se mesure en $m^2.s^{-1}$.

Nombre de Reynolds

Le Nombre de Reynolds est un nombre sans dimension caractérisant un écoulement de fluide homogène de vitesse moyenne v_{moy}, autour d'un obstacle (ou dans un conduit) de dimension caractéristique L. Ce nombre permet de déterminer l'importance du transfert de quantité de mouvement par convection par rapport à celui par diffusion. Le terme de convection est représenté par $(\boldsymbol{v}\cdot\boldsymbol{\nabla})\boldsymbol{v}$ dans l'équation de Navier-Stokes et celui de diffusion est représenté par $\nu\Delta\boldsymbol{v}$. En utilisant les grandeurs caractéristiques, l'ordre de grandeur du terme de convection est donné par $\frac{v_{\text{moy}}^2}{L}$. Le terme de diffusion possède l'ordre de grandeur $\nu\frac{v_{\text{moy}}}{L^2}$. Le nombre de Reynolds est ensuite défini par :

$$\mathcal{R}e = \frac{\text{“convection”}}{\text{“diffusion”}} = \frac{v_{\text{moy}}^2}{L}\frac{L^2}{\nu v_{\text{moy}}} \Rightarrow \mathcal{R}e = \frac{\rho L v_{\text{moy}}}{\eta} \tag{2.31}$$

La valeur du nombre de Reynolds renseigne sur la nature de l'écoulement. Par exemple, si $\mathcal{R}e < 1$, l'écoulement est dit *rampant*. Une caractéristique de ce type d'écoulement est que les lignes de courant glissent les unes sur les autres en restant parallèles. Un écoulement rampant nécessite généralement une vitesse *faible*. Plus de cas seront illustrés dans la section suivante.

Suivant le nombre de Reynolds, différentes expressions de la traînée peuvent être établies. Pour un nombre de Reynolds petit devant 1, la formule de Stokes peut être utilisée. La traînée γ est approximée par $\gamma = 6\pi\eta L v_{\text{moy}}$. Pour des nombres de Reynolds plus élevés, vérifiant $10^2 < \mathcal{R}e < 10^5$, la relation suivante peut être obtenue $\gamma = k\rho v_{\text{moy}}^2 L^2$, où k est une constante à déterminer. Dans le cas présent, la traînée sera proportionnelle au *carré de la vitesse*.

2.2.2 Description du système

Un écoulement d'air unidirectionnel, de vitesse v_{moy}, est considéré. Un cylindre rigide à base circulaire, de diamètre D_{iam}, est inséré dans cet écoulement telle que la direction caractéristique du cylindre soit orthogonale à celle de l'écoulement. Le rayon du cylindre est défini par R. Le nombre de Reynolds caractéristique de l'écoulement est $\frac{\rho D_{\text{iam}} v_{\text{moy}}}{\eta}$. Une coupe plane de l'écoulement est ensuite étudiée. La géométrie du problème est illustrée dans la figure 2.1.

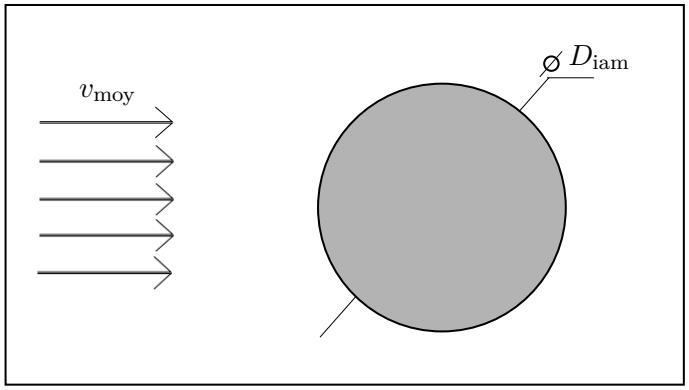

FIGURE 2.1 – Schéma du plan d'étude

L'écoulement autour d'un cylindre est un cas bien documenté de par sa nature académique. Un schéma des lignes de courant pour différentes valeurs du nombre de Reynolds, pour un instant donné, est représenté sur la figure 2.2. Comme la géométrie de l'obstacle et la nature du fluide ne changent pas, augmenter le nombre de Reynolds revient à augmenter la vitesse incidente du fluide v_{moy}. Des phénomènes complexes, difficiles à modéliser, apparaissent lorsque le nombre de Reynolds caractéristique de l'écoulement augmente. Sur la figure 2.2, les commentaires suivants peuvent être faits :

— (a) écoulement rampant (lignes de courants stationnaires et symétriques par rapport au cylindre) pour $\mathcal{R}e < 1$
— (b) apparition d'une zone de recirculation en aval du cylindre pour $\mathcal{R}e \approx 10$. Les dimensions des tourbillons augmentent avec $\mathcal{R}e$. Dans le sillage du cylindre, le champ de vitesse subit d'importantes fluctuations spatiales et temporelles.
— (c) détachement alternatif des tourbillons (dit de Bénard - Von Karman) pour $\mathcal{R}e > 49$. Cette transition est une *bifurcation de Hopf* décrite par l'équation de Stuart-Landau [7]. L'écoulement est périodique et les lignes de courant sont encore identifiables. L'écoulement n'est plus stationnaire.
— (d) pour des nombres de Reynolds de plus en plus élevés, le décollement de la couche limite se fait de plus en plus à l'amont du cylindre et les tourbillons deviennent turbulents. Dans ces zones turbulentes, le champ de vitesse devient très difficile à prédire. Avec les modélisations actuelles, une très faible variation des conditions initiales peut engendrer des écoulements notablement différents.
— (e) pour $\mathcal{R}e > 5.10^6$, l'écoulement devient encore plus complexe et les zones turbulentes s'élargissent.

Ces phénomènes apparaissent de manière continue et les bornes données sur les nombres de Reynolds sont approximatives.

Dans le cas présent, l'écoulement est supposé incompressible (v_{moy} faible devant la vitesse du son dans l'air, soit environ $340 m.s^{-1}$ à température ambiante). La vitesse du fluide incident et le diamètre du cylindre sont choisis de telle sorte que le nombre de Reynolds caractéristique de l'écoulement soit égal à 200. Ceci correspond au cas (c) de la figure 2.2. Ce type d'écoulement présente de nombreuses caractéristiques complexes tels que le décollement de couches limites, les reflux et une zone instable en aval du cylindre. Tout ceci, sans introduire de zones turbulentes.

Pour illustrer cette configuration, un champ de pression et de vorticité sont représentés sur la figure 2.3. Les données nécessaires pour réaliser ces figures ont été obtenues à partir du code de simulation présenté dans la suite. Les tourbillons de Von Kàrmàn sont

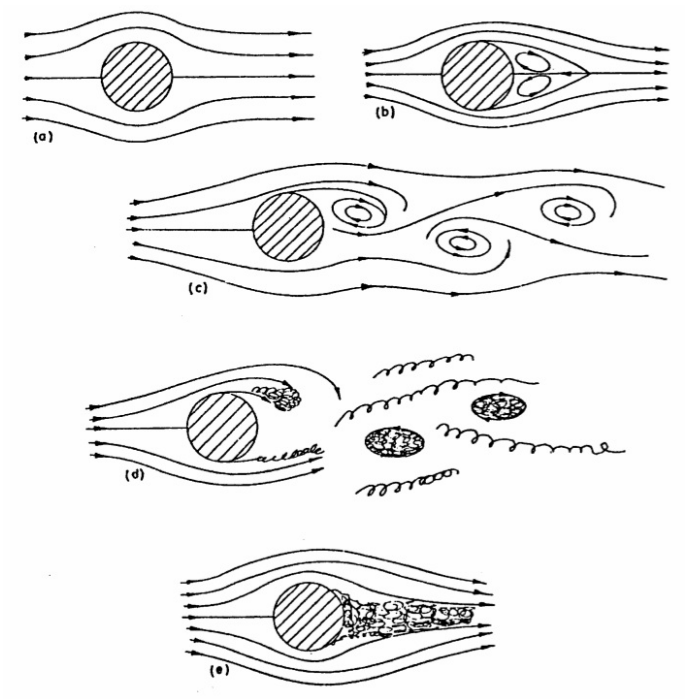

FIGURE 2.2 – Lignes de courant pour différents $\mathcal{R}e$ [8]

clairement visibles tout comme les zones de basse pression.

Outre la résolution du problème d'estimation présenté au début de ce chapitre, un autre objectif plus pratique est considéré : la réduction de la traînée subie par le cylindre par le biais d'une commande active exploitant un très faible nombre de mesures de pression en surface du cylindre. Ces objectifs sont liés puisque l'obtention d'une telle commande nécessitera une estimation de la traînée **via une estimation du champ de pression total**. Ceci permet de donner une finalité pratique au travail effectué. En utilisant l'exemple de l'aile d'avion, les raisons motivant cet objectif sont la réduction :

— de la consommation de carburant,
— de l'émission de CO_2,
— des vibrations induites par les vortex et
— de l'intensité du son émis par les moteurs.

Pour un nombre de Reynolds très faible devant 1, l'écoulement serait laminaire et les forces de pression sur la majorité de la surface du cylindre se compenserait. Ceci n'est plus le cas lors d'un écoulement à $\mathcal{R}e = 200$ où les tourbillons formés en aval déstabilisent la répartition des forces de pression.

La traînée γ est due aux forces de pression et de viscosité. Soient γ_P la traînée provenant des forces de pression et γ_ν celle provenant des forces de viscosité. Ces deux termes vérifient la relation

$$\gamma = \gamma_P + \gamma_\nu. \tag{2.32}$$

Le nombre de Reynolds de l'écoulement étudié est de 200. De ce fait, γ_P est dominante devant γ_ν.

Le même raisonnement peut être fait sur les coefficients de traînées associées C_P et C_ν. La figure 2.4 illustre l'évolution du coefficient de traînée lorsque aucune commande n'est appliquée, une fois le régime permanent atteint. Les coefficients de traînée due à la pression et à la viscosité sont aussi représentés. L'élément remarquable est la nature périodique de

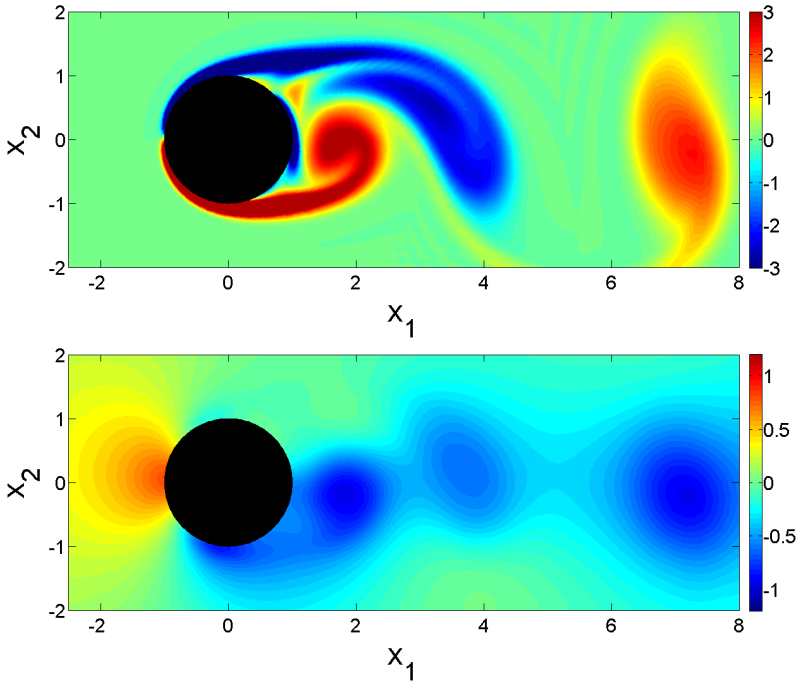

FIGURE 2.3 – Illustration de l'écoulement - vorticité (haut) , pression relative (bas)

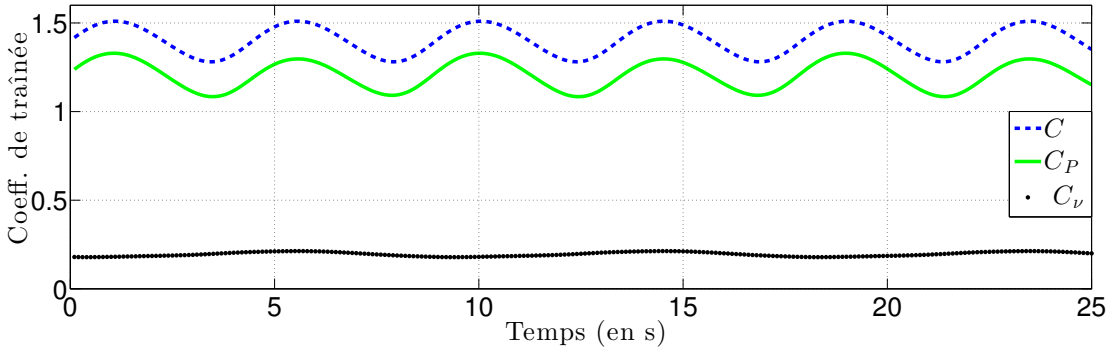

FIGURE 2.4 – Évolution du coefficient de traînée - commande nulle appliquée

leur évolution. Le spectre d'un tel signal présente une raie dominante. En effet, pour un nombre de Reynolds de 200, l'écoulement contient des tourbillons de Von Kármán. Ces dernières se décrochent de manière périodique à la fréquence dite de Von Kármán. La traînée présente une période deux fois plus faible que celle de Von Kármán. Ceci provient de la nature symétrique de l'obstacle qui fait que les tourbillons du haut et du bas agissent de manière identique sur la traînée. La valeur moyenne de C_ν est bien inférieure à celle de C_P.

Avant de pouvoir agir sur l'écoulement, il faut obtenir des informations sur l'état de ce dernier. Comme seules des mesures de pression sont disponibles, il semblerait que la composante accessible de la traînée soit uniquement celle due à la pression. Dans un premier temps, ces mesures ne seront utilisées que pour estimer le champ de pression et la traînée associée. Les chapitres 3 et 4 proposeront des algorithmes permettant d'estimer la traînée totale. Un nouveau problème, plus général, sera en effet présenté en (3.3.1).

A partir des informations recueillies, il faut pouvoir agir sur l'écoulement afin de réduire la traînée. Cette action peut être de nature passive ou active suivant les contraintes. L'approche passive peut être réalisée par une modification de la géométrie du cylindre

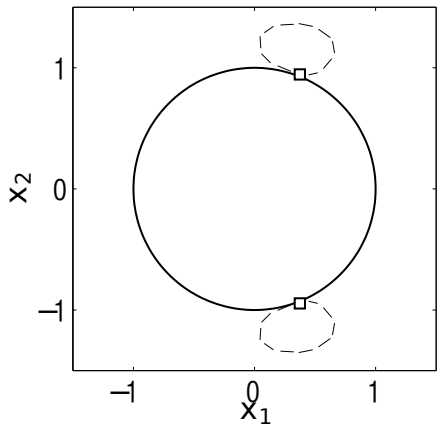

FIGURE 2.5 – Position des actionneurs et leur zone d'effet

en ajoutant des rainures ou en modifiant le revêtement de sa surface. Dans cette étude, une commande active est considérée puisqu'elle permet de s'adapter à plus de situations qu'une méthode passive et qu'elle présente aussi de meilleurs résultats, au prix d'une plus grande complexité de conception.

Parmi la pléthore de méthodes possibles [9, 10, 11, 12], tels que le chauffage du cylindre ou sa mise en rotation instationnaire autour de sa génératrice, la méthode retenue ici fait intervenir des actionneurs plasma. Seules deux pastilles sont utilisées afin de perturber l'écoulement le moins possible. Elles sont placées en aval du cylindre, de part et d'autre de l'axe $x_2 = 0$. La figure 2.5 permet d'illustrer leur placement, représenté par des carrés, ainsi que leur zone d'action, délimitée par des pointillés.

L'air au niveau des pastilles est ionisé par l'application d'une forte tension. Ce gaz ionisé est plus connu sous le nom de plasma. La tension appliquée est une représentation du champ électrique au niveau de la pastille qui va ensuite mettre en mouvement le fluide ionisé. En d'autres termes, un vent ionique dont la direction est parallèle à la surface du cylindre est créé. La zone d'action de chaque actionneur s'étend sur environ 5% de la circonférence [13]. Les contraintes de conception font que le fluide ne peut qu'être accéléré en surface, vers l'aval de l'écoulement. Ceci correspond bien à l'effet souhaité car il faut repousser le décollement de la couche limite. Un exemple de commande de ces actionneurs est une commande intermittente. Seul un actionneur est utilisé à la fois et va faciliter le décrochement du tourbillon correspondant en le *poussant*.

La commande recherchée est représentée par le vecteur

$$\boldsymbol{u}^T = (u_1\ u_2) \in \mathbb{R}_+^2. \tag{2.33}$$

u_i est considérée sans dimension et représente la tension appliquée sur la pastille. Par construction $u_i \leq 1$. Une contrainte supplémentaire est d'utiliser une loi de commande la moins énergétique possible.

Afin de visualiser l'effet de tels actionneurs, deux cas extrêmes sont représentés sur la figure 2.6. La figure de gauche est associée à une commande $\boldsymbol{u} = (1, 0)^T$. L'actionneur du haut est ainsi activé et accélère localement le fluide vers l'aval du cylindre. La figure de droite est associée à une commande $\boldsymbol{u} = (0, 1)^T$. Cette fois-ci, l'actionneur du bas est activé et accélère aussi localement le fluide vers l'aval du cylindre.

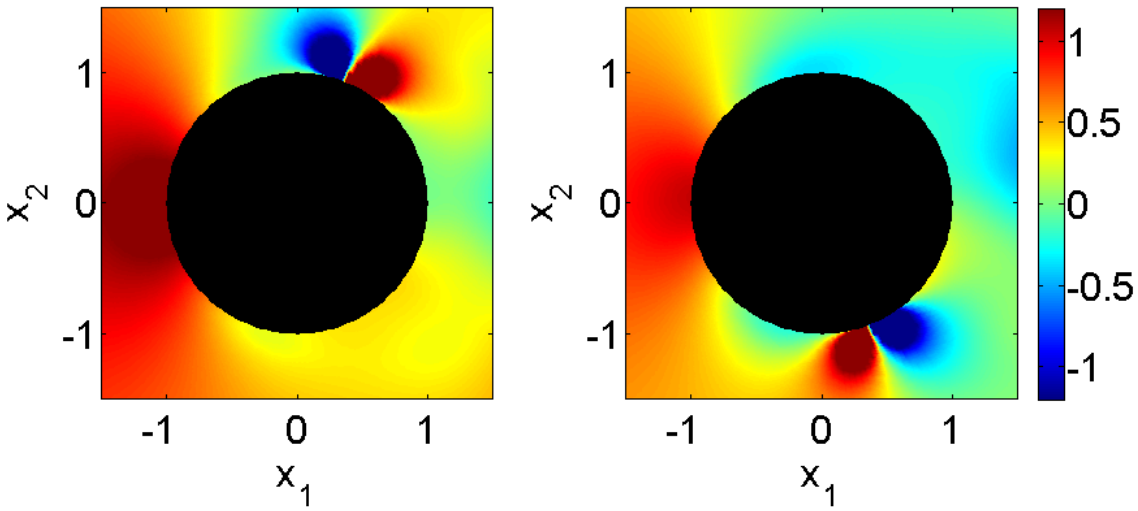

FIGURE 2.6 – Exemples de champ de pression - différentes commandes appliquées

2.2.3 Détermination de l'état du système

Mise en équation

Afin d'obtenir les données nécessaires à l'étude, une résolution numérique des inconnues de l'écoulement doit être faite. A partir des équations classiques de la mécanique des fluides rappelées dans les pages précédentes, les équations aux dérivées partielles utilisées pour décrire l'écoulement vont être élaborées.

L'étude est supposée plane afin de réduire la complexité des calculs. Ceci ne modifie pas la généralité des approches conçues. Cependant, certains phénomènes physiques secondaires sont négligés [14] .

Les inconnues naturelles de l'écoulement sont la pression P et la vitesse $\boldsymbol{v}$ (plane), soit 3 inconnues. Dans la suite, la pression relative sera manipulée. Elle est définie par rapport à la pression en amont de l'écoulement, avant la perturbation causée par l'obstacle. Ces inconnues vont être exprimées en fonction de la vorticité $\boldsymbol{\omega}$ et de la fonction de courant ψ afin de manipuler deux inconnues scalaires, simplifiant ainsi la résolution.

Un système de coordonnées cartésiennes est retenu pour obtenir les équations régissant la dynamique de l'écoulement. Le repère cartésien choisi est $(O, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ forment une base orthonormale de l'espace. Le vecteur vitesse s'exprime dans cette base sous la forme :

$$\boldsymbol{v} = v_x \boldsymbol{x} + v_y \boldsymbol{y} + v_z \boldsymbol{z}. \tag{2.34}$$

Un bon choix des axes peut faire en sorte que $v_z = 0$. De plus, une telle étude plane impose l'invariance par translation de $\boldsymbol{v}$ selon z. La vorticité associée à ce champ des vitesses est

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \wedge \boldsymbol{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_x}{\partial z}\right)\boldsymbol{x} + \left(\frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}\right)\boldsymbol{y} + \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)\boldsymbol{z} = \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)\boldsymbol{z} = \omega_z \boldsymbol{z}. \tag{2.35}$$

La vorticité est uniquement portée par $\boldsymbol{z}$ et elle permet d'obtenir la première relation liant le champ vectoriel des vitesses à un scalaire $\omega = \omega_z$:

$$\omega_z \boldsymbol{z} = \boldsymbol{\nabla} \wedge \boldsymbol{v}. \tag{2.36}$$

La fonction de courant est maintenant utilisée puisque l'écoulement est considéré incompressible. Cette dernière est définie par

$$\boldsymbol{\psi} = \psi_x \boldsymbol{x} + \psi_y \boldsymbol{y} + \psi_z \boldsymbol{z}. \tag{2.37}$$

Ensuite,

$$\boldsymbol{v} = \boldsymbol{\nabla} \wedge \boldsymbol{\psi} = \left(\frac{\partial \psi_z}{\partial y} - \frac{\partial \psi_x}{\partial z}\right)\boldsymbol{x} + \left(\frac{\partial \psi_x}{\partial z} - \frac{\partial \psi_z}{\partial x}\right)\boldsymbol{y} + \left(\frac{\partial \psi_y}{\partial x} - \frac{\partial \psi_x}{\partial y}\right)\boldsymbol{z}. \tag{2.38}$$

L'étude plane impose aussi l'invariance par translation de $\boldsymbol{\psi}$ selon z. L'expression précédente devient :

$$\boldsymbol{v} = \left(\frac{\partial \psi_z}{\partial y}\right)\boldsymbol{x} + \left(-\frac{\partial \psi_z}{\partial x}\right)\boldsymbol{y} + \left(\frac{\partial \psi_y}{\partial x} + \frac{\partial \psi_x}{\partial y}\right)\boldsymbol{z}. \tag{2.39}$$

Puisque le champ des vitesses n'est porté que selon $\boldsymbol{x}$ et $\boldsymbol{y}$, la fonction de courant peut être choisie égale à

$$\boldsymbol{\psi} = \psi_z \boldsymbol{z}. \tag{2.40}$$

La vorticité est maintenant reliée à la fonction de courant par :

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \wedge \boldsymbol{v} = \boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} \wedge \boldsymbol{\psi}) = \boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \boldsymbol{\psi}) - \Delta \boldsymbol{\psi} = \boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot (\psi_z \boldsymbol{z})) - \Delta \psi_z \boldsymbol{z}. \tag{2.41}$$

Comme ψ_z ne dépend pas de z, $\boldsymbol{\nabla} \cdot (\psi_z \boldsymbol{z}) = 0$. Après projection sur $\boldsymbol{z}$, la relation précédente devient :

$$\boxed{\omega_z = -\Delta \psi_z}. \tag{2.42}$$

L'objectif est maintenant de remanier l'équation de Navier-Stokes pour qu'elle ne fasse intervenir que la vorticité et la fonction de courant. Dans le cadre présent, celle-ci s'écrit sous la forme :

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = \frac{1}{\rho}(-\boldsymbol{\nabla} P) + \nu \boldsymbol{\nabla} \Delta \boldsymbol{v}. \tag{2.43}$$

Le rotationnel de cette relation permet d'écrire :

$$\frac{\partial \omega_z}{\partial t}\boldsymbol{z} + \boldsymbol{\nabla} \wedge (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = -\frac{1}{\rho}\boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} P) + \nu \boldsymbol{\nabla} \wedge (\Delta \boldsymbol{v}). \tag{2.44}$$

Tout d'abord, $\boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} P) = \boldsymbol{0}$. Ensuite, l'analyse vectorielle permet d'écrire $(\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = \boldsymbol{\nabla}\left(\frac{\boldsymbol{v} \cdot \boldsymbol{v}}{2}\right) + (\boldsymbol{\nabla} \wedge \boldsymbol{v}) \wedge \boldsymbol{v}$. En remarquant que $\boldsymbol{\nabla}\left(\boldsymbol{\nabla}\left(\frac{\boldsymbol{v} \cdot \boldsymbol{v}}{2}\right)\right) = 0$, le terme recherché s'écrit :

$$\boldsymbol{\nabla} \wedge (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = \boldsymbol{\nabla} \wedge ((\boldsymbol{\nabla} \wedge \boldsymbol{v}) \wedge \boldsymbol{v}). \tag{2.45}$$

Après simplification, cette relation devient :

$$\boldsymbol{\nabla} \wedge (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v} = (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{\nabla} \wedge \boldsymbol{v} - (\boldsymbol{\nabla} \wedge \boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v}. \tag{2.46}$$

Le calcul se simplifie ensuite en remarquant que $\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$ et $\boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} \wedge \boldsymbol{v}) = 0$. Enfin, en introduisant la vorticité puis en projetant l'équation 2.44 sur $\boldsymbol{z}$ (en remarquant que $((\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{v}) \cdot \boldsymbol{z} = 0$), la relation suivante est obtenue :

$$\frac{\partial \omega_z}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \omega_z = \nu \Delta \omega_z. \tag{2.47}$$

En exprimant la vitesse avec la fonction d'écoulement, la relation recherchée est :

$$\boxed{\frac{\partial \omega_z}{\partial t} + \boldsymbol{\nabla} \wedge (\psi_z \boldsymbol{z}) \cdot \boldsymbol{\nabla} \omega_z = \eta \Delta \omega_z}. \tag{2.48}$$

Voici enfin le système d'équations à résoudre :

$$\begin{cases} \dfrac{\partial \omega_z}{\partial t} + \boldsymbol{\nabla} \wedge (\psi_z \boldsymbol{z}) \cdot \boldsymbol{\nabla} \omega_z = \eta \Delta \omega_z & (a) \\ \omega_z = -\Delta \psi_z & (b) \end{cases} \tag{2.49}$$

Résolution numérique

Pour obtenir les valeurs de P et de $\boldsymbol{v}$ en tout point de l'écoulement, il est nécessaire de résoudre les équations 2.49 (dont les inconnues sont ψ_z et ω_z). Vu leur complexité, une résolution directe est inconcevable. Comme ces relations sont des équations aux dérivées partielles, la méthode des éléments finis est particulièrement adaptée pour effectuer cette résolution. Les méthodes volumes finies et différences finies sont aussi concernées. Par contre, les écoulements fluides présentent la particularité d'avoir des variations spatiales et temporelles brutales, résultant en un espace discrétisé de grande dimension afin d'effectuer une description de qualité.

Vu la nature de l'obstacle, un maillage cylindrique est particulièrement adapté au problème. En utilisant un tel maillage contenant 201 cercles concentriques composés de 201 points identiquement espacés chacun, le nombre de nœuds est de 40 401. La variation du rayon de ces cercles n'est pas linéaire afin de mailler plus précisément les zones complexes de création des vortex et de décollement de couches limites. Il y a ainsi une concentration de cercles dans le voisinage du cylindre. L'espacement entre deux cercles consécutifs suit une loi géométrique de raison 1.0172. La limite du domaine d'étude est un cercle de diamètre $30D_{\text{iam}}$, centré sur la génératrice du cylindre. La figure 2.7 illustre le principe du maillage retenu, avec bien moins de points de maillage afin que la figure soit exploitable.

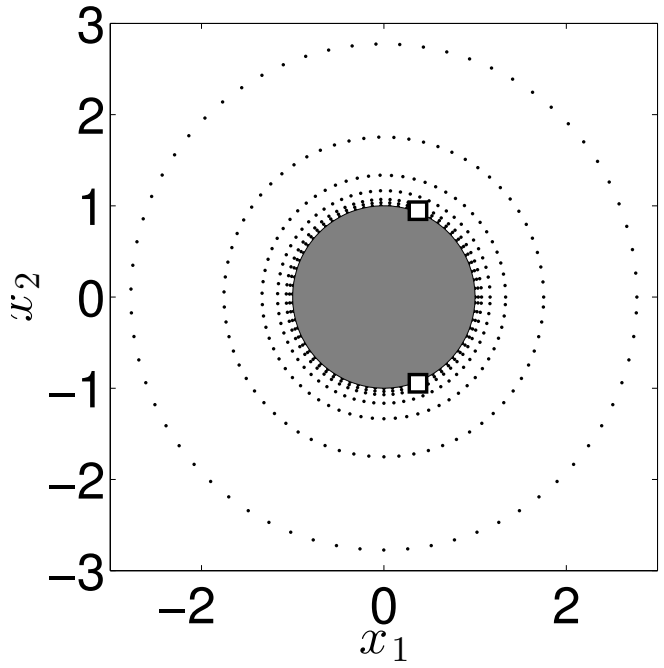

FIGURE 2.7 – Schéma simplifié du maillage

Une telle simulation nécessite aussi une discrétisation temporelle. Elle doit être assez fine afin de pouvoir correctement simuler le comportement du fluide. Par contre, ceci se fait au prix d'un plus grand temps de calcul pour une même durée simulée. Un compromis est ainsi recherché. Vu qu'un nombre de Reynolds de 200 est utilisé, un pas de discrétisation temporelle de $\delta t = 0.1$s est satisfaisant. Ensuite, la détermination des paramètres de l'écoulement n'a pas besoin d'être obtenue avec une grande finesse. Ces derniers sont donc calculés tous les $t_{simu} = 4\delta t = 0.4s$ dans les cas ici traités.

Afin de contourner les obstacles calculatoires dus au grand nombre d'inconnues, une

approche consiste à réduire la dimension du problème. Le programme retenu utilise la méthode Proper Orthogonal Decomposition (POD) détaillée dans [9]. Cette méthode de réduction de la dimension est essentielle à la suite de l'étude et sera décrite dans la section suivante. Le code permet de récupérer le champ de pression, le champ de vitesse, le champ de vorticité et la fonction courant à chaque instant. Des compléments techniques peuvent être trouvés dans [15]

L'effet des actionneurs est modélisé par l'ajout d'une source de quantité de mouvement dans l'écoulement du fluide. Dans une certaine zone d'action, située au voisinage de chaque actionneur, la quantité de mouvement des particules fluides est imposée. Une modélisation plus complexe d'un tel actionneur plasma (débutant à partir de la tension de commande réellement appliquée) en vue d'être intégré dans un processus de commande est encore un problème d'actualité [16].

Les caractéristiques physiques et géométriques des capteurs ne sont pas intégrées dans la simulation. Cela implique que leur effet sur l'écoulement n'est pas pris en compte. Les mesures sont formées artificiellement en récupérant les valeurs du champ de pression aux points correspondant aux positions souhaitées des capteurs. Pour simuler l'effet des capteurs sur l'écoulement, leur géométrie devrait être incorporée à celle du cylindre.

La connaissance de l'écoulement dans sa totalité n'est pas nécessaire pour déterminer le coefficient de traînée de pression C_P. La seule connaissance de la pression à la surface du cylindre suffit pour la calculer. Cependant, comme l'un des intérêts de cet exemple pratique est d'illustrer une méthode d'estimation, le champ de pression dans un voisinage immédiat est étudié. Dans les deux cas (étude de surface ($n_{\boldsymbol{x}} = 201$) ou étude de voisinage ($n_{\boldsymbol{x}} = 3015$)), la condition $n_{\boldsymbol{x}} \gg n_S$ est vérifiée afin d'obtenir une qualité d'estimation suffisante sur C_P. Le nombre de capteurs retenus ne dépassera pas $n_S = 10$.

Le domaine $\mathcal{C}$ dont la pression est à estimer est une couronne centrée sur le cylindre. Le rayon intérieur est celui du cylindre, englobant ainsi sa surface, et le rayon extérieur est choisi égal à $R + R/4$. L'espace est normalisé par rapport au rayon du cylindre. Ce rayon normalisé est alors de 1 (sans dimension). Le domaine est ainsi caractérisé par $\mathcal{C} = \{\boldsymbol{x} \in \Omega, \|\boldsymbol{x}\|_2 \leq 1.25\} \subset \Omega$. Ce choix a été fait afin d'inclure les zones d'effet des actionneurs plasma dans le domaine d'étude. Le champ de pression étudié est ainsi défini par

$$\boldsymbol{y} = \{P(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{C}\} \in \mathbb{R}^{n_{\boldsymbol{x}}}, \tag{2.50}$$

avec $n_{\boldsymbol{x}} = 3015$.

Le code de simulation est ensuite utilisé pour générer la matrice des *snapshots* Y. Cette matrice doit contenir des informations sur les états du système susceptibles d'être rencontrés en pratique. Cette collection de *snapshots* sert à décrire l'espace dans lequel le champ de pression évolue. Sachant que $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{x}}}$, sans information supplémentaire, $\boldsymbol{y}$ évolue dans un espace de dimension $n_{\boldsymbol{x}}$. Par contre, en pratique, de nombreuses configurations de $\boldsymbol{y}$ sont inaccessibles pour des raisons physiques. Il est impossible d'obtenir un champ de pression comportant uniquement des valeurs négatives par exemple. De même, certaines évolutions temporelles sont impossibles telles que des discontinuités du champ de pression. Il est probable qu'un espace de dimension plus faible que celui de l'espace canonique puisse contenir les différentes réalisations de $\boldsymbol{y}$. En utilisant l'exemple de la réduction de la traînée, les *snapshots* retenus doivent contenir les champs les plus susceptibles d'être rencontrés lors de l'application d'une commande *performante*.

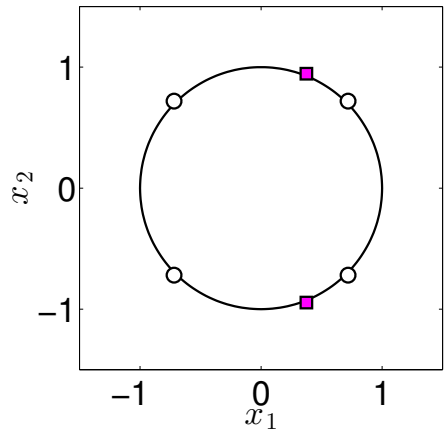

FIGURE 2.8 – Positions $\mathcal{C}_{\mathrm{u}4}$

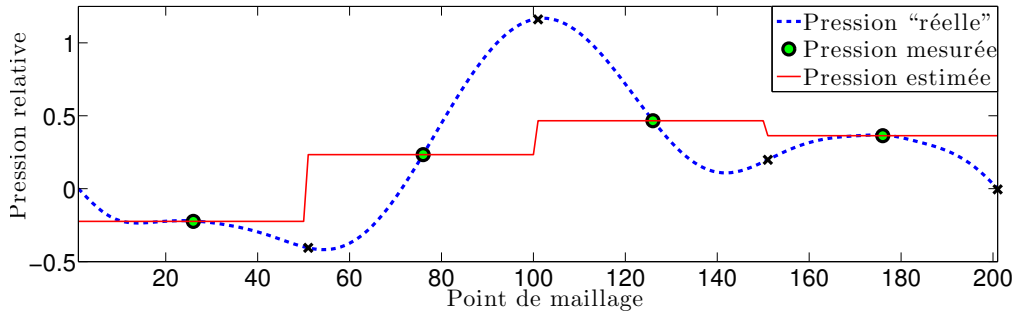

FIGURE 2.9 – Approche naïve - Interpolation constante de la pression

2.3 Approche classique

Cette section a pour objectif de présenter les manières les plus courantes pour résoudre le problème *Pb.Estimation*. La première partie concerne une approche qualifiée de naïve et illustre clairement la difficulté du problème. La second partie concerne la méthode POD et sa mise en pratique pour résoudre le problème.

2.3.1 Approche naïve

Ces prochaines lignes ont pour objectif de justifier l'utilisation d'une approche plus complexe exploitant la totalité de la séquence d'entraînement. A première vue, une interpolation à partir des mesures de pression, pour rappel $\boldsymbol{s}$, permet d'estimer la traînée provenant de la pression γ_P. Dans la suite, sans mention du contraire, la traînée fera référence à celle de **pression**, et non de viscosité. La séquence test est ici associée à une commande nulle. Seulement 4 capteurs sont considérés et ils sont ici disposés uniformément le long de la surface. La figure 2.8 illustre ce placement. Les capteurs sont représentés par des **cercles** et les actionneurs par des **carrés**.

La première interpolation utilisée est une interpolation dite constante. La pression au voisinage d'un capteur est supposée identique à celle donnée par la mesure. La figure 2.9 permet d'illustrer cette interpolation dans le cas de 4 capteurs uniformément répartis le long de la surface du cylindre, pour un *snapshot* donné.

La seconde interpolation considérée est une interpolation linéaire. La pression entre deux capteurs consécutifs (lorsqu'on se déplace continuement le long de la surface du cy-

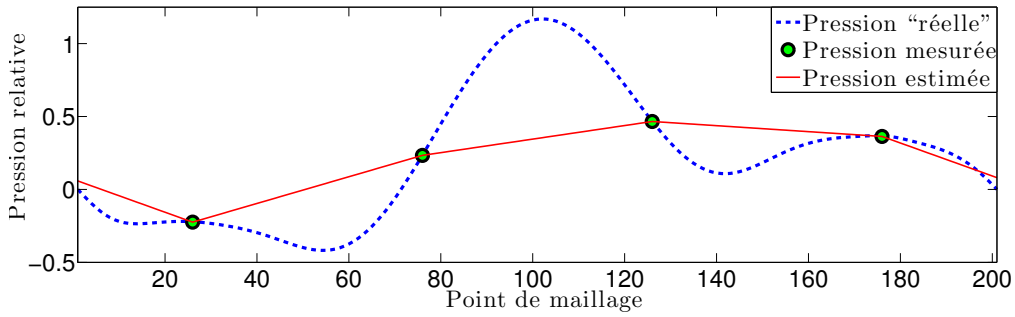

FIGURE 2.10 – Approche naïve - Interpolation linéaire de la pression

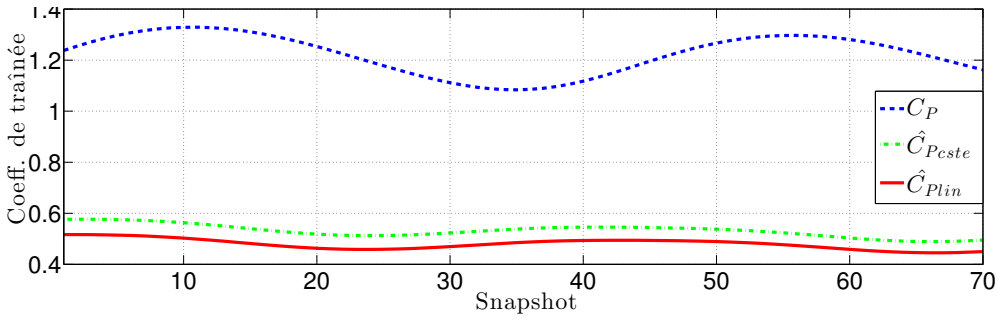

FIGURE 2.11 – Approche naïve - Estimation du coefficient de traînée

lindre) est supposée varier linéairement. Les résultats obtenus pour la même disposition des capteurs et pour le même *snapshot* sont représentés sur la figure 2.10.

L'axe des abscisses (pour les figures 2.9 et 2.10) est associé à l'index des points du maillage sur la surface du cylindre.

Pour cette même disposition des capteurs, ces deux méthodes d'interpolations sont utilisées pour estimer le coefficient de traînée de pression C_P. Soient $\hat{C}_{Pcste}$ et $\hat{C}_{Plin}$ les traînées de pression estimées. Au vu des courbes représentant $\hat{C}_P$ dans la figure 2.11, ces méthodes ne sont pas fiables. Elles sont ici incapables d'estimer correctement la traînée. Ceci est encore plus visible lorsque peu de capteurs sont utilisés ou lorsque ces derniers sont mal placés. Pour les écoulements considérés ici, plus il y a de capteurs en aval du cylindre, plus ces approches seront exploitables.

Étant donné que l'une des contraintes imposées lors de cette étude est l'utilisation d'un faible nombre de capteurs, ces approches à base d'interpolation naïve sont à éviter. Bien évidemment, un meilleur placement que celui utilisé ici peut être retenu, mais même avec cette modification, une telle approche naïve ne pourra jamais donner d'aussi bons résultats que les méthodes présentées dans la suite.

En exploitant la **totalité de la séquence d'entraînement**, une collection de champs permettant de décomposer la pression va être calculée. Ainsi, l'interpolation de la pression sera effectuée à partir de ces vecteurs, **ces modes**, permettant une bien meilleure estimation qu'une interpolation linéaire classique.

2.3.2 Approche POD

Une approche très répandue pour résoudre le problème *Pb.Estimation* est de trouver une base de faible dimension permettant d'approximer l'espace dans lequel évolue $\boldsymbol{y}$. Ceci est une approche classique qui consiste à réduire la taille du problème. Soit $\Psi \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_D}$

la base en question où n_D est le nombre de vecteurs composant cette base, appelés encore modes. En notant $\boldsymbol{\psi}_i \in \mathbb{R}^{n_{\boldsymbol{x}}}$ le $i^{\text{ème}}$ mode, la matrice Ψ s'écrit :

$$\Psi = \left(\boldsymbol{\psi}_1 \dots \boldsymbol{\psi}_{n_D}\right). \tag{2.51}$$

Il faut remarquer qu'une telle collection de vecteurs n'est pas une base de $\mathbb{R}^{n_{\boldsymbol{x}}}$ à proprement parler. Cette dernière forme une famille libre mais n'est pas génératrice de $\mathbb{R}^{n_{\boldsymbol{x}}}$. Cependant, cette collection de vecteurs est, par abus, appelée une base. Chaque vecteur $\boldsymbol{y}$ est décrit en fonction de sa décomposition sur Ψ, noté $\boldsymbol{a} \in \mathbb{R}^{n_D}$. Ce vecteur $\boldsymbol{a}$ est choisi comme celui qui minimise la norme suivante

$$||\boldsymbol{y} - \Psi\boldsymbol{a}||_2. \tag{2.52}$$

La matrice

$$A = (\boldsymbol{a}_1 \dots \boldsymbol{a}_{n_{\text{snap}}}) \tag{2.53}$$

contient ces décompositions. $\boldsymbol{a}$ est aussi appelée une représentation réduite.

Pour un instant donné, à partir des n_S mesures, il faudra estimer les n_D termes contenus dans $\boldsymbol{a}$ pour obtenir une estimée de $\boldsymbol{y}$. Une représentation réduite $\boldsymbol{a}$ est ainsi plus accessible que $\boldsymbol{y}$ à partir des seules mesures $\boldsymbol{s}$. En effet, moins de termes doivent être estimés (n_D au lieu de $n_{\boldsymbol{x}}$) à partir des n_S mesures. Soit $\widehat{\boldsymbol{a}}$ l'estimée de $\boldsymbol{a}$ obtenue à partir des n_S mesures. Ensuite, $\widehat{\boldsymbol{y}}$, l'estimée de $\boldsymbol{y}$, est définie par

$$\widehat{\boldsymbol{y}} = \Psi\widehat{\boldsymbol{a}}. \tag{2.54}$$

Ces vecteurs sont regroupés dans la matrice

$$\widehat{A} = \left(\widehat{\boldsymbol{a}}_1 \dots \widehat{\boldsymbol{a}}_{n_{\text{snap}}}\right). \tag{2.55}$$

La relation matricielle suivante est enfin obtenue : $\widehat{Y} = \Psi\widehat{A}$.

Les inconnues du problème d'estimation sont maintenant la base Ψ et une procédure pour obtenir une estimée de la représentation réduite $\widehat{A}$.

La méthode la plus utilisée [7, 17, 18, 19] pour déterminer Ψ est la méthode Proper Orthogonal Decomposition (POD). Les spécificités de cette dernière sont décrites ci-dessous et plus de compléments peuvent être trouvés dans [1, 20]. La POD est une technique efficace d'analyse de données. C'est une procédure linéaire qui consiste à déterminer une famille de modes propres orthogonaux représentatifs d'une collection de données. Cette méthode est aussi appelée Principal Component Analysis (PCA) et possède de nombreux points communs avec l'approximation de Karhunen-Loéve.

En considérant le champ $f(\boldsymbol{x}_i, t)$ présenté au début de ce chapitre, l'approche POD consiste à déterminer une série de fonctions permettant de l'approximer. Soient $\{\phi_i\}_{i\in[1,n_D]}$ les n_D modes spatiaux et $\{a_i\}_{i\in[1,n_D]}$ les coefficients temporels. Ces fonctions sont à valeurs réelles. L'approximation est décrite de la manière suivante :

$$f(\mathbf{x}, t) \approx \sum_{i=1}^{n_D} \phi_i(\mathbf{x}) a_i(t) \tag{2.56}$$

Les $\{\phi_i\}_{i\in[1,n_D]}$ sont aussi appelées modes POD et ont la propriété de former une famille orthogonale (après avoir défini un produit scalaire sur l'espace des fonctions de carré intégrable L^2). Cette propriété permet l'obtention de $\boldsymbol{a}$ à partir de $\boldsymbol{y}$ à l'aide d'une projection orthogonale.

La détermination de telles fonctions de manière exacte est délicate et requiert la connaissance parfaite de $f(\boldsymbol{x}, t)$. Elle fait aussi intervenir la résolution de problèmes vecteurs propres / valeurs propres de grandes tailles. La méthode des *snapshots*, proposée par Sirovich [21], est couramment utilisée en pratique. Elle est parfaitement adaptée au cadre actuel où seul Y est disponible. Elle permet d'obtenir directement une matrice Ψ, sans avoir besoin de chercher une famille de fonctions. Soit $U\Sigma V^T$ la décomposition en valeurs singulières (SVD pour Singular Value Decomposition) de la matrice Y. Plus d'informations sur la SVD peuvent se trouver dans le livre [22]. $U \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}}$ contient les vecteurs singuliers à gauche, $V \in \mathbb{R}^{n_{\text{snap}} \times n_{\boldsymbol{x}}}$ contient les vecteurs singuliers à droite et Σ est une matrice diagonale contenant les valeurs singulières. Puisque les valeurs singulières sont classées dans un ordre décroissant par construction, en ne retenant que les n_D premiers modes, une décomposition de Y est donnée par

$$Y_{n_D} = U_{n_D} \Sigma_{n_D} V_{n_D}^T, \tag{2.57}$$

où U_{n_D} (respectivement V_{n_D}) est la restriction de U (respectivement V) au n_D premières colonnes et Σ_{n_D} est la restriction de Σ aux n_D premières lignes et n_D premières colonnes. Plus n_D est grand, meilleure sera l'approximation de Y au prix d'une base Ψ plus grande. Selon de théorème d'Eckart-Young [22], la matrice Y_{n_D} est la meilleure matrice de rang n_D qui minimise l'erreur $||Y - Y_{n_D}||_F$. Cette réduction étant une étape essentielle, une SVD réduite de Y peut être calculée pour réduire les calculs nécessaires. La matrice contenant les représentations réduites est obtenue en effectuant la projection orthogonale de chaque *snapshot* sur les vecteurs de Ψ. Elle est aussi donnée par la SVD de Y via $A_{n_D} = \Sigma_{n_D} V_{n_D}^T$. La méthode des *snapshots* permet donc, via une SVD, d'obtenir la matrice Ψ.

Il ne manque plus que la méthode d'obtention de $\boldsymbol{a}$ à partir des mesures. En effet, $\boldsymbol{y}$ ne sera pas disponible en pratique, seul $\boldsymbol{s}$ sera disponible. En supposant que $\boldsymbol{a}$ soit connu, il est possible de le relier aux mesures $\boldsymbol{s}$. En effet, $\boldsymbol{y} \approx \Psi \boldsymbol{a}$ (l'approximation venant du fait qu'un nombre limité de modes est utilisé). Ensuite, en utilisant la matrice des capteurs C, la relation suivante peut être écrite :

$$\boldsymbol{s} \approx C\Psi\boldsymbol{a}. \tag{2.58}$$

En s'inspirant de cette relation, la procédure la plus communément utilisée pour obtenir une estimée de $\boldsymbol{a}$ à partir de $\boldsymbol{s}$ consiste à résoudre le problème suivant :

$$\widehat{\boldsymbol{a}} = \underset{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_D}}{\arg\min} ||\boldsymbol{s} - C\Psi\widetilde{\boldsymbol{a}}||_2. \tag{2.59}$$

Ceci correspond à la minimisation d'un critère moindre carré. Pour garantir un problème bien posé, il faut qu'il y ait autant d'inconnues que d'équations. Ceci impose la contrainte suivante :

$$n_S \geq n_D. \tag{2.60}$$

Les hypothèses de départ font que le nombre de capteurs utilisés sera *faible*. Ainsi, pour exploiter le plus d'information possible, il semble intéressant d'utiliser une base ayant $n_D = n_S$. Ce point sera illustré dans la suite. Un élément essentiel de ce problème concerne les propriétés de la matrice $C\Psi$. Si la condition $n_D = n_S$ est vérifiée, la matrice $C\Psi$ est une matrice carrée et il n'y a qu'une seule solution $\widehat{\boldsymbol{a}} = (C\Psi)^{-1}\boldsymbol{s}$. Le choix des capteurs est fait de manière à assurer l'inversibilité de $C\Psi$. Dans le cas où $n_D \neq n_S$, une estimée $\widehat{\boldsymbol{a}}$ peut quand même être choisie en utilisant la pseudo-inverse de Moore-Penrose [22], aussi connue sous le nom d'inverse généralisée. Elle sera nommée pseudo-inverse dans la suite.

Soit $A \in \mathbb{R}^{n \times m}$ un matrice réelle quelconque où $n \in \mathbb{N}^*$ correspond au nombre de lignes et $m \in \mathbb{N}^*$ correspond au nombre de colonnes. La pseudo-inverse de A est l'unique matrice $X \in \mathbb{R}^{m \times n}$ telle que les 4 conditions suivantes soient satisfaites :

— $AXA = A$,
— $XAX = X$,
— $(AX)^T = AX$ et
— $(XA)^T = XA$.

La matrice X est aussi notée A^+ pour indiquer son lien avec A. Dans le cas où A est une matrice carrée, sous réserve d'existence de son inverse A^{-1}, la pseudo-inverse vérifie $A^+ = A^{-1}$. L'article [23] propose de nombreuses méthodes de calcul de la pseudo-inverse.

Un autre point important est le rôle de la pseudo-inverse dans la résolution de systèmes linéaires mal-posés. Soient $A \in \mathbb{R}^{n\times m}$, $\boldsymbol{b} \in \mathbb{R}^m$ et $\boldsymbol{x} \in \mathbb{R}^n$. A et $\boldsymbol{b}$ sont connus et $n > m$. Le vecteur $\boldsymbol{x}$ est obtenu à travers le problème :

$$\boldsymbol{x} = \underset{\widetilde{\boldsymbol{x}}\in\mathbb{R}^n}{\arg\min}\, ||A\widetilde{\boldsymbol{x}} - \boldsymbol{b}||_2 \tag{2.61}$$

Vu que $n > m$, il y a plus d'inconnues (composantes de $\boldsymbol{x}$) que d'équations (m). Il n'existe alors pas de solution unique à ce problème mais une infinité. Cependant, parmi ces solutions, l'unique vecteur $\boldsymbol{x}$ de norme 2 minimale est donné par $A^+\boldsymbol{b}$. Ainsi, lors de la résolution de ce type de problème, la solution de norme 2 minimale est très souvent utilisée. Outre sa facilité d'obtention, via la pseudo-inverse, elle minimise l'énergie du résidu $||A\boldsymbol{x} - \boldsymbol{b}||_2$ régularisant ainsi la solution.

A l'aide de cette opération, la relation générale suivante est obtenue

$$\widehat{\boldsymbol{a}} = (C\Psi)^+\boldsymbol{s}. \tag{2.62}$$

Cette expression permet aussi de définir l'opérateur $\Psi\,(.)$ du problème *Pb.Estimation* par

$$\Psi\,(\boldsymbol{s}) = \widehat{\boldsymbol{a}} = (C\Psi)^+\boldsymbol{s}, \forall \boldsymbol{s} \in \mathbb{R}^{n_S}. \tag{2.63}$$

Cet opérateur est linéaire et peut être représenté par la seule matrice $(C\Psi)^+$. La position des capteurs, via la matrice C, joue ainsi un rôle primordial sur la qualité de l'estimation. Ceci sera aussi illustré dans la section suivante.

Enfin, la procédure complète est maintenant formalisable.
— (Offline) Déterminer Ψ en effectuant une SVD réduite de Y d'ordre n_S
— (Online) Déterminer $\widehat{\boldsymbol{y}}$ à partir de $\boldsymbol{s}$ selon : $\widehat{\boldsymbol{y}} = \Psi(C\Psi)^+\boldsymbol{s}$

La première étape est réalisée hors-ligne. Elle est calculatoire et est réalisée une fois pour toute à partir de la séquence d'entraînement. Ensuite, la seconde étape est utilisée en temps réel à partir des mesures réellement effectuées sur le système. Elle ne fait intervenir que des opérations rapides. Ceci est donc la méthode la plus utilisée pour estimer un vecteur de grande dimension à partir de peu de mesures, avec seulement une séquence d'apprentissage comme donnée.

La section suivante va illustrer les performances de cette approche. L'effet des différents paramètres sera aussi étudié.

2.4 Caractérisation des performances

Les performances de la méthode POD sur le système test sont illustrées ci-dessous.

2.4.1 Données

Les sous-sections suivantes vont définir les **données** qui seront utilisées dans la majorité des expériences réalisées lors de ce travail de thèse. Ces données concernent :

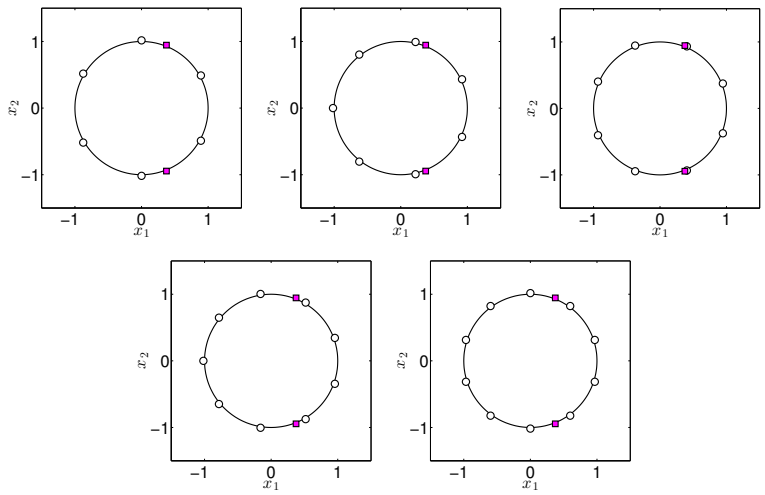

FIGURE 2.12 – Positions $\mathcal{C}_{\mathrm{u}}$ - $6 \leq n_S \leq 10$

— la position des capteurs,
— la séquence d'apprentissage,
— et la mesure des performances.

Positions retenues des capteurs : $\mathcal{C}_{\mathbf{opt}}$, $\mathcal{C}_{\mathbf{f}}$ et $\mathcal{C}_{\mathbf{u}}$

Dans un premier temps, 3 types de configurations différentes sont retenues. Les figures 2.12, 2.13 et 2.14 permettent d'illustrer ces différentes positions. Les capteurs sont représentés par des **cercles** et les actionneurs par des **carrés**.

La première configuration correspond à un **placement uniforme** des capteurs le long de la surface. Ces positions seront dénotées dans la suite par $\mathcal{C}_{\mathrm{u}}$ ou $\mathcal{C}_{\mathrm{u}n_S}$ si le nombre de capteurs utilisés est spécifié.
L'écart angulaire entre deux capteurs successifs est identique. 5 dispositions correspondant aux cas $6 \leq n_S \leq 10$ sont représentées sur la figure 2.12. Un *grand nombre* de capteurs est ici utilisé.

La seconde configuration, présentée dans la figure 2.13, correspond à un placement sans connaissance particulière sur le système. Ces positions sont nommées **fixes** et seront dénotées par $\mathcal{C}_{\mathrm{f}}$ ou $\mathcal{C}_{\mathrm{f}n_S}$ si le nombre de capteurs utilisés est spécifié.
Ce placement est sous-optimal et non-adapté pour l'estimation du champ de pression. Chaque capteur placé n'est jamais retiré. Un *faible nombre* de capteurs est disponible et $1 \leq n_S \leq 5$.

La dernière configuration correspond à un placement plus *a posteriori* des capteurs, représentés sur la figure 2.14. Ces positions sont nommées, par abus, **positions optimales**, et seront dénotées par $\mathcal{C}_{\mathrm{opt}}$ ou $\mathcal{C}_{\mathrm{opt}_{n_S}}$ si le nombre de capteurs utilisés est spécifié.
Une fois encore, au plus 5 capteurs peuvent être utilisés. Les capteurs sont ici concentrés dans les zones informatives pour la tâche retenue. Ces positions ont été obtenues avec l'algorithme SENSORSPACE qui sera présenté dans le chapitre 5.

Séquences d'apprentissage : $\mathcal{S}_{\mathrm{l}}$ et $\mathcal{S}_{\mathrm{c}}$

Plusieurs commandes génériques ont été appliquées et la configuration du champ de pression a été enregistrée une fois toutes les 0.4 unités de temps. Parmi ces dernières, une

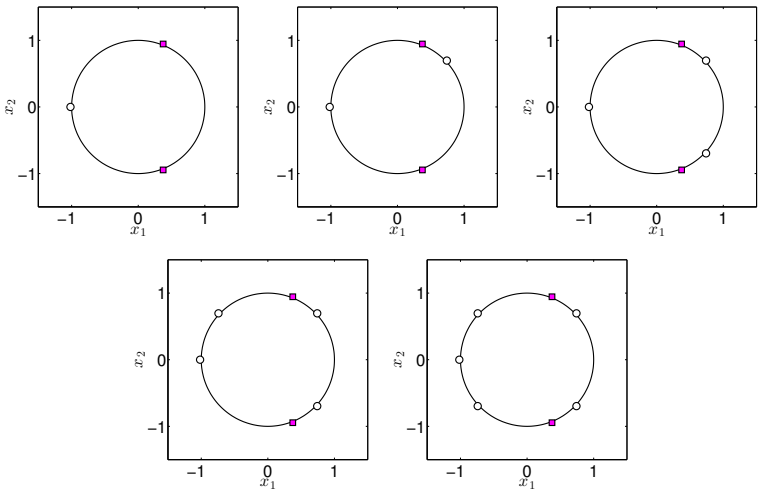

FIGURE 2.13 – Positions $\mathcal{C}_{\mathrm{f}}$ - $1 \leq n_S \leq 5$

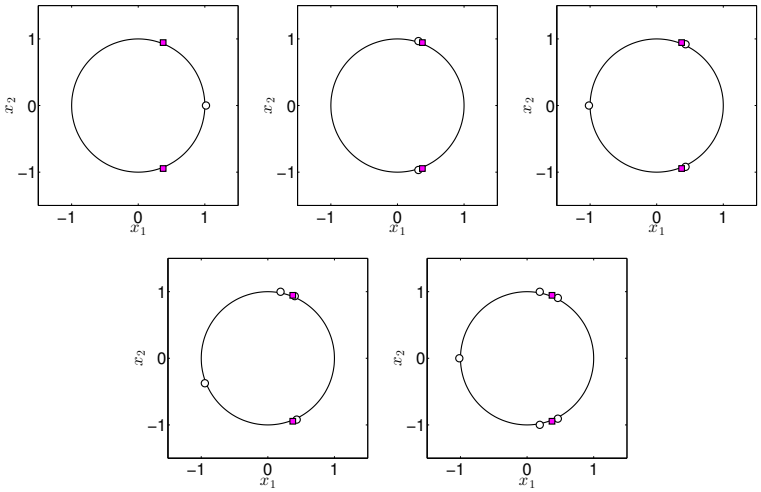

FIGURE 2.14 – Positions $\mathcal{C}_{\mathrm{opt}}$ - $1 \leq n_S \leq 5$

commande nulle et des commandes constantes ont été appliquées. Les phases transitoires et permanentes sont capturées afin de récupérer le plus d'information possible. Ensuite, une commande de nature similaire à celle de la *commande performante attendue* est appliquée. Comme la traînée subie par le cylindre est principalement due au décrochement des vortex, une commande présentant la même périodicité que cette dernière (période de von Kàrmàn) semble très prometteuse. Les actionneurs sont activés l'un après l'autre et leur commande est d'allure sinusoïdale. Ceci permet de maintenir un écoulement *symétrique* et de faciliter le décrochement du tourbillon responsable de l'augmentation de la traînée. Plus de configurations peuvent être ajoutées dans la matrice Y. Par exemple, des dynamiques plus complexes peuvent être excitées grâce à des séquences binaires pseudo-aléatoires, des chirps et des multi-sinusoïdes.

Ici, deux séquences d'entraînement ont été générées. Une séquence dite ***courte*** notée $\mathcal{S}_\mathrm{c}$ (2.15) et une autre dite ***longue*** notée $\mathcal{S}_\mathrm{l}$ (2.16). Pour chaque type de séquence, la commande appliquée ainsi que la traînée totale γ associée seront représentées.

La séquence *courte* $\mathcal{S}_\mathrm{c}$ correspond à une durée *faible* d'application des commandes *sinusoïdales* où les dynamiques mises en œuvre n'ont pas eu le temps de pleinement s'exprimer. $n_\mathrm{snap} = 1309$ *snapshots* ont été retenus. Les données relatives à cette séquence se trouvent dans la figure 2.15.

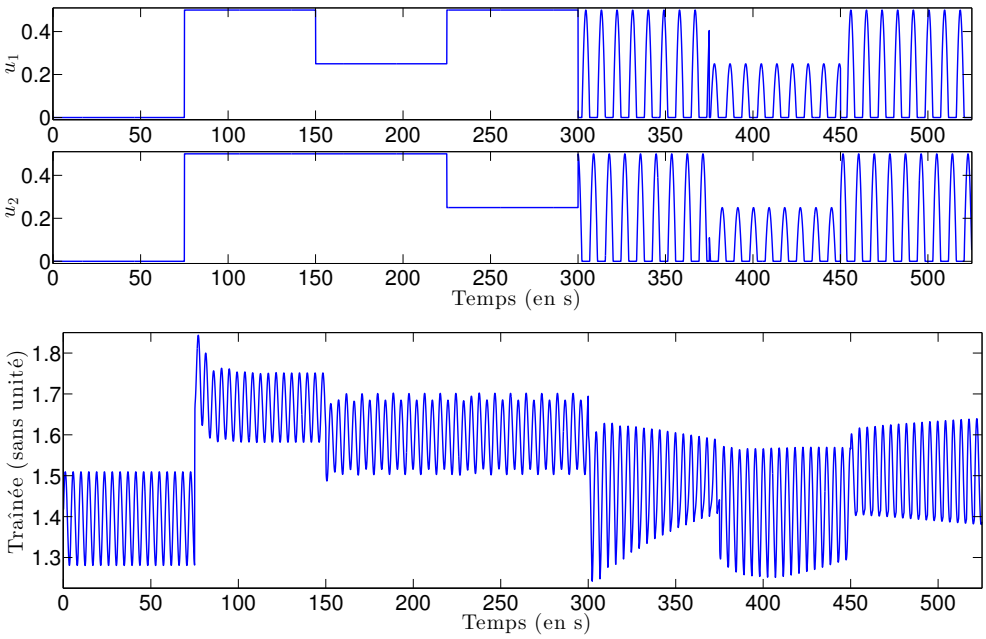

FIGURE 2.15 – Séquence d'apprentissage $\mathcal{S}_\mathrm{c}$ - commande (haut) , coefficient de traînée (bas)

La séquence *longue* $\mathcal{S}_\mathrm{l}$ est associée à un temps *long* d'application des commandes *sinusoïdales*. $n_\mathrm{snap} = 1875$ *snapshots* ont été retenus. La figure 2.16 contient les informations relatives à cette séquence.

Critères de performances : ϵ_r, ϵ_{C_p}

Les performances de reconstruction sont mesurées sur le coefficient de traînée due à la pression et sur la fidélité du champ dans le voisinage étudié. Suivant l'objectif, il peut être intéressant d'estimer le coefficient de traînée avec précision au détriment du champ

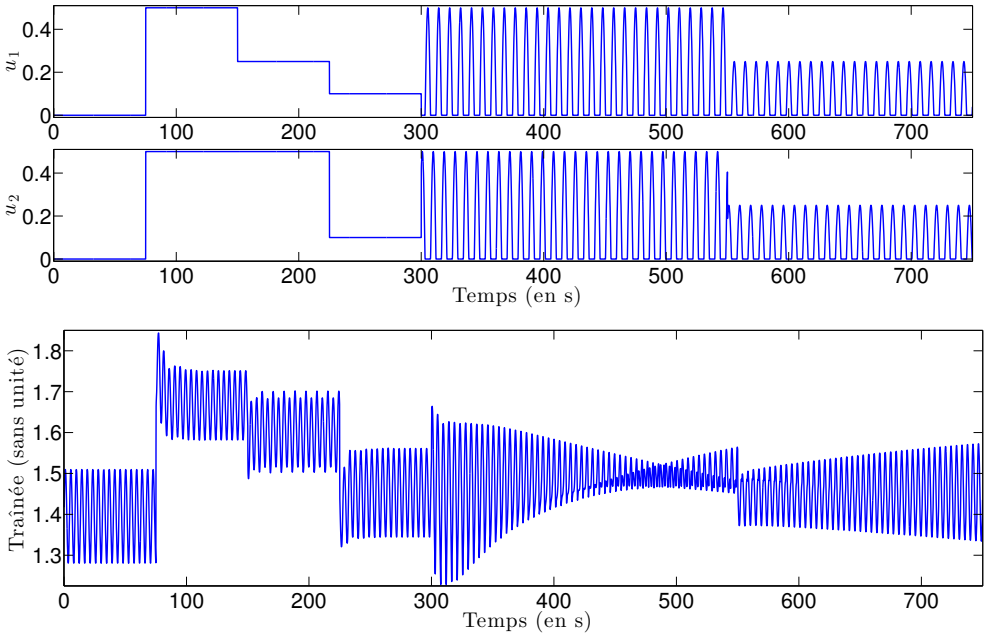

FIGURE 2.16 – Séquence d'apprentissage $\mathcal{S}_1$ - commande (haut) , coefficient de traînée (bas)

de pression total.

Le coefficient de traînée estimée $\hat{\gamma_P}$ est obtenu via une interpolation constante à partir de $\widehat{\boldsymbol{y}}$. Cette interpolation naïve est maintenant largement suffisante puisque 201 points sont disponibles. Cela revient en effet à utiliser 201 capteurs, comparé à l'approche vue en 2.3.1. Le coefficient de traînée γ_P de référence due à la pression est obtenu par la même interpolation mais avec $\boldsymbol{y}$, donnée par le simulateur. Pour un instant donnée i, l'erreur relative faite sur le coefficient de traînée due à la pression est

$$\epsilon_{C_p i} = |\hat{C}_{Pi}/C_{Pi}| \times 100. \tag{2.64}$$

L'erreur moyenne ϵ_{C_p} effectuée sur la totalité de la séquence d'entraînement est ensuite donnée par :

$$\epsilon_{C_p} = \text{mean}\left\{\epsilon_{C_p i}\right\}, \tag{2.65}$$

où mean $\{.\}$ correspond à l'opérateur moyenne, effectuée pour toutes les valeurs possibles de i, c'est-à-dire $i = 1 \ldots n_{\text{snap}}$.

Ce second critère est plus général et concerne la totalité des champs de pression de la séquence d'entraînement. L'erreur relative ϵ_r sur toute la séquence d'entraînement est calculée (voir 2.18). Elle est définie par

$$\epsilon_r = ||Y - \widehat{Y}||_F/||Y||_F \times 100. \tag{2.66}$$

2.4.2 Résultats

Les performances de la méthode POD dépendent des paramètres suivants :
— le nombre de capteurs et leur position : C,
— la séquence d'entraînement : Y, et
— le nombre de modes POD retenus : n_D

Effet des capteurs

Dans cette sous-partie, le nombre de modes POD retenus est égal au nombre de capteurs : $n_D = n_S$. La séquence d'entraînement est la séquence longue.

Les erreurs relatives moyennes ϵ_{C_p} et ϵ_r sont déterminées pour différentes positions des capteurs. Les résultats sont donnés dans les tableaux 2.1 et 2.2.

n_S	$\mathcal{C}_{\text{opt}} - \epsilon_r$	$\mathcal{C}_{\text{opt}} - \epsilon_{C_p}$	$\mathcal{C}_{\text{f}} - \epsilon_r$	$\mathcal{C}_{\text{f}} - \epsilon_{C_p}$
1	64.0%	36.3%	72.1%	10.7%
2	46.6%	60.4%	137%	15.8%
3	20.6%	2.89%	110%	12.8%
4	12.3%	5.86%	103%	12.9%
5	3.68%	1.42%	219%	30.1%

TABLE 2.1 – POD - Performances $\mathcal{C}_{\text{opt}}$ et $\mathcal{C}_{\text{f}}$

n_S	$\mathcal{C}_{\text{u}} - \epsilon_r$	$\mathcal{C}_{\text{u}} - \epsilon_{C_p}$
6	193%	11.9%
7	2.00%	0.173%
8	0.851%	0.252%
9	22.0%	0.310%
10	4.66%	0.350%

TABLE 2.2 – POD - Performances $\mathcal{C}_{\text{u}}$

Tout d'abord, l'impact des capteurs sur les performances de l'estimation est très important, comme prévu. L'estimation du champ total ou du coefficient de traînée dépend du nombre et du positionnement des capteurs. Augmenter le nombre de capteurs utilisés ne garantit pas une meilleure estimation, comme le montre le cas des positions $\mathcal{C}_{\text{f}}$. Il faut les **disposer correctement**.

Ensuite, il n'y a pas de corrélation simple entre les performances ϵ_r et ϵ_{C_p}. Une estimation convenable du champ total ne garantit pas une estimation convenable du champ de pression au voisinage du cylindre. Ceci permet d'insister sur le fait que les capteurs seront placés différemment selon **l'utilisation** qui sera faite des mesures.

Les cases colorées illustrent un comportement notable : celui d'avoir une erreur ϵ_r qui diminue avec le nombre de capteurs. Ceci est en fait une conséquence attendue des positions $\mathcal{C}_{\text{opt}}$, données par SENSORSPACE (en particulier, le critère utilisé 5.16). Elles ont été choisies pour faire le moins d'erreur possible selon ϵ_r et non selon ϵ_{C_p}.

Enfin, l'utilisation de capteurs placés uniformément ne garantit pas non plus une erreur convenable. Cependant, plus le nombre de capteurs augmente, moins leur positionnement est *critique*. Ceci n'est pas absolu, comme le montre le cas $n_S = 9$. Une rotation de la disposition des capteurs peut cependant aider pour ce cas.

Effet de la séquence d'entraînement

Les positions $\mathcal{C}_{\text{opt}}$ des capteurs sont utilisées. Le nombre de modes POD est encore choisi tel que $n_D = n_S$.

Le champ de pression associé aux séquences $\mathcal{S}_{\text{l}}$ ou $\mathcal{S}_{\text{c}}$ sont estimés à partir des bases POD longues Ψ_l ou courtes Ψ_c. Les résultats se trouvent dans le tableau 2.3. De ce fait, la case $\Psi_c \rightarrow \mathcal{S}_{\text{l}}$ signifie que la base POD associée à la séquence courte Ψ_c est utilisée

afin d'estimer le champ de pression associée à la séquence longue $\mathcal{S}_\mathrm{l}$ (à partir des mesures faites sur la séquence longue $\mathcal{S}_\mathrm{l}$).

n_S	$\Psi_l \to \mathcal{S}_\mathrm{l}$	$\Psi_c \to \mathcal{S}_\mathrm{l}$	$\Psi_l \to \mathcal{S}_\mathrm{c}$	$\Psi_c \to \mathcal{S}_\mathrm{c}$
1	64.0%	67.6%	56.6%	56.7%
2	46.6%	44.0%	42.6%	35.11%
3	20.6%	20.8%	18.0%	17.9%
4	12.3%	12.5%	11.4%	11.1%
5	3.68%	3.92%	3.84%	3.60%

TABLE 2.3 – POD - Performances $\mathcal{C}_\mathrm{opt}$- Effet de la séquence d'entraînement

Il est remarquable que les deux bases POD peuvent être utilisées pour effectuer des estimations convenables sur les deux séquences.

Cependant, quelques remarques fondamentales peuvent être faites à partir de cette petite expérience. Les cases bleues (foncées) montrent que la base POD de la séquence à estimer n'est pas forcément celle qui permet le meilleur ϵ_r ! La case verte (claire) montre que la base POD Ψ_l peut causer une *grande* erreur sur ϵ_r, bien plus que celle effectuée pour les autres n_S (comparaison des deux dernières colonnes).

Ceci provient encore du fait que l'information disponible sur le système est réduite et dépend fortement de la **position des capteurs**. Pour ressentir ceci, le cas où l'information totale est disponible est utilisée. Ceci revient à utiliser 201 capteurs. n_D correspond au nombre de modes POD retenu.

n_D	$\Psi_l \to \mathcal{S}_\mathrm{l}$	$\Psi_c \to \mathcal{S}_\mathrm{l}$	$\Psi_l \to \mathcal{S}_\mathrm{c}$	$\Psi_c \to \mathcal{S}_\mathrm{c}$
1	54.2%	55.5%	48.3%	46.5%
2	38.9%	40.5%	35.2%	32.4%
3	20.4%	20.5%	17.8%	17.7%
4	9.52%	9.68%	8.81%	8.62%
5	3.63%	3.87%	3.80%	3.56%

TABLE 2.4 – POD - Performances 201 capteurs - Effet de la séquence d'entraînement

Le tableau 2.4 contient les résultats de cette expérience. Cette fois-ci, la base Ψ_l est bien la plus adaptée à estimer les données $\mathcal{S}_\mathrm{l}$. La même remarque peut être faite pour les données $\mathcal{S}_\mathrm{c}$ à partir de la base Ψ_c. Les deux bases sont capables d'être utilisées sur les deux jeux de données.

Contre l'intuition initiale, la base Ψ_c s'avère relativement plus performante que la base Ψ_l pour effectuer l'estimation sur les deux séquences. Une étude approfondie sur le choix de la séquence d'apprentissage n'a pas été réalisée lors de ce travail de thèse.

Une séquence de longue durée permet d'obtenir une base POD riche au détriment d'un temps de calcul plus long. Elle est qualifiée d'informative puisqu'elle contient *a priori* plus d'états possibles différents du champ étudié. Elle n'est pas garantie d'être optimale selon les données mesurées rencontrées en pratique.

Cette sous-section illustre le fait qu'il faut du recul dans la conception de la séquence d'entraînement. Ceci sera vu plus en détail dans 3.5.

Effet du nombre de modes POD retenus

La séquence d'entraînement retenue est $\mathcal{S}_\mathrm{l}$. Les capteurs utilisés sont disposés sur leurs positions $\mathcal{C}_\mathrm{opt}$. Cette partie à pour objectif d'illustrer l'importance du bon choix du nombre

de modes POD. Le nombre et la position des capteurs sont fixés. Utiliser plus de modes POD que de capteurs semble voué à l'échec puisqu'il y aura plus d'inconnues que d'équations. Le tableau 2.5 permet d'illustrer ce point.

L'erreur d'estimation relative ϵ_r est calculée avec l'approche POD pour différentes valeurs du nombre de modes POD n_D utilisé. Par exemple, le cas $n_S = 4$ et $n_D = 7$ signifie qu'à partir de 4 mesures de pression, il faut estimer les 7 inconnues du vecteur $\boldsymbol{a}$.

n_S	$n_D = 1$	$n_D = 2$	$n_D = 3$	$n_D = 4$	$n_D = 5$	$n_D = 6$	$n_D = 7$
1	64.0%	64.8%	66.4%	66.4%	94.8%	94.8%	99.9%
2	58.1%	46.6%	56.9%	57.6%	59.8%	60.2%	60.3%
3	58.1%	45.7%	20.6%	20.8%	29.9%	30.0%	30.6%
4	58.5%	46.3%	20.7%	12.3%	16.1%	26.3%	26.4%
5	57.8%	50.0%	20.6%	9.84%	3.68%	3.74%	5.18%

TABLE 2.5 – POD - Performances $\mathcal{C}_{\text{opt}}$ - Effet de n_D

La valeur de n_D qui permet d'effectuer la meilleure estimation est bien $n_D = n_S$, comme prévue par la théorie.

La position des capteurs joue un rôle clé dans cette série d'expérience. Si les positions $\mathcal{C}_{\text{f}}$ sont utilisées, la reconstruction est désastreuse peu importe le nombre de modes n_D retenus.

La totalité des résultats obtenus pour les positions $\mathcal{C}_{\text{f}}$ se trouve dans le tableau 2.6. Les cases colorées indiquent l'erreur d'estimation la plus faible pour une position donnée des capteurs. Lorsque les capteurs sont mal placés, le choix du nombre de modes n_D devient plus délicat. Dès l'ajout d'un deuxième mode POD, l'erreur d'estimation augmente. Il est ici très difficile d'estimer un $\widehat{\boldsymbol{a}}$ performant. Bien qu'utiliser $n_D = n_S$ présente le plus de **potentiel**, ce dernier n'est jamais atteint à cause de l'estimation désastreuse de $\widehat{\boldsymbol{a}}$.

n_S	$n_D = 1$	$n_D = 2$	$n_D = 3$	$n_D = 4$	$n_D = 5$	$n_D = 6$	$n_D = 7$
1	72.1%	74.6%	82.9%	82.9%	84.3%	84.3%	84.5%
2	72.2%	138%	97.2%	80.7%	82.6%	83.2%	83.5%
3	72.3%	97.1%	110%	91.9%	80.7%	82.0%	82.5%
4	73.0%	99.2%	112%	103%	91.3%	80.8%	81.4%
5	73.6%	101%	112%	102%	219%	212%	88.8%

TABLE 2.6 – POD - Performances $\mathcal{C}_{\text{f}}$ - Effet de n_D

2.4.3 Limitations

Sensibilité à la position des capteurs

Comme vu dans les paragraphes précédents, cette approche est extrêmement sensible au positionnement des capteurs. Lorsque peu de capteurs sont disponibles, leur placement devient de plus en plus critique. Une très *faible* erreur (norme ℓ_2) effectuée sur les données mesurées peut causer une *grande* erreur sur l'estimation de la représentation réduite.

Positions interdites

Certaines positions de l'espace ne sont pas accessibles, rendant alors le placement de capteurs dans ces zones impossible. L'approche classique de placement de capteurs (voir le chapitre 5) peut proposer des positions non accessibles. Dans ce cadre, la méthode classique peut présenter des performances dégradées et aucune modification *a posteriori* de la base POD est considérée.

Nombre limité de modes

L'approche POD est limitée de part sa méthode d'obtention de la représentation réduite. Elle utilise **autant de modes que le nombre de capteurs** retenus. Le cas non bruité est ici considéré.

L'approche POD considère implicitement que le nombre de modes utilisés sera suffisant et que le positionnement des capteurs est de qualité. Si ces conditions ne sont pas vérifiés, un grand défaut peut être rencontré. La base POD contient n_D modes et est alors noté Ψ_{n_D}. Soit $\boldsymbol{y}$ le champ à estimer. Ce dernier peut s'écrire :

$$\boldsymbol{y} = \Psi_{n_D}\boldsymbol{a} + \delta\boldsymbol{y} \tag{2.67}$$

où $\boldsymbol{a}$ est la représentation réduite de taille n_D de $\boldsymbol{y}$ et $\delta\boldsymbol{y}$ est l'erreur de description faite sur $\boldsymbol{y}$ due au nombre limité de modes utilisés.

La mesure $\boldsymbol{s}$ vérifie $\boldsymbol{s} = C\boldsymbol{y}$. Cette dernière peut encore s'écrire :

$$\boldsymbol{s} = C\left(\Psi_{n_D}\boldsymbol{a} + \delta\boldsymbol{y}\right) = C\Psi_{n_D}\boldsymbol{a} + C\delta\boldsymbol{y} \tag{2.68}$$

En introduisant le terme $\delta\boldsymbol{s} = C\delta\boldsymbol{y}$, l'égalité suivante peut être écrite :

$$\boldsymbol{s} = C\Psi_{n_D}\boldsymbol{a} + \delta\boldsymbol{s} \tag{2.69}$$

L'approche POD considère que la mesure disponible est $C\Psi_{n_D}\boldsymbol{a}$ et non $\boldsymbol{s}$, celle réellement effectuée. Ceci est négligeable dans le cas où $||\delta\boldsymbol{s}||_2 \ll ||C\Psi_{n_D}\boldsymbol{a}||_2$, ce qui a souvent lieu lorsqu'un grand nombre de modes POD est utilisé. Dans le cas présent, le faible nombre de capteurs utilisés fait que le nombre de modes POD sera possiblement trop faible et l'approche d'estimation sera perturbée par le terme $\delta\boldsymbol{s}$.

L'estimée $\widehat{\boldsymbol{a}}$ est donnée par

$$\widehat{\boldsymbol{a}} = \left(C\Psi_{n_D}\right)^{+}\boldsymbol{s} \tag{2.70}$$

Comme $n_D = n_S$, la matrice $C\Psi_{nD}$ est carrée. Si cette dernière est inversible, ce qui est souvent le cas avec un placement correct des capteurs, la relation suivante peut être obtenue :

$$\widehat{\boldsymbol{a}} = \boldsymbol{a} + \left(C\Psi_{n_D}\right)^{+}\delta\boldsymbol{s} \tag{2.71}$$

La matrice $\left(C\Psi_{n_D}\right)^{+}$ peut amplifier l'erreur causée par $\delta\boldsymbol{s}$ sur $\widehat{\boldsymbol{a}}$ et ensuite sur $\widehat{\boldsymbol{y}} = \Psi_{n_D}\widehat{\boldsymbol{a}}$. Le choix de la position des capteurs C est alors essentiel.

Nécessité d'une nouvelle approche

Dans ce contexte *faible nombre de capteurs*, la limitation due au nombre de modes et au positionnement des capteurs sur l'estimation de la représentation réduite fait que les

performances d'estimations globales peuvent être très décevantes.

Une spécificité de la méthode recherchée sera d'exploiter une position donnée des capteurs. Ceci dans le but de s'**adapter** à cette position. Ainsi, une base réduite permettant d'obtenir une représentation réduite la plus accessible possible va être déterminée à partir de cette configuration particulière. Pour une position donnée des capteurs, l'approche conçue se veut être plus performante que l'approche POD, justifiant ainsi son utilité. La base ne sera pas optimale d'un point de vue descriptif comme la POD mais elle permettra une meilleure estimation pratique. En d'autres termes, il peut être intéressant d'utiliser un grand nombre de modes sous-optimaux qu'un faible nombre de modes optimaux dans une situation non réalisable en pratique. Ceci peut se faire au prix d'une étape off-line plus coûteuse. L'étape on-line ne doit pas voir son temps de calcul être *trop* modifié afin d'autoriser un fonctionnement temps réel. Enfin, la méthode devra aussi permettre de placer les capteurs en vue d'une estimation performante.

2.5 Conclusion

Ce chapitre a permis de fixer le cadre technique de ce travail de thèse. Bien que d'autres problèmes seront soulevés dans la suite, le socle de base a été présenté. Les outils principaux de mécanique des fluides ont permis d'appréhender le système de référence qui sera utilisé tout au long de ce travail. Les hypothèses faites sur les données disponibles et les contraintes d'estimation ont été exposées. La méthode la plus communément utilisée faisant intervenir la POD a été détaillée.

Les positions de capteurs $\mathcal{C}_{\text{opt}}$ et $\mathcal{C}_{\text{f}}$ seront principalement utilisées dans la suite du manuscrit. Ceci permettra de comparer les méthodes conçues dans une situation optimale pour la POD et une situation qui l'est bien moins.

La séquence $\mathcal{S}_{\text{l}}$ sera utilisée afin d'illustrer le bon fonctionnement des méthodes lorsque de nombreux *snapshots* sont disponibles.

Le chapitre suivant va permettre de présenter la contribution principale de ce manuscrit : une méthode d'estimation de type machine learning exploitant les outils liés à la parcimonie. Cette dernière va tenter de surmonter les limitations de l'approche POD au prix d'une plus grande complexité de conception.

Bibliographie

[1] Laurent Cordier and Michel Bergmann. Proper Orthogonal Decomposition : An Overview. In *Lecture series 2002-04 and 2003-04 on Post-Processing of Experimental and Numerical Data*. Von Karman Institute for Fluid Dynamics, 2003.

[2] Xiu Yang, Daniele Venturi, Changsheng Chen, Chryssostomos Chryssostomidis, and George Em Karniadakis. EOF-based constrained sensor placement and field reconstruction from noisy ocean measurements : Application to Nantucket Sound. *Journal of Geophysical Research*, 115(C12) :C12072, December 2010.

[3] Ido Bright, Guang Lin, and J Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(12) :127102, 2013.

[4] Xiao-Hua Zhang, You-Lin Xu, Songye Zhu, and Sheng Zhan. Dual-type sensor placement for multi-scale response reconstruction. *Mechatronics*, 24(4) :376–384, June 2014.

[5] Leehter Yao, William A. Sethares, and Daniel C. Kammer. Sensor placement for on-orbit modal identification via a genetic algorithm. *AIAA Journal*, 31(10) :1922–1928, October 1993.

[6] M. Meo and G. Zumpano. On the optimal sensor placement techniques for a bridge structure. *Engineering Structures*, 27(10) :1488–1497, August 2005.

[7] Bernd R. Noack, Konstantin Afanasiev, Marek MorzyŃdki, Gilead Tadmor, and Frank Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497 :335–363, December 2003.

[8] Jean-Marie Brébec, Michel Briffaut, and Philippe Denève. *Hprépa - Mécanique des fluides 2ème année PC-PC*, PSI-PSI**. Hachette Supérieur, 2004.

[9] Lionel Mathelin and Olivier P. Le Maître. Robust control of uncertain cylinder wake flows based on robust reduced order models. *Computers & Fluids*, 38(6) :1168–1182, June 2009.

[10] Johannes Gerhard, Mark Pastoor, Rudibert King, Bernd R. Noack, Andreas Dillmann, Marek MorzyŃdki, and Gilead Tadmor. Model-based control of vortex shedding using low-dimensional Galerkin models. In *33rd AIAA Fluids Conference and Exhibit*, pages 2003–4262, Orlando, Florida, USA, 2003.

[11] Onofrio Semeraro, Shervin Bagheri, Luca Brandt, and Dan S. Henningson. Transition delay in a boundary layer flow using active control. *Journal of Fluid Mechanics*, 731 :288–311, August 2013.

[12] Nicolas Gautier and Jean-Luc Aider. Feed-forward control of a perturbed backward-facing step flow. *Journal of Fluid Mechanics*, 759 :181–196, October 2014.

[13] John R. Roth, Daniel M. Sherman, and Stephen P. Wilkinson. Electrohydrodynamic Flow Control with a Glow-Discharge Surface Plasma. *AIAA Journal*, 38(7) :1166–1172, July 2000.

[14] C H K Williamson. Vortex dynamics in the cylinder wake. *Annu. Rev. Fluid Mech.*, 28 :477–539, 1996.

[15] Olivier P. Le Maitre, Robert H. Scanlan, and Omar M. Knio. Estimation of the flutter derivatives of an NACA airfoil by means of Navier–Stokes simulation. *Journal of Fluids and Structures*, 17(1) :1–28, January 2003.

[16] Mohammed Rizi. *Contrôle de la couche de cisaillement d'un écoulement de cavité.* PhD thesis, Ecole Normale Supérieure de Cachan, 2015.

[17] Clarence W. Rowley. Model Reduction For Fluids Using Balanced Proper Orthogonal Decomposition. *International Journal of Bifurcation and Chaos*, 15(03) :997–1013, March 2005.

[18] Mohammad Samimy, Marco Debiasi, Edgar Caraballo, A. Serrani, X. Yuan, J. Little, and J.H. Myatt. Feedback control of subsonic cavity flows using reduced-order models. *Journal of Fluid Mechanics*, 579 :315, May 2007.

[19] K Willcox and J Peraire. Balanced Model Reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, 40(11) :2323—-2330, 2002.

[20] Michel Bergmann and Laurent Cordier. Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models. *Journal of Computational Physics*, 227(16) :7813–7840, August 2008.

[21] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. Part I : coherent structures. *Quaterly of Applied Mathematics*, 45(3) :571–591, 1987.

[22] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hokpinks University Press, Baltimore, MD, 3rd editio edition, 1996.

[23] Medhat a. Rakha. On the Moore–Penrose generalized inverse matrix. *Applied Mathematics and Computation*, 158(1) :185–200, October 2004.

CHAPITRE 3
Exploitation de la Parcimonie

Chapitre 3

Exploitation de la Parcimonie

Contenu

Ce chapitre a pour but de présenter la contribution principale de ce travail de thèse : une méthode exploitant la parcimonie et la position des capteurs afin d'estimer les valeurs d'un champ de grande dimension à partir d'un faible nombre de mesures effectuées sur ce dernier.

Tout d'abord, la notion de représentation creuse va être illustrée. Ceci sera fait en présentant le **Compressed Sensing** qui exploite principalement de telles représentations.

Ensuite, à partir des outils exposés, une première stratégie d'estimation sera proposée. La compatibilité de cette approche avec les bases réduites usuelles (dont la POD) sera étudiée. Les limitations (2.4.3) d'une telle démarche seront surmontées grâce à l'introduction de la position des capteurs dans une méthode de création de dictionnaire spécifique, issue de la communauté Compressed Sensing. Les performances d'une telle méthode seront explicitées.

Ce chapitre se termine sur une généralisation de la méthode aux cas où les mesures ne sont plus une restriction du champ à estimer.

3.1 Les représentations creuses

Comme cela a été vu dans le chapitre précédent, le problème principal à résoudre dans le cadre de l'approche POD est le suivant :

$$\widehat{\boldsymbol{a}} = \underset{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_D}}{\arg\min} ||\boldsymbol{s} - C\Psi\widetilde{\boldsymbol{a}}||_2, \tag{3.1}$$

où Ψ est la base POD d'ordre n_D. Il y a n_S équations (une par capteur) et n_D inconnues (le nombre d'éléments dans $\widehat{\boldsymbol{a}}$). Pour garantir un problème bien posé et permettre la meilleure estimation possible, il faut au plus choisir autant de vecteurs de bases que de nombre de capteurs.

L'idée principale pour obtenir des meilleurs résultats que l'approche POD est d'**utiliser un plus grand nombre de modes**. Cela revient à vérifier la condition $n_D > n_S$. Ainsi, il va exister des $\widehat{\boldsymbol{a}}$ permettant d'obtenir de meilleurs résultats que le cas $n_D = n_S$. L'obstacle qui doit maintenant être franchi consiste à obtenir un tel $\widehat{\boldsymbol{a}}$. Le problème classique devient mal posé puisque le nombre de capteurs n'a pas augmenté. Il n'y a plus une solution $\widehat{\boldsymbol{a}}$ unique. Ensuite, le problème de sensibilité présentée au chapitre précédent (voir la partie 2.4.3) est encore d'actualité. Une telle approche peut paraître extrêmement optimiste mais elle est utilisée avec succès dans le domaine du Compressed Sensing à travers les représentations creuses.

Le Compressed Sensing (CS) est un domaine émergent en traitement du signal qui a gagné en popularité de par ses applications en acquisition de données. L'échantillonnage d'un signal d'intérêt, ici représenté par un vecteur, dans le but de le stocker sous une forme compacte est considéré. La théorie de Shannon [1] montre qu'un échantillonnage uniforme à la fréquence de Nyquist, définie comme étant le double de la plus grande fréquence utile présente dans le signal d'intérêt, permet une reconstruction future du même signal sans perte d'information. Ainsi, un signal de grande dimension peut être conservé sous la forme d'une collection d'échantillons de taille plus faible (directement comprimée). Par contre, de nombreuses applications font face à des contraintes d'acquisition et de stockage puisque la fréquence de Nyquist est trop élevée. L'idée principale du CS est de combiner les étapes d'échantillonnage et de compression. La méthode vise ainsi directement l'acquisition sous une forme comprimée. Elle repose sur le fait que de nombreux signaux peuvent s'exprimer de manière très concise dans des bases adaptées. Un tel type de représentation est la **représentation creuse** où le vecteur d'intérêt est décrit pas un très faible nombre de termes non-nuls. Le CS ne se limite pas qu'à cet aspect et traite aussi de l'échantillonnage non uniforme, des algorithmes de conception de telles bases et de la mesure de leurs performances. Pour une introduction complète à la théorie du CS, les lectures suivantes sont conseillées [2, 3, 4, 5].

Soit $\boldsymbol{y} \in \mathbb{R}^{n_x}$ le vecteur d'intérêt. Ce dernier possède la représentation $\boldsymbol{a} \in \mathbb{R}^{n_D}$ dans la base $\Phi \in \mathbb{R}^{n_x \times n_D}$, aussi qualifiée de dictionnaire par la communauté, et vérifie $\boldsymbol{y} = \Phi\boldsymbol{a}$. Dire que $\boldsymbol{y} \in \mathbb{R}^{n_x}$ possède une représentation creuse signifie qu'il existe un vecteur $\boldsymbol{a}$ vérifiant l'égalité ci-dessus et dont certaines composantes sont nulles. Le nombre d'éléments non-nuls est $K \in \mathbb{N}^*, K \leq n_D$, et correspond au **degré de parcimonie** de la représentation. Enfin, pour introduire le vocabulaire communément utilisé, un tel vecteur est dit K-*sparse*. Par abus de langage, un vecteur possédant des composantes de valeur absolue négligeable devant d'autres composantes, dites dominantes, peuvent aussi être qualifiés de *sparse*. Enfin, $\boldsymbol{y}$ est aussi qualifié de compressible dans la base Φ si l'inégalité $||\boldsymbol{y} - \Phi\boldsymbol{a}||_2 \leq \epsilon$ est vérifiée avec $\boldsymbol{a}$ *sparse* et ϵ petit suivant l'application (par exemple, $\epsilon =$

$10^{-2}||\boldsymbol{y}||_2$). Cela signifie qu'un vecteur creux est capable de représenter convenablement le vecteur d'intérêt $\boldsymbol{y}$.

Ainsi, le vecteur $\boldsymbol{y}$ est associé à $\boldsymbol{a}$ qui ne possède que K termes non-nuls. $\boldsymbol{y}$ est effectivement décrit par K vecteurs de base au lieu de n_D. En choisissant $n_D > n_S$ et $K = n_S$, le problème est presque redevenu bien posé. En effet, il y a plus que K inconnues puisqu'il faut aussi déterminer la position des termes non nuls de $\boldsymbol{a}$. La véritable force d'une telle description est apparente lorsqu'une séquence de vecteurs est étudiée. Chaque $\boldsymbol{y}_i$ n'utilisera que K vecteurs de base, mais ces vecteurs peuvent être différents à chaque instant i. Ainsi, un dictionnaire adapté à une telle utilisation devra posséder une certaine redondance. Il peut être intéressant d'y stocker différents modes *proches* afin que le meilleur soit utilisé dans la bonne situation. Voici la raison pour laquelle utiliser une représentation creuse ne revient pas à seulement tronquer le dictionnaire.

Il faut maintenant être capable d'obtenir une telle représentation creuse. Ceci se fait à l'aide d'une base Φ qui autorise de telles représentations et à l'aide d'algorithmes spécifiques. Un système d'équations linéaires mal-posé est devenu bien-posé à l'aide de l'ajout d'une contrainte de nature non-linéaire sur la forme de la solution.

Les paragraphes suivants vont présenter les outils les plus communément utilisés pour obtenir des représentations creuses et pour quantifier la capacité d'une base quant à la promotion de ces dernières.

3.1.1 Outils

Cette sous-section a pour but de présenter les outils principaux utilisés en CS. Ces derniers sont utilisés tout au long du manuscrit et sont nécessaires pour décrire et quantifier le fonctionnement des algorithmes conçus.

"Norme 0"

L'outil le plus commode utilisé dans le domaine du Compressed Sensing est la "norme 0", noté ℓ_0. La "norme 0" d'un vecteur $\boldsymbol{a}$ *K-sparse* est $||\boldsymbol{a}||_0 = K$. Elle renvoie le nombre d'éléments non-nul. Il faut ajouter que cette norme n'en est pas une au sens strict du terme. Elle ne vérifie pas la propriété d'homogénéité, c'est-à-dire :

$$\forall(\lambda, \boldsymbol{a}) \in \mathbb{R} \times \mathbb{R}^{n_D}, ||\lambda \boldsymbol{a}|| = |\lambda|.||\boldsymbol{a}||, \tag{3.2}$$

où $||.||$ est une norme quelconque définie sur un espace euclidien. Cela dit, **par abus**, cet opérateur $||.||_0$ est quand même appelée la norme ℓ_0.

Une autre manière de définir cette norme est via la notion de *support* et de *cardinal*. L'opérateur support, $\text{supp}\,(.)$, renvoie l'ensemble des index des composantes non-nulles d'un vecteur. Par exemple, en introduisant le vecteur $\boldsymbol{w} = (0\ 3\ 0\ 9\ 6)^T$, son support est $T = \text{supp}\,(\boldsymbol{w}) = \{2\ 4\ 5\}$. L'opérateur cardinal, $\text{card}\,(.)$, renvoie le nombre d'éléments dans un ensemble. En reprenant le vecteur exemple ci-dessus, $\text{card}\,(T) = \text{card}\,(\text{supp}\,(\boldsymbol{w})) = 3$. On a bien

$$||\boldsymbol{w}||_0 = \text{card}\,(\text{supp}\,(\boldsymbol{w})) = 3 \tag{3.3}$$

Avec cet outil, l'obtention d'une représentation creuse se formalise selon :

$$\widehat{\boldsymbol{a}} = \underset{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_D}, ||\widetilde{\boldsymbol{a}}||_0 \leq K}{\arg\min} ||\boldsymbol{y} - \Phi \widetilde{\boldsymbol{a}}||_2, \tag{3.4}$$

où la base Φ reste à déterminer.

Cependant, comme seule l'information du champ de pression au niveau des capteurs est disponible, le problème peut maintenant s'écrire :

$$\widehat{\boldsymbol{a}} = \underset{\widetilde{\boldsymbol{a}}\in\mathbb{R}^{n_D}, ||\widetilde{\boldsymbol{a}}||_0\leq K}{\arg\min} ||\boldsymbol{s} - C\Phi\widetilde{\boldsymbol{a}}||_2, \tag{3.5}$$

où, tout comme au chapitre précédent, C indique la position des capteurs et $\boldsymbol{s} = C\boldsymbol{y}$.

Une autre formulation très répandue de ce problème est :

$$\widehat{\boldsymbol{a}} \in \underset{\widetilde{\boldsymbol{a}}\in\mathbb{R}^{n_D}}{\arg\min} \|\boldsymbol{s} - C\,\Phi\,\widetilde{\boldsymbol{a}}\|_2 \quad \text{tel que} \quad \|\widehat{\boldsymbol{a}}\|_0 \leq K. \tag{3.6}$$

Le pendant de cette formulation est le suivant :

$$\widehat{\boldsymbol{a}} \in \underset{\widetilde{\boldsymbol{a}}\in\mathbb{R}^{n_D}}{\arg\min} \|\widetilde{\boldsymbol{a}}\|_0 \quad \text{tel que} \quad \|\boldsymbol{s} - C\,\Phi\,\widehat{\boldsymbol{a}}\|_2 \leq \epsilon, \tag{3.7}$$

où $\epsilon > 0$ est l'erreur maximale autorisée sur la reconstruction des mesures. La résolution de ce problème dépend principalement de la matrice $C\Phi$. Dans le cadre considéré, la matrice Φ peut être créée de toute pièce. Il faut donc trouver une matrice Φ qui permette à la fois d'obtenir une représentation creuse $\boldsymbol{a}$ et d'estimer correctement $\boldsymbol{y}$.

Restrictive Isometry Property

Afin de ne pas alourdir les notations, le problème 3.4 est retenu. Un critère permettant de garantir l'existence d'une solution creuse au problème est donné par la **Restrictive Isometry Property**, appelée plus souvent RIP.

Soit T un sous-ensemble de $\{1, \ldots, n_D\}$. La restriction de Φ aux colonnes indexées par T est notée Φ_T.

La constante K-RIP, notée δ_K, est définie comme le plus petit scalaire tel que la relation ci-dessous soit vérifiée, pour tout sous-ensemble T (tel que $\text{card}\,(T) \leq K$) et pour tout vecteur $\boldsymbol{w} \in \mathbb{R}^{\text{card}(T)}$:

$$(1 - \delta_K)\ ||\boldsymbol{w}||_2^2 \leq ||\Phi_T\boldsymbol{w}||_2^2 \leq (1 + \delta_K)\ ||\boldsymbol{w}||_2^2. \tag{3.8}$$

Une telle constante δ_K permet de mesurer la capacité d'une restriction Φ_T à se **comporter comme une base orthonormale**.

Si $\delta_K < 0.307$, alors la solution $\widehat{\boldsymbol{a}}$ *K-sparse* obtenue à partir de $\boldsymbol{y}$ et Φ (3.4) vérifie [6, 7] :

$$\|\boldsymbol{a} - \widehat{\boldsymbol{a}}\|_2 \leq \frac{1}{0.307 - \delta_K}\left(\epsilon + \frac{\|\boldsymbol{a} - \boldsymbol{a}_{\max_K}\|_1}{\sqrt{K}}\right), \tag{3.9}$$

où

- $\boldsymbol{a}$ est la solution recherchée (mais inconnue)
- ϵ vérifie : $||\boldsymbol{y} - \Phi\boldsymbol{a}||_2 \leq \epsilon$ et permet de modéliser un niveau de bruit de mesure maximal
- $\boldsymbol{a}_{\max_K}$ correspond au vecteur $\boldsymbol{a}$ où seuls les K plus grands éléments (en valeur absolue) sont retenus (les autres composantes sont mises à zéro).

Plus précisément, la relation vérifiée par cette erreur est :

$$\|\boldsymbol{a} - \widehat{\boldsymbol{a}}\|_2 \leq \frac{2\sqrt{2}\sqrt{1+\delta_K}}{1 - C_0\,\delta_K}\left(\epsilon + \frac{\|\boldsymbol{a} - \boldsymbol{a}_{\max_K}\|_1}{\sqrt{K}}\right), \tag{3.10}$$

avec $C_0 = 1 + \frac{23}{2\sqrt{26}}$.

Ce résultat montre que l'erreur effectuée sur la représentation creuse recherchée est proportionnelle à ϵ et au résidu $\|\boldsymbol{a} - \boldsymbol{a}_{\max_K}\|_1$.

Cohérence

Un autre critère plus facile à mettre en œuvre est présenté ci-dessous. La capacité de la base Φ à autoriser une représentation creuse peut être mesurée à l'aide de la notion de cohérence d'une matrice. La cohérence d'une matrice Φ est le scalaire correspondant au plus grand terme de la matrice de Gram $\Phi^T\Phi$, non situé sur la diagonale, associée à la matrice Φ rendue unitaire [8]. Soit $\mu(\Phi)$ la cohérence, elle est définie par

$$\mu(\Phi) = \max_{1 \leq i \neq j \leq n_D} \frac{\left|\boldsymbol{\phi}_i^T \boldsymbol{\phi}_j\right|}{\|\boldsymbol{\phi}_i\|_2 \left\|\boldsymbol{\phi}_j\right\|_2}, \tag{3.11}$$

où $\boldsymbol{\phi}_i$ correspond à la $i^{\text{ième}}$ colonne de Φ.

La cohérence d'une matrice est en fait une *mesure de son orthogonalité*. Plus une base est orthogonale, plus sa cohérence est proche de 0. Plus d'informations sur la cohérence d'une matrice peuvent être trouvées dans [9] et un développement plus complet se situe dans [10].

Si un système linéaire $\boldsymbol{y} = \Phi\widetilde{\boldsymbol{a}}$ possède une solution $\boldsymbol{a}$ K-*sparse* qui vérifie la condition ci-dessous, alors cette dernière est une solution **la plus *sparse* possible** [9] :

$$K \leq \frac{1 + \mu(\Phi)}{2\,\mu(\Phi)}. \tag{3.12}$$

La cohérence n'est pas directement exploitable pour le problème considéré.

En prenant comme exemple l'approche POD, la matrice qui serait exploitée pour obtenir une représentation creuse est $C\Phi$ où Φ est la base POD d'ordre n_S, le nombre de capteurs présents dans C. Avec 3 capteurs placés avec SensorSpace (voir le chapitre 5, la cohérence de la matrice est : $\mu(C\Phi) = 0.713$.

Selon le critère précédent, $K \leq \frac{1+\mu(\Phi)}{2\,\mu(\Phi)} = 1.2$. S'il existe une solution $\boldsymbol{a}$ 1-*sparse* au problème

$$\boldsymbol{s} = C\Phi\widetilde{\boldsymbol{a}},$$

cette dernière sera la plus *sparse* possible.

Ces solutions *très sparse*, si elles existent, ne sont pas forcément celles qui permettront une estimation de qualité du vecteur $\boldsymbol{y}$. Pour l'exemple considéré, des solutions 3-*sparse* seront recherchées. De ce fait, **la cohérence ne sera pas extensivement utilisée** dans le suite du manuscrit.

3.1.2 Résolution

L'objectif est ici de résoudre une équation de la forme :

$$\boldsymbol{a} \in \underset{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_D}}{\arg\min} ||\boldsymbol{y} - \Phi\,\widetilde{\boldsymbol{a}}||_2 \quad \text{tel que} \quad ||\widetilde{\boldsymbol{a}}||_0 \leq K, \tag{3.13}$$

où Φ est un dictionnaire quelconque. Φ et $\boldsymbol{y}$ sont connus et il faut obtenir une représentation creuse $\boldsymbol{a}$ K-*sparse* qui permet de minimiser la norme 2 ci-dessus 3.13.

La contrainte de parcimonie sur $\boldsymbol{a}$ via la norme ℓ_0 impose que la résolution ne puisse pas se faire par des méthodes linéaires classiques [8]. Le problème est de nature combinatoire de par le choix du support de la solution. Ce dernier est qualifié de *NP-hard* et une solution exacte est souvent hors de portée à cause de la grande taille de l'espace solution. Cependant, une fois un support pour $\boldsymbol{a}$ choisi, la valeur des composantes de ce vecteur peut être déterminée par des méthodes plus classiques puisque le critère à minimiser est

du type $||\boldsymbol{y} - A\boldsymbol{x}||_2$ avec $\boldsymbol{x}$ le vecteur inconnu. Ce critère devient en effet convexe et une solution globale peut être obtenue. Les paragraphes suivants ont pour objectif de présenter différentes approches pour résoudre ce problème 3.13.

Comme il peut être vu dans [11], il existe 3 grandes catégories de méthodes pour résoudre ce problème :

— les algorithmes de nature combinatoire,
— les algorithmes de minimisation de normes ℓ_1,
— les algorithmes *greedy*.

Les algorithmes de nature **combinatoire** cherchent à effectuer le moins de tests possibles pour trouver le support du vecteur creux ainsi que la valeur de ces composantes. Les algorithmes Count-Min Sketch et Count-Median Sketch font partie de cette famille. Cependant, ces algorithmes nécessitent souvent *un contrôle total sur la matrice* Φ. Ceci n'est pas le cas ici. De ce fait, ce type d'algorithme ne sera pas utilisé dans la suite.

L'origine de la complexité de ce problème d'estimation réside dans la pseudo-norme ℓ_0. Le problème d'optimisation peut être simplifié à l'aide d'une relaxation sur cette "norme". **En la remplaçant par la norme** ℓ_1, le problème *perd sa nature combinatoire* et des méthodes classiques peuvent être appliquées. Le problème peut être vu comme la minimisation d'une quantité convexe soumise à une contrainte conique. Cependant, une modification supplémentaire consistant à intégrer la contrainte dans le terme principal est très souvent réalisée. Une telle approche fait intervenir un Lagrangien et le problème devient :

$$\widehat{\boldsymbol{a}} = \arg\min_{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_D}} \|\boldsymbol{y} - \Phi\, \widetilde{\boldsymbol{a}}\|_2 + \tau\ \|\widetilde{\boldsymbol{a}}\|_1 \,, \tag{3.14}$$

avec $\tau > 0$ un coefficient permettant de régler le compromis qui est fait entre promouvoir la parcimonie de $\widehat{\boldsymbol{a}}$ et faire le moins d'erreur possible sur la description de $\boldsymbol{y}$. Modifier la valeur de τ permet d'explorer le front de Pareto de la solution. L'algorithme LASSO [12] fait partie des plus connus de cette catégorie. Il faut remarquer que ces algorithmes ne produisent pas une solution creuse sans étape de troncature supplémentaire. Malheureusement, suivant les cas (matrice Φ, vecteur $\boldsymbol{y}$), la troncature effectuée sur $\widehat{\boldsymbol{a}}$ (en ne conservant que les K plus grand termes en valeur absolue) peut avoir des conséquences néfastes sur l'approximation de $\boldsymbol{y}$. Ce type d'algorithme ne sera donc pas utilisé dans la suite.

Les ***greedy algorithms*** sont des algorithmes qui vont sacrifier l'exactitude de la solution afin de gagner en rapidité, satisfaisant ainsi au besoin de pouvoir fonctionner en temps réel. La majorité utilise une succession d'approximations locales agissant sur le support de la solution. A chaque étape de l'algorithme, la meilleure décision locale est prise. Un tel algorithme est ainsi qualifié de *greedy*, puisqu'il en est attendu une solution de nature globale malgré la nature locale des étapes. Certains *greedy algorithms* sont capables de justifier de garanties similaires à celles des algorithmes de type ℓ_1.

Parmi les algorithmes de type *greedy*, deux algorithmes seront examinés : **Iterative Hard Thresholding** (IHT) [13] et **Orthogonal Matching Pursuit** (OMP) [9].

L'algorithme IHT consiste en une modification de l'algorithme classique de descente à l'aide du gradient. La solution courante est tronquée à la fin de chaque itération afin de ne garder que les K plus grands termes lorsqu'une solution K-*sparse* est recherchée.

L'algorithme OMP cherche le meilleur support possible via une approche *greedy*. Ce dernier ajoute un élément au support à chaque itération. Le paragraphe suivant contient

plus d'informations sur cette approche puisqu'elle est grandement utilisée dans la suite. Ces deux algorithmes manipulent naturellement des solutions *sparse* contrairement aux approches ℓ_1.

OMP

OMP est l'algorithme principal retenu pour résoudre le problème 3.4. Pour illustrer plus facilement OMP, le problème d'obtenir une représentation K-*sparse* nommée $\boldsymbol{a}$ de $\boldsymbol{y}$ à travers le dictionnaire Φ (de n_D éléments) est considéré.

Les choix *motivant l'utilisation d'OMP* sont les suivants. Tout d'abord, ce dernier manipule une solution $\boldsymbol{a}$ creuse sans nécessiter de troncature, permettant ainsi de ne pas introduire une erreur supplémentaire sur l'approximation $\Phi\boldsymbol{a}$ du vecteur $\boldsymbol{y}$ (contrairement à l'algorithme IHT). Le problème de sensibilité (2.4.3) est encore prépondérant et il vaut mieux essayer de s'en prémunir le plus possible en manipulant une solution réellement K-*sparse*. Ensuite, OMP utilise un nombre constant d'itérations et permet ainsi d'avoir un temps de calcul contrôlé. Il est parfaitement **adapté à une utilisation en temps réel**.

Soit T_{OMP} l'ensemble qui contient les indices du support de $\boldsymbol{a}$. La restriction de $\boldsymbol{a}$ aux éléments indexés par T_{OMP} est notée $\boldsymbol{a}_{T_{\text{OMP}}}$ et la restriction de Φ aux colonnes indexées par T_{OMP} est notée $\Phi_{T_{\text{OMP}}}$. L'état courant de $\boldsymbol{a}_{T_{\text{OMP}}}$ est le vecteur qui minimise la norme $||\boldsymbol{y} - \Phi_{T_{\text{OMP}}}\boldsymbol{a}_{T_{\text{OMP}}}||_2$. Lors de chaque itération de l'algorithme, un élément supplémentaire est ajouté au support T_{OMP} jusqu'à ce que ce dernier contienne K éléments. L'index d'un tel élément est choisi comme étant celui de la colonne de Φ la plus corrélée avec le résidu $\boldsymbol{r} \in \mathbb{R}^{n_D}$. Ce dernier est l'erreur effectuée sur la description de $\boldsymbol{y}$ et est défini par $\boldsymbol{r} = \boldsymbol{y} - \Phi\boldsymbol{a}$ où $\boldsymbol{a}$ est l'état actuel du vecteur creux recherché. L'initialisation se fait avec $\boldsymbol{a} = \boldsymbol{0}$ et le résidu initial est ainsi $\boldsymbol{r} = \boldsymbol{y}$.

Le pseudo-code de l'algorithme OMP se trouve dans l'algorithme 1.

Algorithm 1 OMP Pseudo-Code

Require: Φ, $\boldsymbol{y}$, K

1: **Initialisation** :
2: $n_D \leftarrow$ nombre de modes de Φ
3: $T_{\text{OMP}} \leftarrow$ ensemble vide qui va contenir les index du support de la représentation creuse
4: $\boldsymbol{a} \leftarrow \boldsymbol{0}$
5: $\boldsymbol{r} \leftarrow \boldsymbol{y}$
6:
7: **for** $j \in [1 \dots K]$ **do**
8: **Choix d'un élément du support** :
9: **for** $i \in [1 \dots n_D]$ **do**
10: Trouver le mode l qui maximise $\frac{\boldsymbol{\phi}_l \cdot \boldsymbol{r}}{\boldsymbol{\phi}_l \cdot \boldsymbol{\phi}_l}$
11: $T_{\text{OMP}} \leftarrow \{T_{\text{OMP}}, l\}$
12: **Mise à jour** :
13: $\boldsymbol{a}_{T_{\text{OMP}}} \leftarrow (\Phi_{T_{\text{OMP}}})^+ \boldsymbol{y}$
14: $\boldsymbol{r} \leftarrow \boldsymbol{y} - \Phi\boldsymbol{a}$
15:
16: **Sortie** : $\boldsymbol{a}$

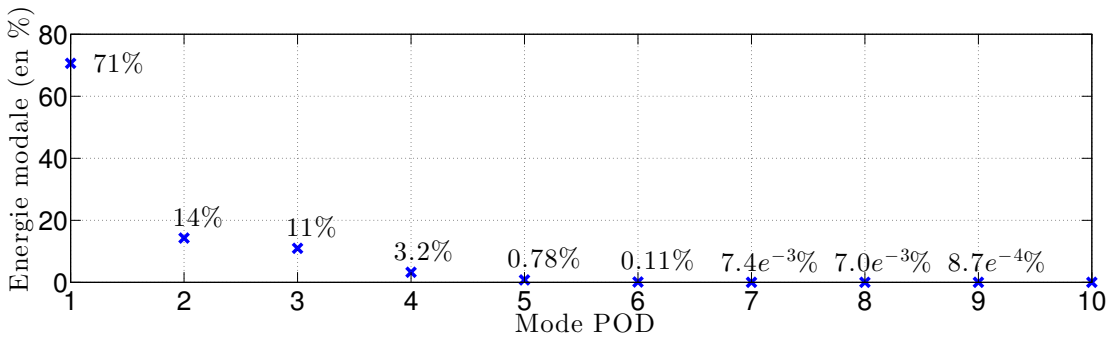

FIGURE 3.1 – POD - Répartition de l'énergie modale

3.1.3 Compatibilité des bases usuelles

Le problème qu'il faut ensuite résoudre est l'**obtention** d'un dictionnaire (d'une "base") Φ. La stratégie adoptée a été de tester quelques bases usuelles afin de savoir si elles pouvaient convenir, allégeant ainsi le développement de la méthode. Certaines bases classiques permettent d'effectuer une estimation très satisfaisante de $\boldsymbol{a}$ mais pas forcément en autorisant des solutions $\boldsymbol{a}$ creuses. S'il s'avère que ces bases ne sont pas satisfaisantes, un dictionnaire devra être créé pour cette application spécifique.

Il ne faut pas oublier l'objectif d'améliorer les performances d'estimation en augmentant le nombre de modes (vecteurs de base) disponibles tout en maintenant un problème soluble à partir du nombre très limité de mesures.

Les bases qui vont être étudiées sont les bases POD, DMD et BPOD. Chaque base va être quantifiée par l'énergie que peut capter chacun de ses modes ainsi que par des exemples de représentations creuses associées dans le cas où le champ total est disponible. Par souci de clarté, les capteurs ne sont pas introduits à ce stade. Cela signifie que la totalité du champ de pression est disponible. Les représentations creuses sont obtenues à partir de l'algorithme OMP.

Base POD

La base POD est utilisée par de nombreuses communautés [14, 15, 16]. Sa propriété d'**orthogonalité** permet d'obtenir des représentations par simple projection orthogonale du vecteur à réduire sur cette même base. Elle présente aussi des modes efficaces dits d'énergies décroissantes. Cette dernière est liée à la valeur singulière du mode associé de Y. Cette propriété est utilisée en pratique afin de savoir combien de modes doivent être retenus afin que l'erreur de troncature soit acceptable.

En considérant la séquence d'apprentissage $\mathcal{S}_{\mathrm{l}}$, l'énergie qui peut être capturée par chaque mode POD est représentée sur la figure 3.1. Cette dernière est obtenue à partir des valeurs singulières de Y : $\sigma_i \in \mathbb{R}_+$ où i indique le numéro du mode POD. Ainsi, le mode i contient $\sigma_i^2 / \sum_{i=1}^{n_{\mathrm{snap}}} \sigma_i^2 \times 100$ % de l'énergie de Y.

Pour atteindre ce genre de performance en pratique, il faudrait pouvoir accéder au champ total (et non seulement au champ de pression dans le voisinage du cylindre).

Chaque mode ajouté permet de *réduire l'erreur* de description mais le fait avec un effet de plus en plus *estompé*. Ce point nous indique qu'il va être très **difficile** d'obtenir une

	$n_S = 1$	$n_S = 2$	$n_S = 3$	$n_S = 4$	$n_S = 5$
cas 1	64.0%	46.6%	20.6%	12.3%	3.68%
cas 2	64.0%	76.3%	299%	14.6%	3.03 10^3 %
cas 3	64.0%	106%	367%	186%	2.94 10^3 %

TABLE 3.1 – OMP - Performances $\mathcal{C}_{\text{opt}}$ - utilisation de la base POD

représentation creuse satisfaisante.

Pour ressentir ceci, une base POD de taille 5 est utilisée et $n_S = 5$ mesures de pression sont disponibles. Il est ici supposé naïvement que les n_S termes des représentations réduites peuvent être estimés parfaitement. Pour augmenter les performances d'estimation avec une approche creuse, il faut d'abord **augmenter la dimension de l'espace réduit** $n_D \in \mathbb{N}^*$. La base POD contient ainsi $n_D > 5$ modes. Cependant, seulement $n_S = 5$ modes peuvent être utilisés à la fois afin de maintenir un problème soluble (solution 5-*sparse*). Vu la décroissance des valeurs singulières, les $n_S = 5$ modes les plus efficaces sont les 5 *premiers modes* de la base POD. Ainsi, les représentations seront creuses mais seuls les 5 premiers termes seront non nuls, rendant ainsi **inutile l'utilisation d'une solution *sparse***.

En pratique, l'estimation de la représentation réduite n'est pas parfaite. L'erreur ainsi produite peut faire que l'utilisation d'autres modes que les 5 premiers devienne une solution acceptable. Cependant, vu la décroissance notable de l'énergie des modes POD dans le cas parfait, les 3 premiers modes (par exemple) seront toujours utilisés et l'approche parcimonieuse utilisera 2 modes POD parmi les modes 4 à 7 par exemple.

Afin de tester expérimentalement la compatibilité de la base POD, l'algorithme OMP est utilisé afin de déterminer une représentation creuse. Les erreurs de reconstruction relatives ϵ_r sont situées dans le tableau 3.1. Les positions de capteurs $\mathcal{C}_{\text{opt}}$ sont utilisées. Pour chaque position considérée, 3 cas sont traités :

— cas 1 : la représentation recherchée est pleine et la matrice utilisée pour l'obtenir est $C\Psi_{n_S}$, où Ψ_{n_S} est la base POD d'ordre n_S
— cas 2 : la représentation creuse recherchée est n_S-*sparse* de taille n_S+1 et la matrice utilisée pour l'obtenir est $C\Psi_{n_S+1}$
— cas 3 : la représentation creuse recherchée est n_S-*sparse* de taille n_S+2 et la matrice utilisée pour l'obtenir est $C\Psi_{n_S+2}$

Par exemple, pour le cas 2, le problème suivant est résolu pour chaque *snapshot* $\boldsymbol{s}$:

$$\boldsymbol{a} \in \underset{\widetilde{\boldsymbol{a}} \in \mathbb{R}^{n_S+1}}{\arg\min} ||\boldsymbol{s} - C\Psi_{n_S+1}\,\widetilde{\boldsymbol{a}}||_2 \quad \text{tel que} \quad ||\widetilde{\boldsymbol{a}}||_0 \leq K \tag{3.15}$$

La reconstruction est désastreuse dès que $n_D > n_S$. D'autres algorithmes de reconstruction creuse ont été testés (IHT et des algorithmes ℓ_1) mais les performances ne sont pas améliorées.

La base POD ne peut pas être directement utilisée pour obtenir des représentations creuses performantes à l'aide d'OMP.

Base DMD

L'approche Dynamic Mode Decomposition (DMD) [17, 18] est très répandue dans la communauté de la mécanique des fluides. Elle suppose que les *snapshots* sont générés par un processus linéaire, stationnaire, discret représenté par la matrice $Z \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}}$. Ainsi, le *snapshot* $\boldsymbol{y}_{i+1}$ est supposé vérifier $\boldsymbol{y}_{i+1} = Z\boldsymbol{y}_i$.

L'objectif de l'approche DMD est d'obtenir les vecteurs propres et valeurs propres de Z sans avoir besoin de déterminer cette matrice. Ces vecteurs propres formeront ensuite la base de décomposition DMD recherchée.

Les deux matrices suivantes sont maintenant définies :

$$\begin{aligned} Y_2 &= \left[\boldsymbol{y}_2, \boldsymbol{y}_3, \ldots, \boldsymbol{y}_{n_{\text{snap}}}\right] \\ Y_1 &= \left[\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n_{\text{snap}}-1}\right] \end{aligned} \tag{3.16}$$

La matrice Z doit ensuite vérifier : $Y_2 = ZY_1$. L'algorithme DMD permet d'obtenir la meilleure représentation F de Z dans la base POD associée à Y_1. Soit U la base POD de Y_1 (obtenue à partir d'une SVD réduite de $Y_1 = U\Sigma V^T$), la matrice F vérifie :

$$Z = UFU^T \tag{3.17}$$

La matrice F est ensuite approximée par

$$F = U^T Y_2 V \Sigma^{-1} \tag{3.18}$$

La base DMD peut ensuite être définie comme $D_{\text{DMD}} = U\Phi_F$ où Φ_F contient les vecteurs propres de F.

Malheureusement, l'utilisation d'une telle base (non orthogonale contrairement à la base POD) ne s'est pas avérée compatible avec une approche représentation creuse. L'algorithme OMP n'a pas pu fournir des représentations creuses satisfaisantes à partir de la matrice CD_{DMD} et des mesures.

Base BPOD (ERA)

BPOD est l'acronyme de Balanced Proper Orthogonal Decomposition. Elle consiste en l'obtention de modes à la fois commandables et observables, deux propriétés très recherchées dans l'objectif d'effectuer de l'estimation/de la commande. Si une telle base permet une représentation creuse, elle peut être à l'origine d'une estimation performante et même, d'une commande performante.

Plus de détails peuvent être obtenus dans [19] et les principales étapes sont listées ci-dessous. Pour obtenir une telle base, il faut exciter le système avec des entrées particulières. Il faut en effet obtenir des informations sur l'observabilité et la commandabilité.

La nature des entrées appliquées doit être impulsionnelle. La forme de l'impulsion est aussi importante puisqu'il est impossible de générer l'entrée théorique nécessaire : l'impulsion de dirac. Une impulsion lissée a été utilisée. 3 séquences d'impulsions ont été appliquées pour capturer le plus de dynamique. La première consiste à appliquer l'impulsion sur u_1, la deuxième consiste à appliquer l'impulsion sur u_2 et la troisième consiste à appliquer les deux impulsions en même temps. Chaque séquence est appliquée une fois le régime permanent atteint. Une partie seulement des *snapshots* recueillis est agglomérée dans des matrices. Cette restriction est faite pour s'assurer que les *snapshots* retenus contiennent le plus d'information possible sur les dynamiques du système étudié. La méthode présentée dans l'article [20] a été utilisée.

Malgré un soin tout particulier apporté aux réglages des différents paramètres, une telle base ne s'est pas révélée compatible avec les représentations creuses recherchées.

Conclusion

Ces trois bases (POD, DMD, BPOD) s'avèrent inefficaces, vu l'utilisation recherchée. Finalement, leur défaut est d'être trop efficaces d'un point de vue de l'énergie modale.

Elles ne font pas apparaître la **redondance** nécessaire pour rendre les représentations creuses intéressantes. Afin de pouvoir utiliser plus de modes que le nombre de capteurs, il faut utiliser une base qui présente des modes **qui se ressemblent** (corrélés). Suivant les mesures effectuées, la reconstruction ne se fera pas toujours à l'aide des mêmes modes. Il est donc nécessaire de créer une base spécialement dans l'objectif d'obtenir une représentation creuse.

3.2 Création d'une base adaptée au problème

Il faut créer une base, un dictionnaire, Φ qui permet d'obtenir une représentation creuse. Ce dictionnaire doit donc présenter de la redondance.

L'algorithme le plus mentionné permettant de répondre à ce problème est l'algorithme K-SVD. Ce dernier complète l'algorithme K-means très utilisé en classification. Une fois l'algorithme K-SVD présenté, il sera intégré dans la méthode d'estimation proposée.

3.2.1 Application directe de la K-SVD

L'algorithme K-SVD [21] fait partie des algorithmes les plus utilisés pour créer des **dictionnaires redondants promouvant des représentations creuses**. Il existe bien sûr d'autres approches telles que la MOD (Method of Directions) [9] qui est très proche de la K-SVD sur le fond. Il convient aussi de mentionner la fameuse décomposition en ondelettes [22], une méthode consistant a créer un dictionnaire composé d'une union de bases orthonormales [23] ainsi qu'une méthode créant un dictionnaire qui cette fois-ci promeut une parcimonie globale [24].

La K-SVD peut être vue comme une généralisation de l'algorithme K-means [25] très utilisé en classification. Cet algorithme est détaillé dans le chapitre 4. Dans le cadre K-means, une représentation 1-*sparse* contrainte (car la valeur non nulle dans la représentation réduite doit être de 1) était recherchée, alors que dans le cadre de la K-SVD, une représentation K-*sparse* est recherchée. Ainsi, une combinaison linéaire d'un nombre restreint de vecteurs de bases est utilisée pour décrire chaque élément de la séquence d'apprentissage.

$K \in \mathbb{N}^*$ est la parcimonie recherchée sur les représentations réduites et $K \geq 1$ (les représentations sont donc T-*sparse*). Le nombre de modes K-SVD recherché est $n_D \in \mathbb{N}^*$. La matrice contenant les représentations réduites est A, définie de la même manière qu'au chapitre précédent (voir la partie 2.53). Dans la suite, une matrice Y réelle est utilisée.

Formulation

Comme il a été vu à travers les équations 3.6 et 3.7, il existe deux manières de formuler le problème.

La *première* cherche à minimiser l'erreur $||Y - \widehat{Y}||_F$ en ajoutant une contrainte sur la norme ℓ_0 des représentations. Celle-ci est connue sous le nom de *error-based K-SVD* et se trouve ci-dessous :

$$\{A, \Phi\} \in \underset{\substack{\widetilde{A} \in \mathbb{R}^{n_D \times n_{\text{snap}}} \\ \widetilde{\Phi} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_D}, ||\widetilde{\boldsymbol{\phi}}_i||_2 = 1 \, \forall i}}{\arg\min} ||\widetilde{\boldsymbol{a}}_l||_0 \quad \forall l \quad \text{tel que} ||Y - \widetilde{\Phi}\, \widetilde{A}||_F \leq \epsilon_{\text{K-SVD}}, \tag{3.19}$$

où $\epsilon_{\text{K-SVD}} \in \mathbb{R}_+^*$ est la plus grande erreur autorisée sur la reconstruction de Y.

La *seconde* cherche à minimiser la norme ℓ_0 des représentations tout en satisfaisant une contrainte sur l'erreur $||Y - \widehat{Y}||_F$. Cette formulation est connue sous le nom de *sparsity-based K-SVD* et se trouve ci-dessous :

$$\{A, \Phi\} \in \underset{\substack{\widetilde{A} \in \mathbb{R}^{n_D \times n_{\text{snap}}} \\ \widetilde{\Phi} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_D}, ||\widetilde{\boldsymbol{\phi}}_i||_2 = 1 \ \forall i}}{\arg\min} ||Y - \widetilde{\Phi}\, \widetilde{A}||_F \quad \text{tel que} \quad ||\widetilde{\boldsymbol{a}}_l||_0 \leq K \quad \forall l, \tag{3.20}$$

où K est le plus grand nombre de termes non-nuls possibles.

La formulation retenue est celle dite ***sparsity-based K-SVD*** (3.20) puisque l'objectif est d'obtenir la plus petite erreur de reconstruction possible sur Y en utilisant la plus grande parcimonie possible, c'est-à-dire $K = n_S$. Le choix de la valeur de K sera étudié dans la sous-section suivante.

K-SVD

Outre la phase d'initialisation et le critère d'arrêt, l'algorithme K-SVD permettant de résoudre le problème 3.20 peut se décomposer en 2 grandes étapes (nommées selon [21]) :
— *Sparse Coding* (**SC**) et
— *Codebook Update* (**CU**).
Ceci permet de **découpler** la recherche de A et de Φ.

L'étape **SC** consiste à déterminer une représentation creuse à partir de Y et de l'état courant du dictionnaire Φ. Le problème à résoudre est :

$$A_{\text{SC}} \in \underset{\widetilde{A}_{\text{SC}} \in \mathbb{R}^{n_D \times n_{\text{snap}}}}{\arg\min} ||Y - \Phi\, \widetilde{A}_{\text{SC}}||_F \quad \text{tel que} \quad ||A_{\text{SC},l}||_0 \leq K \quad \forall l, \tag{3.21}$$

où $A_{\text{SC}} := (A_{\text{SC},1} \dots A_{\text{SC},n_{\text{snap}}})$ contient les représentations creuses recherchées. $A_{\text{SC},l}$ est la $l^{\text{ième}}$ colonne de A_{SC}. Le dictionnaire est fixé, ainsi ce problème est résolu pour chaque $\boldsymbol{y}$ à l'aide d'un algorithme d'obtention de représentation creuse. Comme détaillé précédemment, **l'algorithme OMP est utilisé ici**.

L'étape **CU**, comme son nom l'indique, consiste à mettre à jour le dictionnaire Φ. Il est par contre impossible de mettre à jour Φ tout en gardant les représentations déterminées à l'étape précédente, soit A_{SC}. Changer un élément du dictionnaire fait que la représentation creuse associée sera différente. Un comportement *greedy* est ainsi introduit à cette étape. Chaque colonne de Φ, c'est-à-dire chaque mode du dictionnaire, va être mis à jour avec la ligne associée de A_{SC}. Ceci est fait avec la contrainte supplémentaire de ne pas modifier le support de A_{SC}. Une telle contrainte permet de rester dans un contexte creux tout en rendant ce sous problème soluble par des méthodes linéaires. Au fur et à mesure que les modes sont mis à jour, la représentation creuse s'éloigne de celle qui a été obtenue à l'étape précédente. Le problème à résoudre dans l'étape CU est :

$$\begin{aligned} \{A, \Phi\} \quad &\in \quad \underset{\widetilde{A}, \widetilde{\Phi}}{\arg\min} ||Y - \widetilde{\Phi}\, \widetilde{A}||_F \\ &\text{tel que} \quad \text{supp}\,(A) = \text{supp}\,(A_{\text{SC}})\,, \quad \|\boldsymbol{\phi}_k\|_2 = 1 \quad \forall k. \end{aligned} \tag{3.22}$$

La méthode de mise à jour est maintenant détaillée. Des nouvelles notations relatives aux manipulations des matrices sont introduites :
— $(\cdot)^l$, qui indique la $l^{\text{ième}}$ ligne d'une matrice,
— $(\cdot)^{\backslash l}$, qui indique la matrice privée de sa $l^{\text{ième}}$ ligne et

— $(\cdot)_{\backslash l}$, qui indique la matrice privée de sa $l^{\text{ième}}$ colonne.
Cette notation peut aussi être étendue aux cas où l est un ensemble d'indices. Dans ce cas, un groupement de lignes ou de colonnes peut être manipulé.

Chaque paire $(\boldsymbol{\phi}_l, A^l_{\text{SC}})$ est mise à jour à l'aide du calcul d'une **SVD**. C'est d'ailleurs cette étape qui explique le terme SVD inclus dans le nom de l'algorithme K-SVD. Cette mise à jour est faite dans un ordre **aléatoire** afin de balayer un plus grand espace des solutions. La mise à jour commence par le recensement des *snapshots* qui utilise effectivement $\boldsymbol{\phi}_l$. Ces indices sont contenus dans l'ensemble ω_l qui est défini par

$$\omega_l = \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l_{\text{SC}}(i) \neq 0 \right\}, \tag{3.23}$$

où $A^l_{\text{SC}}(i)$ correspond au $i^{\text{ème}}$ élément de A^l_{SC}.
La matrice erreur est ensuite formée par :

$$E = Y - \Phi_{\backslash l} A^{\backslash l}_{\text{SC}}. \tag{3.24}$$

La restriction de E aux colonnes indexées par ω_l est nommée E_r. Enfin, la SVD de E_r est calculée et permet d'obtenir les matrices U, Σ et V telles que $E_r = U\Sigma V^T$. U et V sont des matrices orthogonales et Σ est une matrice contenant les valeurs singulières de E_r sur sa diagonale et des 0 ailleurs. Ceci permet d'effectuer la mise à jour de la paire considérée selon

$$A^l_{\text{SC},r} = \Sigma_{1,1} V^1, \quad \boldsymbol{\phi}_l = U_1, \tag{3.25}$$

où $A^l_{\text{SC},r}$ est la restriction de A^l_{SC} aux éléments indexés par ω_l.

Ces deux étapes sont répétées jusqu'à l'obtention d'une solution satisfaisante. Le pseudo-code se trouve dans l'algorithme 2. Plus d'informations sur l'algorithme K-SVD peut-etre trouvé dans l'article fondateur [21]. Ensuite, des détails sur l'implémentation peuvent être trouvés dans [26] où, entre autre, le calcul de la SVD est approché pour réduire le temps de calcul pris par l'étape CU.

Un autre point essentiel est la mise à jour de chaque élément du dictionnaire lors de l'étape CU. Comme il a été vu précédemment, plus le nombre de modes mis à jour augmente, plus la représentation associée est éloignée de celle obtenue par l'étape SC. Une autre approche de mise à jour testée consiste à ne modifier que le dictionnaire sans modifier la représentation réduite. Ceci permet une variation de l'erreur moins brutale mais amplifie le phénomène d'être piégé dans un minimum local. A cause de ce phénomène, cette modification n'a pas été retenue.

Implémentation et remarques techniques

L'algorithme K-SVD est maintenant utilisé pour répondre à *Pb.Estimation*. Les paragraphes suivants contiennent les informations pratiques nécessaires pour utiliser l'algorithme sur le système test. Ensuite, les performances obtenues avec la K-SVD sont illustrées et comparées avec l'approche POD.

Les paramètres nécessaires pour utiliser l'algorithme sont :
— la séquence d'apprentissage Y,
— le critère d'arrêt,
— la taille du dictionnaire n_D,
— la parcimonie des vecteurs recherchée K,

Algorithm 2 K-SVD Pseudo-Code

Require: Y, n_D, K, D_{ini}, n_{it}

1: **Initialisation** :
2: $\Phi \leftarrow D_{\text{ini}}$
3:
4: **for** $j \in [1 \dots n_{\texttt{it}}]$ **do**
5: **Sparse Coding** :
6: $A \leftarrow \underset{\widetilde{A}_{\text{SC}} \in \mathbb{R}^{n_D \times n_{\text{snap}}}}{\arg\min} \quad ||Y - \Phi\, \widetilde{A}_{\text{SC}}||_F \quad \text{tel que} \quad ||A, l||_0 \leq K \quad \forall l,$
7:
8: **Codebook Update** :
9: **for** $l \in [1 \dots n_D]$ **do**
10: $\omega_l \leftarrow \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l(i) \neq 0 \right\}$
11: $E \leftarrow Y - \Phi_{\backslash l} A^{\backslash l}$
12: $E_r \leftarrow E_{\omega_l}$
13: $U \Sigma V^T \leftarrow$ décomposition SVD de E_r
14: $A_r^l = \Sigma_{1,1} V^1$
15: $\boldsymbol{\phi}_l \leftarrow U_1$
16:
17: **Sortie** : Φ, A

— l'initialisation .

Le **critère d'arrêt** retenu est ici le nombre d'itérations effectuées n_{it}. Une itération correspond à un cycle SC - CU. Bien que des critères sur la valeur de l'erreur effectuée ou sur la variation de cette erreur soient plus adaptés à un usage courant, le nombre d'itérations a été retenu afin d'être sûr d'atteindre l'arrêt de l'algorithme lors de chaque test.

De plus, il est courant d'avoir des variations brutales de l'erreur lors de l'utilisation de cet algorithme. La cause principale des variations brutales du critère à minimiser est le remplacement de vecteur de base. Ceci a lieu dans deux cas :

— un vecteur de base n'est utilisé par **aucune** représentation réduite,
— un vecteur de base est trop proche, d'un point de vue **angulaire**, d'un autre mode du dictionnaire.

Une collection de vecteurs non utilisés est créée lors de l'initialisation. Lorsqu'un vecteur de base vérifie l'une des deux conditions ci-dessus, il est remplacé par un vecteur de cette collection qui est ensuite mise à jour. Ceci permet de s'assurer qu'un élément n'est pas utilisé plus d'une fois. En choisissant un *grand* nombre d'itérations, on autorise des variations lentes de l'erreur qui peuvent créer des situations où un vecteur de base doit être remplacé. Plus de situations sont ainsi balayées.

Le critère de **parcimonie** est choisi tel que $K = n_S$. Il est ici attendu de pouvoir récupérer n_S termes non-nuls de la représentation réduite à partir de n_S mesures. Ceci est optimiste et sera étudié dans la suite (cas $K < n_S$). Il faut aussi se souvenir que l'algorithme OMP ne peut fournir qu'un vecteur au plus n_S-*sparse* à partir d'un vecteur d'entrée de n_S éléments. Un autre algorithme est ainsi plus adapté si une parcimonie plus faible (plus de termes non nuls) est recherchée.

La **taille du dictionnaire** n_D doit ensuite être fixée. Afin de pouvoir obtenir une meilleure estimation par rapport à la POD, il faut au moins utiliser plus de n_S modes. En

effet, l'approche POD est limitée à utiliser autant de modes que le nombre de capteur afin de garantir un système bien posé pour l'obtention des représentations réduites. Ensuite, plus le dictionnaire contient de modes, plus il va tirer parti des avantages des représentations creuses pour décrire convenablement Y. Il faut ainsi $n_D > K$, ce qui est le cas puisque $K = n_S$. Cependant, outre les temps de calcul des algorithmes creux légèrement rallongés, il va être de plus en plus délicat d'obtenir une représentation creuse exploitable à partir des mesures seulement. Un compromis doit donc être trouvé entre les performances rendues possibles par un *grand* dictionnaire et le fait de pouvoir en pratique obtenir une telle représentation.

Vu l'importance de la parcimonie K sur le choix de la taille du dictionnaire, un outil plus adapté est utilisé : le degré de parcimonie d'un vecteur. Ce dernier, appelé encore **multiplicateur** $n_{\text{mul}} \in \mathbb{R}_+^*$, est défini comme le rapport entre la taille du dictionnaire et la parcimonie de la représentation réduite, soit

$$n_{\text{mul}} = \frac{n_D}{K}. \tag{3.26}$$

Parmi les valeurs testées, $n_{\text{mul}} = 3$ semble bien répondre au compromis recherché. D'autres valeurs sont aussi illustrées dans la suite telles que $n_{\text{mul}} = 10$ qui au prix d'un temps de calcul plus long permet d'obtenir de bien meilleures performances (dans le cas non-bruité). De plus il est maintenant plus aisé de comparer des bases de tailles différentes mais de même multiplicateur.

Un dernier point sur la taille du dictionnaire concerne le phénomène dit *d'**overfitting***. Il ne faut pas oublier que la méthode développée doit aussi être exploitable sur des champs non contenus dans la séquence d'entraînement. Augmenter de manière trop importante la taille du dictionnaire fait que ce dernier va être de plus en plus adapté à la séquence d'entraînement. Un cas extrême serait un dictionnaire contenant autant de modes que de *snapshots*. Dans ce cas, utiliser l'algorithme K-SVD en recherchant une solution 1-*sparse* ferait que chaque mode du dictionnaire soit en fait un *snapshot*. Afin de pouvoir être flexible, il ne faut pas utiliser un dictionnaire trop *grand*. Une modification de la K-SVD pour répondre à ce problème a été proposée dans [27] avec la Double Sparse K-SVD. Cette méthode est facile à mettre en pratique une fois qu'un algorithme K-SVD classique est disponible. La recherche du dictionnaire se fait maintenant sous contrainte. Un premier dictionnaire D_1 sert de base pour le second dictionnaire D_2 qui doit être déterminé. Le dictionnaire total sera $\Phi = D_1 D_2$, où D_1 et D_2 ont les dimensions appropriées. Le dictionnaire D_1 permet de guider la recherche de Φ et conduit, d'après l'auteur, à la réduction de ce phénomène d'overfitting. Bien sur, le choix de D_1 devient crucial au bon fonctionnement de la méthode et nécessite aussi un peu de pratique avant de pouvoir en choisir une performante.

Enfin, la **nature aléatoire** de l'algorithme fait qu'il est essentiel de le lancer plusieurs fois pour un jeu de paramètre donné. Le balayage aléatoire des éléments du dictionnaire lors de l'étape de *Codebook Update* permet de balayer un large spectre de dictionnaire mais peut conduire à des solutions locales, non-optimales. La méthode retenue à été d'initialiser différemment l'algorithme et d'effectuer plus de 30 lancers par simulation. Le meilleur dictionnaire est ensuite retenu.

Il ne reste plus qu'à régler le problème de l'**initialisation**. Plusieurs techniques ont été utilisées. Ces dernières sont :

— D_{ini1} : initialisation avec **nombres aléatoires** pris selon une distribution **uniforme centrée** puis normalisation des colonnes,

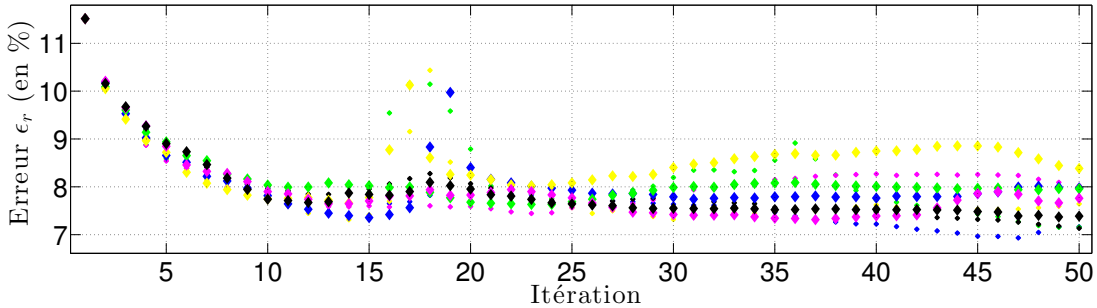

FIGURE 3.2 – K-SVD - Illustration du comportement aléatoire

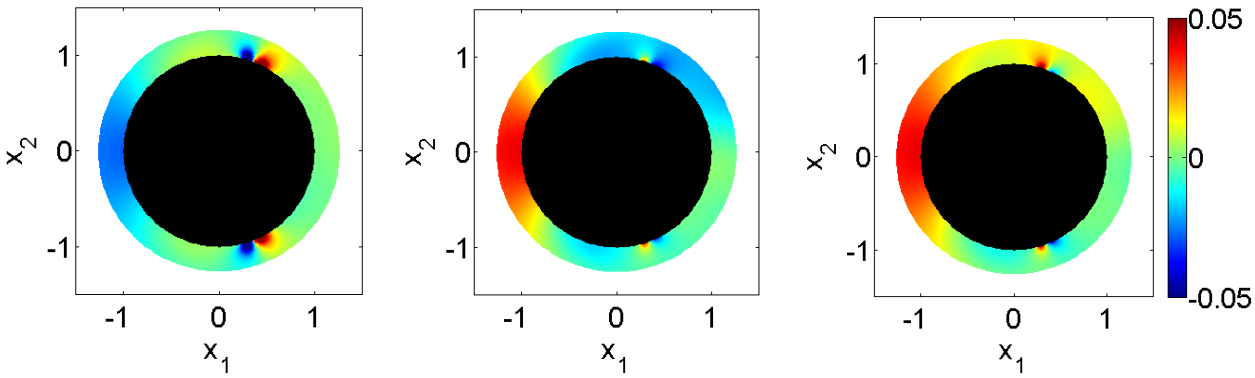

FIGURE 3.3 – K-SVD - 3 premiers modes

— D_{ini2} : initialisation avec des nombres aléatoires pris selon une distribution **gaussienne centrée** puis normalisation des colonnes,
— D_{ini3} initialisation avec une combinaison aléatoire de *snapshots* de Y,
— D_{ini4} initialisation avec un mélange de modes POD et une combinaison aléatoire de *snapshots* de Y.

En pratique, D_{ini1} et D_{ini4} se sont avérés très satisfaisants. D_{ini1} permet de balayer un espace de solutions "plus large" (lorsque plusieurs lancers sont considérés) que D_{ini4} mais au prix d'une convergence plus lente.

Performances - données complètes

Les performances de l'approche K-SVD sont maintenant illustrées à partir des données générées par le système test. L'information **totale** Y est disponible. **Les capteurs ne sont pas utilisés**. Ceci permet de se faire une idée plus concrète des possibilités de la K-SVD et de ses limites. La majorité des courbes traite de l'erreur de reconstruction relative $\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F$. Par contre $\widehat{Y}$ est obtenue de différentes manières.

La figure 3.2 illustre la variation de ϵ_r en fonction du nombre d'itérations effectuées pour 10 lancers de l'algorithme K-SVD avec les mêmes conditions initiales. Le type D_{ini3} est ici utilisé. Cette matrice est stockée afin d'être utilisée plusieurs fois par l'algorithme K-SVD. A la fin de chaque cycle SC-CU, l'erreur relative ϵ_r est calculée avec $\widehat{Y} = \Phi_k A_{\text{SC}k}$, où Φ_k correspond au dictionnaire obtenu à la fin du cycle k et $A_{\text{SC}k}$ contient les représentations creuses associées à ce dictionnaire. Les 50 premières itérations sont retenues. La courbe illustre ainsi le comportement aléatoire de l'algorithme K-SVD.

La figure 3.3 illustre 3 modes K-SVD. Le dictionnaire K-SVD utilisé ici est associé

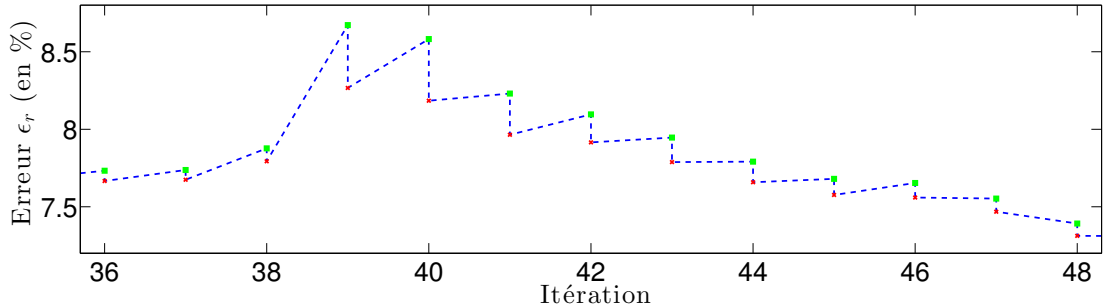

FIGURE 3.4 – K-SVD - ϵ_r obtenue à la fin du SC (carré) et du CU (croix)

au cas $K = 3$ et $n_{\text{mul}} = 3$. De gauche à droite, les modes représentés sont : 3,5 et 8.

La figure 3.4 permet d'illustrer comment l'algorithme K-SVD réduit l'erreur ϵ_r (obtenue en fin de SC). Seules quelques itérations d'un lancer sont représentées. Au lieu de ne considérer que l'erreur obtenue en fin de SC (marqueur carré), l'erreur obtenue en fin d'étape CU (marqueur croix) est aussi affichée. Cette dernière est calculée lorsque chaque paire $(\boldsymbol{\phi}_l, A^l_{\text{SC}})$ a été mise à jour. La matrice A_{SC} utilisée lors de cette étape a ainsi été obtenue par SVD successives dépendantes (et non par l'algorithme OMP seulement). L'étape CU permet toujours de réduire l'erreur de description faite sur la séquence d'entraînement.

Si l'objectif était d'obtenir la meilleure représentation creuse de Y, il suffirait de sélectionner le dictionnaire et la représentation associée à la fin de l'étape CU. Cela dit, c'est la représentation creuse obtenue à l'étape SC qui sera exploitable en pratique.

Ce paragraphe a pour fonction d'illustrer les performances de l'approche K-SVD dans le cas où la totalité du champ est connue. Cela revient à utiliser $C = I_{n_{\boldsymbol{x}}}$ avec la formulation considérée, où $I_{n_{\boldsymbol{x}}}$ est la matrice identité d'ordre $n_{\boldsymbol{x}}$. Bien que cette situation soit idyllique, elle permet de comparer les performances descriptives des méthodes K-SVD et POD. Elle sert aussi de limite théorique illustrant le cas où aucune erreur n'est effectuée sur la représentation réduite (creuse ou non).

Pour la méthode POD, l'erreur relative ϵ_r est calculée en retenant un certain nombre de modes K. Les capacités descriptives de la POD sont ainsi étudiées, comme cela a été vu dans le chapitre précédent. La configuration K-SVD la plus comparable consiste à utiliser une parcimonie de K. Ainsi, pour chaque dictionnaire K-SVD testé, seulement K modes peuvent être utilisés à la fois, comme l'approche POD. Plusieurs tailles de dictionnaires sont utilisées et sont caractérisées par leur degré de parcimonie $n_{\text{mul}} = 1, 3$ et 10. $n_{\text{mul}} = 1$ est un cas limite où les représentations creuses ne le sont plus, elles sont en fait pleines. Les autres n_{mul} correspondent à des situations de parcimonie classique. La représentation réduite pour la POD est obtenue par projection orthogonale du *snapshot* considéré. Pour obtenir celle correspondant à l'approche K-SVD, OMP est utilisé afin de trouver une représentation creuse *K-sparse*. Ainsi, le même nombre de modes est effectivement utilisé, malgré les différentes tailles de dictionnaires/bases. Pour chaque paire (n_{mul}, K), le meilleur dictionnaire K-SVD est retenu (selon les critères vu précédemment).

Les résultats sont regroupés sur la figure 3.5 et dans le tableau 3.2.

Plus le nombre de modes effectifs utilisé K est grand, plus l'erreur relative faite sur la description de la totalité de la séquence d'entraînement diminue. Avec une parcimonie $K = 5$ et un dictionnaire de taille $n_D = 10K = 50$, l'erreur relative effectuée est inférieure à 1%. Cette erreur diminue très vite lorsque K augmente. Ceci est une conséquence de la

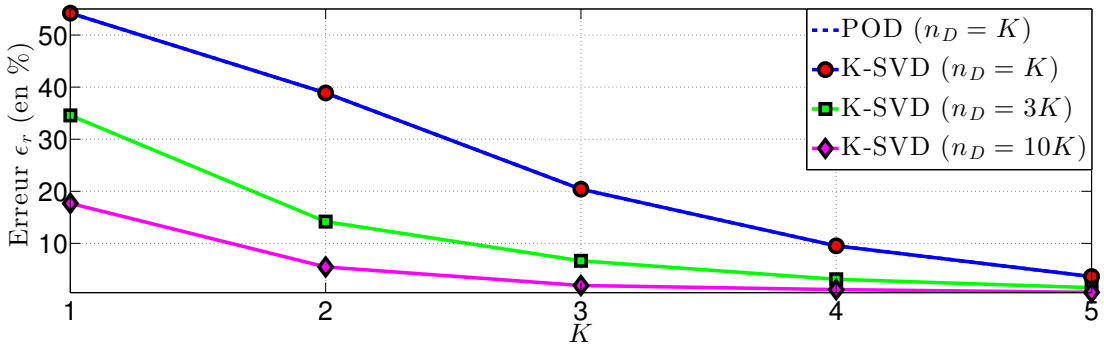

FIGURE 3.5 – K-SVD - Performances $\boldsymbol{y}$ disponible

K	POD $n_D = K$	K-SVD $n_D = K$	K-SVD $n_D = 3\,K$	K-SVD $n_D = 10\,K$
1	54.2%	54.2%	34.6%	17.7%
2	38.9%	38.9%	14.2%	5.48%
3	20.4%	20.4%	6.65%	1.92%
4	9.50%	9.53%	3.11%	1.12%
5	3.60%	3.63%	1.50%	0.596%

TABLE 3.2 – K-SVD - Performances $\boldsymbol{y}$ disponible

nature de la séquence d'apprentissage utilisée qui s'avère compressible (peut être décrit convenablement avec peu de modes). Pour un autre système test présentant des phénomènes plus complexes, il faudrait plus de modes pour obtenir des erreurs aussi faibles.

Le premier comportement notable concerne la courbe $n_{\text{mul}} = 1$, c'est-à-dire $n_D = K$. Les performances de K-SVD avec $n_D = K$ sont si similaires à celles obtenues avec l'approche POD que les deux courbes se superposent.

Du point de vue de l'erreur ϵ_r, la base K-SVD trouvée se comporte comme la base POD. Cependant, sauf pour le cas $n_{\text{mul}} = 1$, les dictionnaires K-SVD D et les bases POD Ψ sont différents. Cette base POD est d'ailleurs optimale dans ce cas ($n_{\text{mul}} = 1$) selon le théorème d'Eckart-Young [28]. L'algorithme K-SVD permet donc de trouver un dictionnaire tout aussi performant dans le cas plein.

Ensuite, lorsque $n_D > K$, la base K-SVD permet de **meilleures performances que la base POD**. Ceci est l'effet recherché à travers l'utilisation de représentations creuses. Lorsque 3 modes effectifs sont utilisés, l'erreur est de 20% avec une représentation dense et diminue jusqu'à moins de 2% avec une représentation creuse ($n_{\text{mul}} = 10$).

Performances - données limitées

Il ne reste plus qu'à vérifier si cette approche fonctionne lorsque la seule information disponible est celle donnée par les capteurs. Les dictionnaires obtenus dans le cadre de la figure 3.5 sont utilisés. Les représentations réduites sont maintenant obtenues en résolvant l'équation 2.62 pour l'approche POD et l'équation 3.21 pour l'approche K-SVD. Les positions de capteurs $\mathcal{C}_{\text{opt}}$ sont utilisées (voir le chapitre 2).

Les résultats associés se trouvent dans le tableau 3.3 et doivent être comparés au tableau 3.2.

K	POD $n_D = K$	K-SVD $n_D = K$	K-SVD $n_D = 3\,K$	K-SVD $n_D = 10\,K$
1	64.0%	64.0%	72.2%	74.9%
2	46.6%	46.6%	92.4%	152%
3	20.6%	20.6%	38.7%	32.2%
4	12.3%	12.3%	101%	13.7%
5	3.68%	3.68%	76.1%	7.43%

TABLE 3.3 – K-SVD - Performances $\mathcal{C}_{\text{opt}}$

Les performances ne sont pas du tout compatibles avec ce qui est attendu. Les représentations creuses obtenues permettent bien de réduire l'erreur au niveau des capteurs, mais ces derniers ne sont pas du tout adaptés pour réduire l'erreur sur le champ total à estimer. Les mêmes essais ont été effectués avec d'autres réalisations de dictionnaire K-SVD et les résultats sont encore catastrophiques. Certaines peuvent donner de meilleurs résultats mais rien n'est garanti. Pour un plus grand nombre de capteurs (10 par exemple), cet effet semble s'estomper. Étant donné que la méthode doit être fonctionnelle dans un contexte de faible nombre de capteurs, il faut modifier l'approche retenue.

Cette perte de performances peut être vue à travers la variété des **supports** des représentations réduites (voir 3.1.1). Le dictionnaire K-SVD, noté D, obtenu dans le cas $K = 3$ et $n_D = 9$ est considéré. Les capteurs retenus pour ce test sont associés aux positions $\mathcal{C}_{\text{opt}_3}$. La matrice A contient des vecteurs 3-*sparse* obtenus à l'aide de l'algorithme OMP appliqué sur la paire (Y, D). A_C est obtenue de la même manière mais avec la paire (S, CD).

Les **supports distincts** présents dans A (courbe du haut) et A_C (courbe du bas) sont représentés sur la figure 3.6. Les croix indiquent la présence d'un élément non nul parmi les composantes du vecteur.

Le nombre de supports maximal possible est $\binom{9}{3} = 84$. Lorsque l'information totale $\boldsymbol{y}$ est disponible (paire (Y, D)), il y a 46 supports distincts qui sont utilisés dans la matrice A. Cependant, lorsque seulement les mesures $\boldsymbol{s}$ sont disponibles (paire (S, CD)), il n'y a que 9 supports distincts utilisés dans A_C .

Les supports obtenus à partir de l'information complète sur le champ $\boldsymbol{y}$ sont bien **plus divers** que ceux obtenus à partir des mesures $\boldsymbol{s}$. Cependant, sans augmenter le nombre de supports distincts, **il doit être possible d'augmenter les performances d'estimation à partir des données $\boldsymbol{s}$**. La figure 3.5 augure de très bonnes performances avec des représentations creuses **si ces dernières deviennent accessibles** à partir des mesures $\boldsymbol{s}$.

3.2.2 Intégration de la position des capteurs - SOBAL

L'algorithme K-SVD doit être **modifié** afin d'**incorporer la position des capteurs**. Il semble nécessaire d'y introduire le fait que les représentations creuses ne peuvent être obtenues qu'à partir de $\boldsymbol{s}$ et non de $\boldsymbol{y}$. Cela ne sert à rien d'autoriser l'existence de représentations creuses efficaces s'il est impossible d'y accéder.

L'algorithme K-SVD est très flexible et a déjà été modifié pour convenir à différents besoins. Cet algorithme est le point de départ des algorithmes :

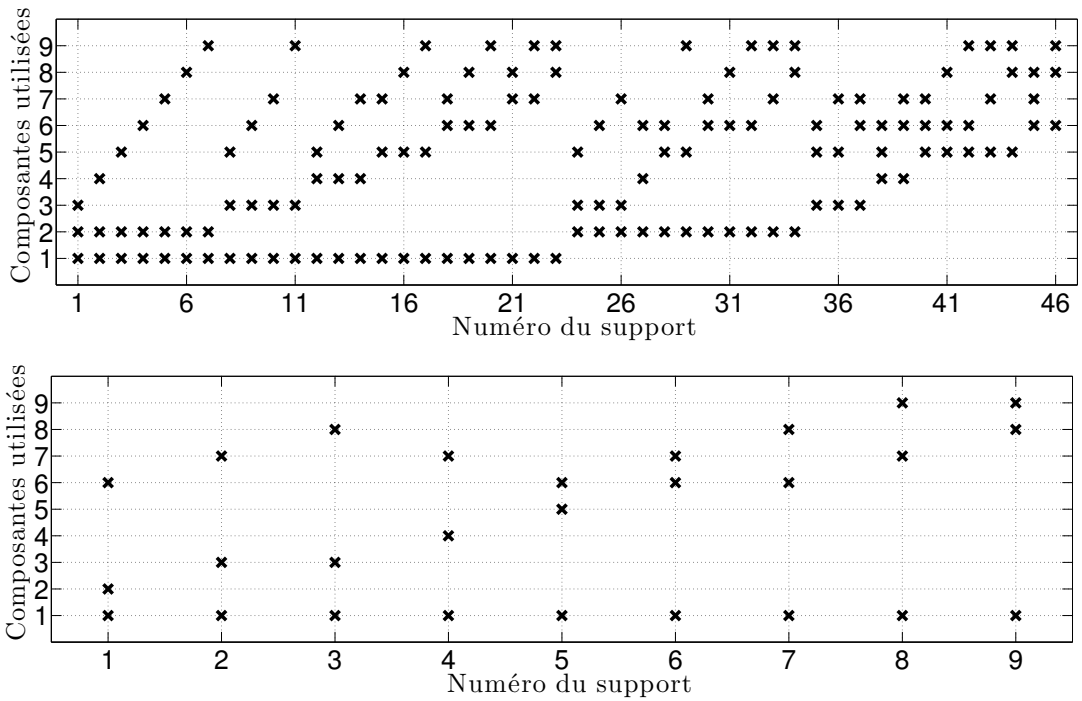

FIGURE 3.6 – K-SVD - Supports distincts des représentations creuses obtenues à partir des données complètes (haut) et des mesures (bas)

— Single-Pass K-SVD [29], qui traite conjointement l'étape *Sparse Coding* et *Codebook Update* pour être plus efficace d'un point de vue stockage mémoire,
— Label Consistant-KSVD (LC-KSVD) [30] qui permet d'obtenir, en plus du dictionnaire, un classificateur linéaire,
— MTL-KSVD [31] qui permet de faire de l'apprentissage en ligne de multiples tâches consécutives,
— Discriminative-KSVD (D-KSVD) [32] qui se focalise sur la discrimination de différentes classes.

Afin de répondre au problème, l'étape ***Sparse Coding* est modifiée**. Le reste de l'algorithme est laissé à l'identique. Cette étape est cruciale puisqu'elle définit les *supports* qui seront ensuite utilisés par l'étape *Codebook Update*. Ainsi, en sacrifiant la variété des supports lors de la conception afin de se limiter à des supports **réellement accessibles à partir des mesures**, l'étape CU va **améliorer les performances pratiques**. Une partie du potentiel descriptif (erreur faite sur l'estimation de Y) est sacrifiée mais le gain effectué sur l'accessibilité de représentations creuses performantes permet d'atteindre les résultats recherchés.

Au lieu d'utiliser la paire (Y, Φ) pour obtenir la représentation creuse, la paire $(S, C\Phi)$ est utilisée. C'est cette paire qui est exploitée en pratique. L'étape SC est remplacée par l'étape ***Sensor Oriented Sparse Coding*** :

$$A_{\mathrm{SC}} \in \underset{\widetilde{A}_{\mathrm{SC}} \in \mathbb{R}^{n_D \times n_{\mathrm{snap}}}}{\arg\min} \quad ||S - C\,\Phi\,\widetilde{A}_{\mathrm{SC}}||_F \quad \text{tel que} \quad ||A_{\mathrm{SC},l}||_0 \leq K \quad \forall l, \tag{3.27}$$

L'algorithme obtenu est nommé par la suite **SOBAL** qui est l'acronyme de **S**ensor **O**riented **BA**sis **L**earning algorithm.

Le pseudo-code de l'algorithme proposé se trouve dans l'algorithme 3.

Algorithm 3 SOBAL Pseudo-Code

Require: Y, n_D, K, D_{ini}, n_{it}

1: **Initialisation** :
2: $\Phi \leftarrow D_{\text{ini}}$
3:
4: **for** $j \in [1 \dots n_{\texttt{it}}]$ **do**
5: **Sensor Oriented Sparse Coding** :
6: $A \leftarrow \underset{\widetilde{A}_{\text{SC}} \in \mathbb{R}^{n_D \times n_{\text{snap}}}}{\arg\min} \; ||CY - C\Phi\,\widetilde{A}_{\text{SC}}||_F \quad \text{tel que} \quad ||A, l||_0 \leq K \quad \forall l,$
7:
8: **Codebook Update** :
9: **for** $l \in [1 \dots n_D]$ **do**
10: $\omega_l \leftarrow \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l(i) \neq 0 \right\}$
11: $E \leftarrow Y - \Phi_{\backslash l} A^{\backslash l}$
12: $E_r \leftarrow E_{\omega_l}$
13: $U \Sigma V^T \leftarrow$ décomposition SVD de E_r
14: $A_r^l = \Sigma_{1,1} V^1$
15: $\boldsymbol{\phi}_l \leftarrow U_1$
16:
17: **Sortie** : Φ, A

La stratégie adoptée pour répondre au problème *Pb.Estimation* est la suivante :

Etape 1 : Création de la base SOBAL (hors-ligne)

1. Créer la séquence de *snapshots* Y à partir d'états utiles pour l'application considérée.
2. Utiliser l'algorithme SOBAL afin d'obtenir un dictionnaire Φ adapté à la configuration C des capteurs.

Etape 2 : Estimation (en-ligne)

1. Utiliser OMP (où un autre algorithme de reconstruction creuse à condition qu'il ait aussi été utilisé dans l'algorithme SOBAL) à partir des mesures $\boldsymbol{s}$ et du dictionnaire Φ afin d'obtenir la représentation creuse associée $\widehat{\boldsymbol{a}}$.
2. Estimer le champ à partir de la relation $\widehat{\boldsymbol{y}} = \Psi\,\widehat{\boldsymbol{a}}$.

3.2.3 Résultats

Les performances de cet algorithme sont maintenant illustrées et comparées à l'approche POD pour 2 configurations des capteurs. Une fois de plus, l'erreur relative ϵ_r définie dans le chapitre 2 est utilisée.

La figure 3.7 contient l'évolution de l'erreur obtenue à la fin de chaque cycle de l'algorithme pour une position donnée des capteurs ($\mathcal{C}_{\text{opt}_3}$ - $n_{\text{mul}} = 10$). Tout comme la figure 3.4, elle permet de se faire une idée du comportement de l'algorithme.

Positions $\mathcal{C}_{\text{f}}$

Les positions $\mathcal{C}_{\text{f}}$ présentées au chapitre précédent sont tout d'abord utilisées. Elles correspondent à un placement sans connaissance préalable du système. Les approches POD et SOBAL utilisent autant de modes effectifs K que le nombre de capteurs n_S.

La figure 3.8 et le tableau 3.4 contiennent les différentes erreurs obtenues. Comme il a déjà été vu au chapitre précédent, l'approche POD est incapable de fournir des résultats sa-

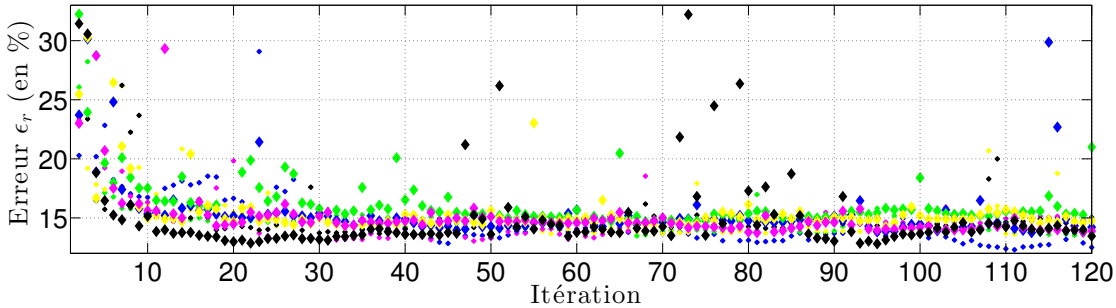

FIGURE 3.7 – SOBAL - Évolution de ϵ_r

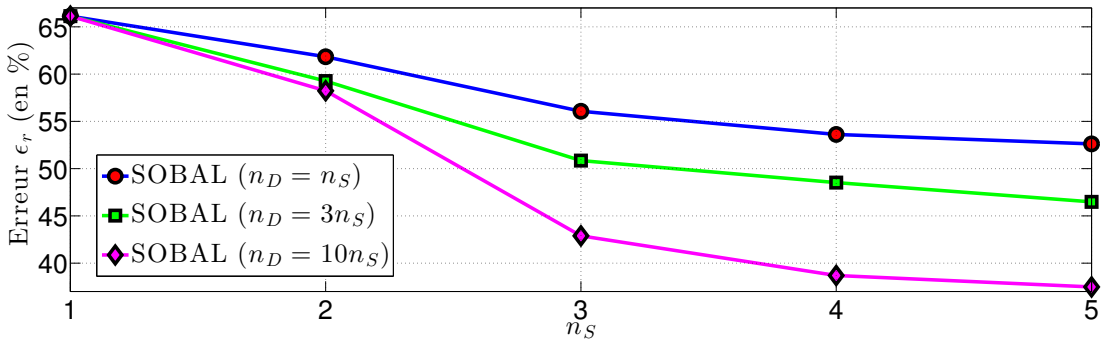

FIGURE 3.8 – SOBAL - Performances $\mathcal{C}_\mathrm{f}$

tisfaisants. Avec $n_S = 1$, l'erreur faite est d'environ 70% et avec $n_S = 5$, l'erreur est encore plus grande et vaut 220%. L'approche K-SVD classique est aussi insatisfaisante comme cela peut être vu à travers les résultats donnés dans le tableau 3.4. Par contre, l'approche SOBAL permet de garantir de bonnes performances d'estimation avec 66% d'erreur relative lorsque un capteur est utilisé et 40% lorsque cinq capteurs sont utilisés. Certes, ces performances ne sont pas exceptionnelles mais au moins, chaque nouveau capteur ajouté permet de réduire l'erreur. Le comportement notable est une fois de plus l'amélioration des performances lorsque la taille du dictionnaire SOBAL augmente. Augmenter n_mul permet de réduire considérablement l'erreur. Par exemple, lorsque 5 capteurs sont utilisés, ϵ_r qui valait 53% avec $n_\mathrm{mul} = 1$ (cas plein) ne vaut plus que 38% avec $n_\mathrm{mul} = 10$. L'approche SOBAL présente de meilleurs résultats que l'approche POD dans le cas plein. Ceci provient tout simplement de la difficulté de remonter jusqu'à la représentation lorsque si peu de capteurs (qui plus est, mal placés) sont utilisés. Un autre point d'intérêt est que certaines positions des capteurs font qu'il est plus facile d'obtenir un gain notable de performance en augmentant la taille du dictionnaire.

Nombre de capteurs n_S	POD $n_D = n_S$	K-SVD $n_D = 1n_S$	K-SVD $n_D = 3n_S$	K-SVD $n_D = 10n_S$	SOBAL $n_D = n_S$	SOBAL $n_D = 3n_S$	SOBAL $n_D = 10n_S$
1	72.0 %	72.0 %	107 %	105 %	66.1 %	66.1 %	66.1 %
2	138 %	138 %	161 %	94.0 %	61.8 %	59.3 %	58.2 %
3	109 %	109 %	83.4 %	77.4 %	56.1 %	50.8 %	42.9 %
4	103 %	103 %	76.9 %	64.7 %	53.6 %	48.5 %	38.7 %
5	219 %	219 %	84.2 %	66.8 %	52.6 %	46.5 %	37.5 %

TABLE 3.4 – SOBAL - Performances $\mathcal{C}_\mathrm{f}$

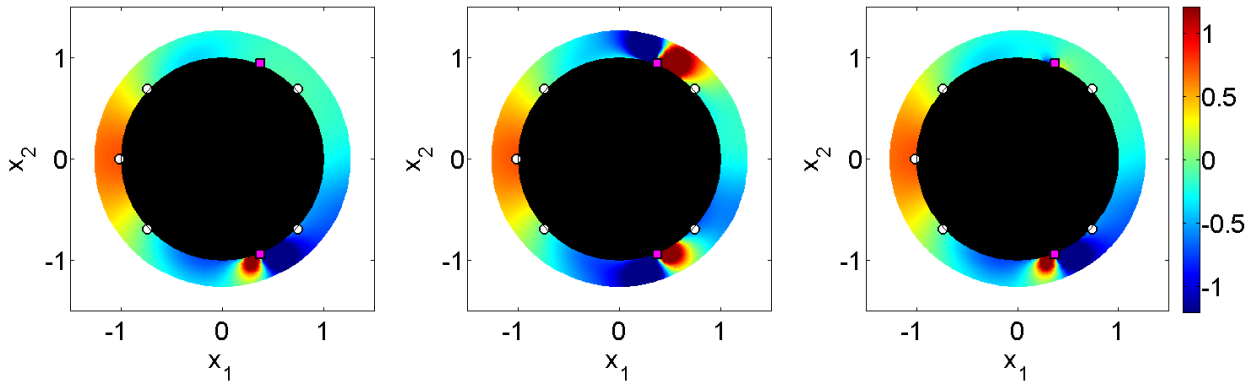

FIGURE 3.9 – Champ de pression relatif : exact (gauche), estimé à partir des modes POD (centre) et estimé avec la SOBAL (droite) - $n_S = 5$

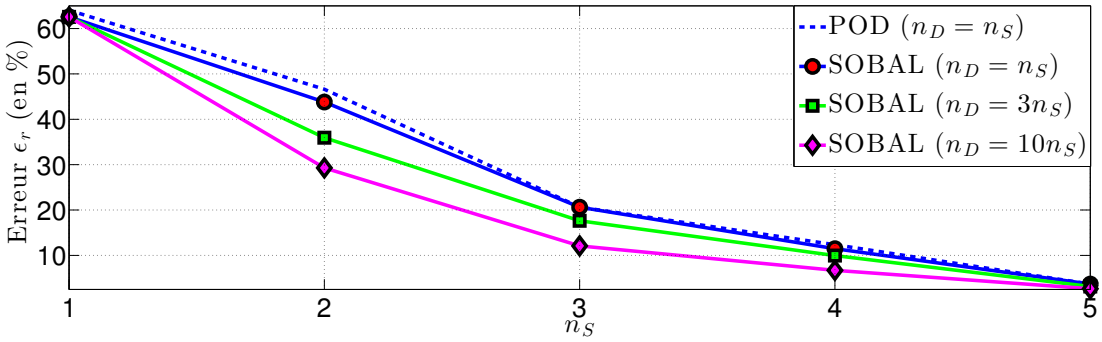

FIGURE 3.10 – SOBAL - Performances $\mathcal{C}_{\text{opt}}$

Pour illustrer ces performances, un *snapshot* précis est utilisé comme exemple dans la figure 3.9. Le champ de gauche correspond au *snapshot* de la séquence d'apprentissage. Le champ du milieu correspond à celui estimé par l'approche POD. L'erreur relative effectuée sur ce *snapshot* est de 275% (norme 2). Enfin, le champ de droite est associé à l'estimation K-SVD et l'erreur relative effectuée est de 7.0% seulement.

Positions $\mathcal{C}_{\text{opt}}$

Les positions $\mathcal{C}_{\text{opt}}$ des capteurs sont ici utilisées. La méthode employée pour sélectionner ces capteurs sera détaillée au chapitre 5. Les placements obtenus ne sont pas optimaux pour l'approche SOBAL mais permettent quand même de très bons résultats d'estimation. Il est en effet très difficile de trouver des capteurs qui vont pouvoir tirer parti de la nature creuse des représentations réduites recherchées (bien que ces positions existent 6.3.2).

Les résultats sont représentés sur la figure 3.10 et le tableau 3.5. Tout d'abord, l'approche POD s'avère bien plus performante que dans les configurations précédentes des capteurs. Pour chaque ajout de capteur, la configuration disponible permet de réduire l'erreur relative d'estimation. L'approche SOBAL permet d'atteindre de meilleurs résultats que l'approche POD, ce qui était l'effet recherché. Augmenter la taille de la base permet encore de réduire l'erreur effectuée sur les mêmes positions des capteurs. Par exemple, lorsque 4 capteurs sont utilisés, l'approche POD permet d'effectuer une erreur relative de $\epsilon_r = 12.3\%$ alors que l'approche SOBAL permet d'atteindre 6.70% avec $n_{\text{mul}} = 10$.

L'algorithme SOBAL, une simple modification de la K-SVD, peut ainsi être utilisée pour obtenir des dictionnaires performants en vue d'une estimation à partir de très peu de capteurs.

Nombre de capteurs n_S	**POD** $n_D = n_S$	**K-SVD** $n_D = n_S$	**K-SVD** $n_D = 3n_S$	**K-SVD** $n_D = 10n_S$	**SOBAL** $n_D = n_S$	**SOBAL** $n_D = 3n_S$	**SOBAL** $n_D = 10n_S$
1	64.0%	64.0%	72.2%	74.9%	62.6%	62.6%	62.6%
2	46.6%	46.6%	92.4%	152%	43.8%	35.9%	29.3%
3	20.6%	20.6%	38.7%	32.2%	20.6%	17.6%	12.1%
4	12.3%	12.3%	101%	13.8%	11.5%	9.96%	6.70%
5	3.68%	3.68%	76.1%	7.43%	3.68%	3.13%	2.65%

TABLE 3.5 – SOBAL - Performances $\mathcal{C}_{\text{opt}}$

3.3 Généralisation

3.3.1 Goal-Oriented - SOBAL généralisé

L'algorithme SOBAL vu jusqu'à présent a permis d'estimer un vecteur de grande dimension $\boldsymbol{y}$ à partir d'une restriction de ce dernier, les mesures $\boldsymbol{s} = C\boldsymbol{y}$. En utilisant ce dernier, la totalité des *snapshots* de Y ont été estimés à partir de $S = CY$.

Le nouvel objectif est maintenant d'**étendre le champ d'application** de l'approche SOBAL. Au lieu de voir cette approche comme manipulant uniquement la séquence Y, cette section considère les 2 séquences Y et S. Afin de ne pas introduire de confusions, une nouvelle notation est introduite :

$$H = \left(\boldsymbol{h}_{t_1} \dots \boldsymbol{h}_{t_{n_{\text{snap}}}}\right) \in \mathbb{R}^{n_{\boldsymbol{h}} \times n_{\text{snap}}}, \tag{3.28}$$

où $n_{\boldsymbol{h}} \in \mathbb{N}^*$ est la dimension d'un *snapshot* de ce champ $\boldsymbol{h}$. Pour alléger les notations, $\boldsymbol{h}_{t_i}$ est aussi noté $\boldsymbol{h}_i$. Le couple suivant de séquences d'entraînement est maintenant considéré :

— H, contenant les éléments du *champ à estimer*, de **grande dimension** et

— S, contenant les éléments du *champ mesuré*, de **faible dimension**.

Ces deux séquences doivent impérativement comporter le même nombre de *snapshots* et leur disposition dans la matrice d'entraînement est primordiale. En effet, à partir de l'élément $\boldsymbol{s}_{t_i}$, il faut obtenir une estimée de $\boldsymbol{h}_{t_i}$.

Ce nouveau problème, nommé *Pb.GoalOriented*, est énoncé ci-dessous :

<u>Données</u> : H, S

<u>Objectif</u> : Trouver une application $\Psi(.)$ définie sur $\mathbb{R}^{n_S} \to \mathbb{R}^{n_{\boldsymbol{h}}}$ qui minimise

$$\epsilon_r = ||H - \widehat{H}||_F / ||H||_F$$

$$\text{où } \widehat{H} = \left(\Psi(\boldsymbol{s}_{t_1}) \dots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}\right)\right).$$

Le champ d'application de ce problème est bien **plus vaste** que *Pb.Estimation*. Ce dernier en est même un cas particulier avec $H = Y$ et $S = CY$.

Les natures de H et S font que le problème va être plus ou moins délicat à résoudre. Une fois résolu, il permettra d'obtenir des estimateurs adaptés à des situations originales.

Par exemple, si S correspond aux mesures de pression et H contient le coefficient de traînée totale (un scalaire associé à chaque *snapshot*), l'estimateur obtenu incorporera une information délicate à obtenir sur la traînée due à la viscosité.

Un autre exemple consiste à insérer le champ des vitesses dans H afin de pouvoir estimer ce dernier à partir du champ de pression uniquement. Il est essentiel de noter que ceci est remarquable dans le cas actuel puisqu'il est impossible d'utiliser des capteurs de vitesse à la surface du cylindre (puisque la vitesse du fluide y est nulle).

Les motivations pour résoudre un tel problème sont nombreuses et l'objectif est ici d'adapter l'approche retenue avec SOBAL afin d'obtenir un algorithme aussi performant

dans ce nouveau contexte.

Lorsque le problème *Pb.Estimation* est vu à travers le cadre de *Pb.GoalOriented*, un couplage apparaît entre H et S. En effet, les deux matrices d'entraînements sont reliées selon :

$$S = CH. \tag{3.29}$$

L'idée résultante est alors d'appliquer l'algorithme SOBAL pour répondre au problème *Pb.GoalOriented* lorsqu'il existe une matrice $J \in \mathbb{R}^{n_S \times n_h}$ telle que $S = JH$ ou $||S - JH||_F$ est *faible*. Ce problème n'est pas traité ici dans sa totalité, seule cette restriction particulière est pour l'instant retenue. Si **un lien linéaire matriciel** existe entre H et S, l'algorithme SOBAL peut alors être utilisé sans modification majeure pour une nouvelle catégorie de problèmes.

Soit Φ le dictionnaire produit par cet algorithme. L'étape de SC manipule ainsi les normes $||\boldsymbol{s} - J\Phi\boldsymbol{a}||_2$. L'étape de CU manipule la norme $||H - \Phi A||_F$. Le pseudo-code de l'algorithme SOBAL généralisé se trouve dans l'algorithme 4.

Algorithm 4 SOBAL Généralisé Pseudo-Code

Require: S, H, J, n_D, K, D_{ini}, n_{it}

1: **Initialisation** :
2: $\Phi \leftarrow D_{\text{ini}}$
3:
4: **for** $j \in [1 \dots n_{\texttt{it}}]$ **do**
5: **Sensor Oriented Sparse Coding** :
6: $A \leftarrow \underset{\widetilde{A}_{\text{SC}} \in \mathbb{R}^{n_D \times n_{\text{snap}}}}{\arg\min} ||S - J\Phi\,\widetilde{A}_{\text{SC}}||_F$ tel que $||A, l||_0 \leq K \quad \forall l,$
7:
8: **Codebook Update** :
9: **for** $l \in [1 \dots n_D]$ **do**
10: $\omega_l \leftarrow \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l(i) \neq 0 \right\}$
11: $E \leftarrow H - \Phi_{\backslash l} A^{\backslash l}$
12: $E_r \leftarrow E_{\omega_l}$
13: $U\Sigma V^T \leftarrow$ décomposition SVD de E_r
14: $A_r^l = \Sigma_{1,1} V^1$
15: $\boldsymbol{\phi}_l \leftarrow U_1$
16:
17: **Sortie** : Φ, A

Un exemple d'utilisation pratique de cette approche est donné dans le chapitre 6 où une mémorisation des mesures passées est exploitée afin d'améliorer l'erreur d'estimation sur le champ de pression. Des méthodes d'obtention de J seront aussi illustrées.

Pour aller plus loin dans la résolution de *Pb.GoalOriented*, il est nécessaire de considérer des séquences H et S quelconques. Ceci va se faire en découplant *les* dictionnaires de l'algorithme SOBAL : Φ et $C\Phi$ (ou $J\Phi$ avec l'approche généralisée).

3.3.2 Tentative de découplage

L'objectif est une fois de plus d'étendre l'algorithme SOBAL afin de répondre au problème *Pb.GoalOriented*. Une telle extension de l'algorithme SOBAL permet de capitaliser sur l'utilisation de la parcimonie et le fait qu'en pratique, seule S est disponible. En

d'autres termes, la parcimonie permet de se défaire de la limitation sur le nombre de modes utilisés dans un schéma classique, et l'utilisation d'une étape *Sensor Oriented Sparse Coding* permet de garantir un bon fonctionnement pratique. *Pb.GoalOriented* est considéré dans le cadre général où une relation matricielle simple entre les deux séquences S et H est inconnue ou, tout simplement, impossible à obtenir vu la nature des séquences en question.

L'approche retenue est d'utiliser deux dictionnaires $D_1 \in \mathbb{R}^{n_S \times n_D}$ et $D_2 \in \mathbb{R}^{n_{\boldsymbol{h}} \times n_D}$:
— D_1 manipule directement $\boldsymbol{s}$ et permet d'**obtenir une représentation creuse** $\boldsymbol{a}$,
— D_2 agit directement sur $\boldsymbol{a}$ et permet d'**estimer** $\boldsymbol{h}$.

Ainsi, l'espace réduit (nommé de cette manière puisque les vecteurs de cet espace ne sont que de taille n_D), où "vit" $\boldsymbol{a}$, permet de faire le lien entre les séquences S et H. De par les nouveaux degrés de libertés disponibles (deux dictionnaires au lieu d'un unique), une telle approche peut possiblement faire mieux que SOBAL pour répondre à *Pb.Estimation*.

Dans ce cadre, l'algorithme SOBAL est vu comme manipulant deux dictionnaires couplés. En effet, la résolution du problème *Pb.Estimation* fait intervenir D_1 et D_2 tels que $D_1 = JD_2$. C'est pour cela que cette nouvelle approche est qualifiée de découplée, par opposition à la vision dictionnaires couplés de l'algorithme SOBAL.

Soit GOBAL, le nouvel algorithme basé sur SOBAL, présenté pour résoudre ce problème. L'approche la plus directe consiste à exploiter les deux étapes clefs de l'algorithme K-SVD : le SC exploitant D_1 et le CU exploitant D_2. Cependant, il apparaît maintenant nécessaire d'introduire une **troisième étape** afin de mettre à jour D_1 (cette mise à jour n'étant plus faite implicitement via D_2). Ci-dessous, une première ébauche d'un cycle de GOBAL est proposée.

La première étape, ***Sensor-Oriented Sparse Coding***, permet de répondre au problème :

$$\boldsymbol{a} \in \arg\min_{\widetilde{\boldsymbol{a}}} ||\boldsymbol{s} - D_1\widetilde{\boldsymbol{a}}||_2 \text{ tel que } ||\boldsymbol{a}||_0 = K, \tag{3.30}$$

pour tout *snapshot*. La matrice A est ainsi formée par la concaténation des $\boldsymbol{a}$.

La seconde étape, mettant à jour D_2, est renommée ***Estimation Codebook Update*** puisque l'objectif de D_2 est de reconstruire H. La mise à jour de D_2 cherche à répondre au problème suivant :

$$\{D_2, A\} \in \arg\min_{\widetilde{D_2},\widetilde{A}} ||H - \widetilde{D_2}\widetilde{A}||_F. \tag{3.31}$$

La résolution de ce problème peut être faite par l'approche *Codebook Update* classique. Chaque paire $\left(\boldsymbol{d}_{2l}, A^l\right)$ est traitée dans un ordre aléatoire. La matrice erreur E_H est formée selon :

$$E_H = H - D_{2\backslash l}A^{\backslash l}. \tag{3.32}$$

A est initialisée avec la valeur donnée par l'étape *Sensor-Oriented Sparse Coding*. Si un mode de D_1 n'est pas utilisé, la ligne associée de A sera nulle et, de ce fait, le mode associé de D_2 sera aussi non utilisé. Le système de remplacement de mode non utilisé doit agir sur D_1 et non sur D_2. La méthode la plus directe consiste à trouver le *snapshot* de H le moins bien estimé avec les dictionnaires actuels puis d'injecter le *snapshot* de S correspondant à la place du mode non exploité de D_1.

La troisième et dernière étape agissant sur D_1 est nommée ***Feature Codebook Update*** puisque l'objectif de D_1 est aussi d'extraire les caractéristiques importantes des mesures

S. La mise à jour de D_1 cherche à répondre au problème suivant :

$$D_1 \in \arg\min_{\widetilde{D_1}} ||S - \widetilde{D_1}A||_F, \tag{3.33}$$

où A est obtenue par l'étape précédente, l'*Estimation Codebook Update*. Une étape CU classique ne peut pas être utilisée car il ne faut pas mettre à jour A, obtenue à l'étape précédente. La résolution de ce problème peut se faire d'une manière itérative exploitant aussi la nature creuse de A. Tout comme la méthode *Codebook Update*, une matrice erreur E_S est formée pour chaque couple $\left(\boldsymbol{d}_{1l}, A^l\right)$ avec $1 \leq l \leq n_D$. Cette dernière est définie par

$$E_S = S - D_{1\setminus l}A^{\setminus l}. \tag{3.34}$$

Afin de minimiser l'erreur $||S - D_1A||_F$, il faut minimiser $||E_S - \boldsymbol{d}_{1l}A^l||_F$. Les indices des *snapshots* qui utilisent le mode $\boldsymbol{d}_{1l}$ sont contenus dans ω_l, définie selon :

$$\omega_l \leftarrow \left\{i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l(i) \neq 0\right\}. \tag{3.35}$$

Si ω_l est vide, A^l correspond au vecteur nul et dans ce cas, agir sur $\boldsymbol{d}_{1l}$ n'a aucune conséquence sur la norme à minimiser. La matrice $E_{S,r}$ est ensuite définie comme la restriction de E_S aux colonnes (*snapshots*) indexées par ω_l. Au lieu de chercher la meilleure matrice de rang 1 décrivant $E_{S,r}$, le meilleur vecteur $\boldsymbol{d}_{1l}$ est recherché sachant que A^l est imposé. La solution de ce problème est, une fois de plus, donnée par la pseudo-inverse. La restriction de A^l aux *snapshots* indexés par ω_l est notée A^l_r. La pseudo-inverse de A^l_r est donnée par

$$A^{l\,+}_r = A^{l\,T}_r \left(A^l_r A^{l\,T}_r\right). \tag{3.36}$$

La mise à jour de cet élément de D_1 peut se faire par :

$$\boldsymbol{d}_{1l} = E_{S,r}A^{l\,+}_r. \tag{3.37}$$

Afin d'obtenir le meilleur dictionnaire D_1, il faut effectuer de nombreuses mises à jour successives de ces colonnes.

Pour résumer, l'étape de SC donne la meilleure représentation creuse de S via le dictionnaire D_1. L'étape suivante a pour objectif de modifier cette représentation creuse en maintenant son support afin de décrire au mieux H. Chaque mode de D_2 n'est traité qu'une fois. Pendant cette étape, D_2 et A sont mis à jour. Ensuite, la troisième et dernière étape consiste à modifier D_1 afin de promouvoir l'obtention du dernier A lors de la prochaine étape de SC.

Malheureusement, ce problème est bien plus *greedy* que SOBAL et fait qu'une telle approche présente fréquemment des problèmes de convergence suivant les séquences S et H retenues. En pratique, un tel algorithme ne converge que très difficilement sur des séquences S et H quelconques et "fonctionne" pour répondre au problème *Pb.Estimation*. La seconde étape réduit bien l'erreur faite sur H mais la modification faite sur A n'est pas forcement atteignable en modifiant D_1. Ceci engendre un nouveau A moins performant que l'ancien et l'étape *Estimation Codebook Update* peut une fois de plus proposer un A inexploitable par le *Feature Codebook Update*.

Ce problème de convergence n'a pas pu être résolu par :

— une modification d'un nombre limité de modes par cycle,

— des lancers successifs de l'étape *Codebook Update*.

Bien que ces modifications réduisent les variations brutales de l'erreur ϵ_r, elles se sont avérées insuffisantes.

3.3.3 GOBAL

Un nouveau GOBAL va donc être proposé. Ce dernier maintient la même philosophie que celle présentée ci-dessus : *Sparse Coding*, mise à jour de D_2 puis mise à jour de D_1. Cependant, 3 grandes modifications vont être faites afin de rendre l'algorithme moins *greedy* :

— la modification de D_1 devient le cœur de l'algorithme et la mise à jour de D_2 est traitée d'une manière optimale ne laissant plus place au hasard,
— un système de conditions initiales incrémentales est mis en place afin de supprimer toute forme de divergence du critère $||H - D_2A||_F$,
— une méthode de mise à jour progressive des modes de D_1 exploitant à chaque étape D_2 est utilisée.

Une dernière modification moins majeure concerne la mise à jour des modes non-utilisés.

Les paragraphes qui suivent détaillent ce nouvel algorithme GOBAL. Le pseudo-code de ce dernier se trouve dans l'algorithme 5.

Sensor Oriented Sparse Coding

L'étape *Sensor Oriented Sparse Coding* est laissée telle quelle. Elle permet, à elle-seule, d'introduire la notion de parcimonie dans l'algorithme. Ainsi, à partir de l'état actuel de D_1, une représentation creuse de S est calculée et mise sous la forme A. Cette représentation réduite, accessible en pratique, doit permettre d'estimer convenablement H.

Estimation Codebook Update

Au lieu de modifier D_2 et A à partir d'une étape *Codebook Update* classique, seul D_2 va ici être traité. A n'est plus modifié et ceci va imposer l'introduction d'une nouvelle méthode de mise à jour de D_1. D_2 doit permettre de minimiser $||H - D_2A||_F$ où A et H sont fixes.

La solution de ce problème est une fois de plus donnée par la pseudo-inverse de Moore-Penrose avec

$$D_2 = H\,(A)^+\,. \tag{3.38}$$

Jusqu'à maintenant, l'algorithme *pinv* de MATLAB a été utilisé pour calculer les pseudo-inverses. L'algorithme **qrginv** est plus rapide que *pinv* et est adapté aux cas des matrices creuses. Proposé par Katsikis, Pappas et Petralias en 2011 [33], **qrginv** se base sur une décomposition QR pour calculer cette pseudo-inverse. La pseudo-inverse classique d'une troncature adaptée de la matrice "R" (résultante de cette décomposition QR) est ensuite calculée.

Une telle approche pourrait être utilisée dans l'algorithme MOD [21]. Cependant, il ne faut pas oublier que l'étape *Codebook Update* de la K-SVD (et de SOBAL) permet un brassage des solutions, et ainsi l'obtention d'un meilleur minimum local.

L'algorithme *qrginv* [33] est utilisé pour effectuer l'*Estimation Codebook Update*.

Sauvegarde d'une solution acceptable

Il faut modifier D_1 afin que l'étape de SC génère un A qui à son tour engendre un meilleur D_2 (minimisant encore plus, par rapport au cycle précédent, $||H - D_2A||_F$). Ce n'est plus un D_2 non-optimal qui va être à la source de la mise à jour de D_1 vu l'approche utilisée ci-dessus (pseudo-inverse ou une succession de *Codebook Update* de type

Algorithm 5 GOBAL Pseudo-Code

Require: S, H, n_D, K, D_{ini1}, n_{it}

1: **Initialisation** :
2: $D_1 \leftarrow D_{\text{ini}}, \quad n_{\text{up}} \leftarrow n_D, \quad \epsilon_{\text{best}} \leftarrow 1e50$
3:
4: **for** $j \in [1 \dots n_{\texttt{it}}]$ **do**
5:
6: **Sensor Oriented Sparse Coding** :
7: $A \leftarrow \underset{\widetilde{A}_{\text{SC}} \in \mathbb{R}^{n_D \times n_{\text{snap}}}}{\arg\min} ||S - D_1 \widetilde{A}_{\text{SC}}||_F \quad \text{tel que} \quad ||A, l||_0 \leq K \quad \forall l,$
8:
9: **Estimation Codebook Update** :
10: $D_2 \leftarrow H\,(A)^+$ (utilisation de qrginv [33])
11:
12: **Sauvegarde** :
13: $\epsilon \leftarrow ||H - D_2 A||_F$
14: **if** $\epsilon < \epsilon_{\text{best}}$ **then**
15: $\epsilon_{\text{best}} \leftarrow \epsilon$
16: $D_{1\text{best}} \leftarrow D_1, \quad D_{2\text{best}} \leftarrow D_2, \quad A_{\text{best}} \leftarrow A$
17: **else**
18: $D_1 \leftarrow D_{1\text{best}}, \quad D_2 \leftarrow D_{2\text{best}}, \quad A \leftarrow A_{\text{best}}$
19: **Procédure de mise à jour de $n_{\texttt{up}}$**
20:
21: **Feature Codebook Update** :
22: **for** $l \in [1 \dots n_{\texttt{up}}]$ **do**
23: $\omega_l \leftarrow \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\, A^l(i) \neq 0 \right\}$
24: **if** ω_l est vide **then**
25: **Procédure de remplacement**
26: **else**
27: $E_H \leftarrow H - D_{2\backslash l} A^{\backslash l}$
28: $U \Sigma V^T \leftarrow$ décomposition SVD de E_{Hr}
29: $A_r^l = \Sigma_{1,1} V^1$
30:
31: $E_S \leftarrow S - D_{1\backslash l} A^{\backslash l}$
32: $\boldsymbol{d}_{tl} \leftarrow E_{Sr} \left(A_r^l \right)^+$
33: $D_1 \leftarrow$ les vecteurs temporaires $\boldsymbol{d}_t$ précédents
34:
35: **Sortie** : D_1, D_2, A

K-SVD). Bien que cette précaution **facilite grandement la convergence**, il faut ajouter une **étape de sauvegarde** avant d'effectuer cette mise à jour pour **se prémunir d'une mauvaise modification** de D_1.

Des modifications successives de D_1 peuvent malheureusement conduire à une erreur sur H de plus en plus grande, ou *qui ne diminue pas suffisamment*. Ceci provient de la difficulté de proposer une bonne modification des colonnes de D_1. La totalité des colonnes de D_1 participe à l'élaboration de A et sans calcul du D_2 optimal associé, il est délicat de prévoir l'impact d'un tel A sur une estimation convenable de H.

Étape 1 - Modification des conditions initiales
Une fois le dictionnaire D_2 optimal calculé, la performance pratique de l'état actuel de la solution (D_1, D_2) est donnée par $||H - D_2A||_F$. Si cette erreur est plus faible que la meilleure erreur ϵ_{best} obtenue jusqu'à présent, l'état actuel de D_1, D_2 et A est sauvegardé dans les matrices $D_{1\text{best}}, D_{2\text{best}}$ et A_{best}. Ces dernières constituent les nouvelles conditions initiales de l'algorithme. Si d'un cycle à l'autre l'erreur augmente, l'état actuel des dictionnaires et de la matrice de représentations réduites est réinitialisé à partir de la meilleure solution obtenue jusqu'à présent.

Une telle étape n'était pas nécessaire avec SOBAL. Au contraire, elle pourrait nuire à la diversité des solutions proposées et à la capacité de se défaire d'un minimum local du critère considéré. Pour GOBAL, elle permet de faciliter l'obtention de la bonne solution vu la difficulté d'obtenir un D_1 satisfaisant.

Étape 2 - Procédure de mise à jour de n_{up}
Il faut prêter une attention particulière au **nombre de modes** de D_1 **mis à jour** lors d'un cycle. Modifier la *totalité* des modes lors d'un cycle est utile au début de la "vie" de l'algorithme. Ceci permet de faire des grandes modifications et de réduire de manière importante le critère.

Cependant, pour diminuer le critère davantage, il est maintenant nécessaire d'agir que sur un nombre limité de modes. **Ce nombre est réduit lorsqu'une minimisation du meilleur critère obtenu n'a pas eu lieu au bout d'un certain nombre de cycles**.

Ceci provient du fait qu'il est difficile de prévoir l'effet de la modification d'un mode de D_1 sur A et ainsi, sur le critère. Lorsque peu de modes sont mis à jour, la possibilité qu'un de ces nouveaux modes perturbe les autres devient plus faible (d'après les différentes expériences réalisées). Cependant, modifier peu de modes fait que le critère ne peut pas grandement diminuer d'un cycle à l'autre. Le système de mise à jour des conditions initiales vu ci-dessus permet d'exploiter au mieux ce comportement. Étant donné que le point de départ d'un cycle est toujours la meilleure solution obtenue lors du déroulement de l'algorithme, il n'y a aucun risque de diminuer le nombre de modes lorsque le critère possède une erreur *trop grande*. Ensuite, si un certain nombre de cycles sont complétés sans diminution du critère, le nombre de modes de D_1 mis à jour est diminué. En d'autre termes, **une approche de plus en plus fine est utilisée**. Les modes mis à jour sont choisis aléatoirement.

Mise à jour de D_1

Il ne reste plus qu'à détailler la méthode de calcul des modes de D_1. Elle est très proche des deux *Codebook Update* précédents. Le triplet $\left(\boldsymbol{d}_{1l}, \boldsymbol{d}_{2l}, A^l\right)$ relatif au mode $1 \leq l \leq n_D$

est étudié. Pour diminuer l'erreur $||H - D_2 A||_F$ en agissant sur le mode l, il faut trouver un meilleur couple $\left(\boldsymbol{d}_{2l}, A^l\right)$.

Ceci est réalisé comme une étape *Estimation Codebook Update* avec le calcul d'une SVD réduite, permettant de trouver la meilleure matrice de rang 1 approchant la matrice erreur E_r relative au mode l et à H. Cette dernière porte l'indice r puisque seuls les *snapshots* qui utilisent réellement le mode l sont retenus. Cette matrice de rang 1 est exprimée sous la forme $\boldsymbol{u}\sigma\boldsymbol{v}^T$. A est ensuite mise à jour selon :

$$A_r^l \leftarrow \sigma \boldsymbol{v}^T. \tag{3.39}$$

où A_r^l est la restriction de A^l aux *snapshots* qui utilisent réellement le mode l.

Il peut paraître sous-optimal de mettre à jour A^l de cette manière puisque cela revient à considérer un dictionnaire D_2 différent de celui calculé à l'étape précédente (puisqu'une nouvelle SVD est effectuée). Si A_r^l est mise à jour selon $A_r^l \leftarrow (D_{2l})^+ E_r$, il s'avère que l'algorithme converge trop rapidement vers un minimum local. La difficulté avec les algorithmes *greedy* est de réussir à trouver un minimum local qui va être proche de la solution globale recherchée. Plusieurs méthodes peuvent être utilisées pour ajouter de la variabilité dans l'algorithme GOBAL, évitant ainsi une convergence trop rapide et définitive vers un minimum local non désiré. Cette méthode rustique donne de bons résultats en pratique.

Ensuite, $\boldsymbol{d}_{1l}$ doit être modifié afin d'espérer rendre la nouvelle valeur de A^l plus accessible par l'étape SC. Ceci est fait par l'approche *Feature Codebook Update* présentée dans l'approche initiale (3.3.2). Soit E_S la matrice erreur relative au mode l et à la matrice S définie par :

$$E_S = S - D_{1\setminus l} A^{\setminus l}. \tag{3.40}$$

La restriction de cette dernière aux colonnes indexées par ω_l est encore notée $E_{S,r}$. La restriction de A^l aux lignes correspondantes est $A_r^{\setminus l}$. Le vecteur $\boldsymbol{d}_{1l}$ désiré est donné par :

$$\boldsymbol{d}_{1l} \leftarrow E_{S,r} \left(A_r^{\setminus l}\right)^+. \tag{3.41}$$

En pratique, on constate qu'il est plus intéressant de **ne pas mettre à jour** D_1 en cours de cycle. Un vecteur **temporaire** $\boldsymbol{d}_{tl}$ est utilisé pour stocker la valeur désirée de $\boldsymbol{d}_{1l}$. Une fois le cycle terminé (les modes sélectionnés ont été mis à jour), les colonnes associées de D_1 sont mises à jour à partir des vecteurs temporaires. Cette précaution permet d'éviter que la mise à jour d'un mode agisse négativement sur les autres via la matrice E_S qui sera recalculée pour le mode suivant.

Cependant, *le gain alors effectué est très faible.* D_1 peut être mis à jour à volonté sans que cela pose de problème majeur (selon les essais effectués).

Procédure de remplacement

Avant d'illustrer les performances de cet algorithme, une nouvelle méthode de **mise à jour de modes non-utilisés** est décrite. Elle est utilisée conjointement avec la méthode de mise à jour de D_1.

Lors de la mise à jour des modes, la méthode de remplacement est appliquée à la place de la méthode de mise à jour classique si **le mode en question est non-utilisé**. Ceci a principalement lieu lors des premiers cycles de l'algorithme où certains modes ne sont pas

utilisés.

En pratique, il s'est confirmé que si plusieurs modes ne sont pas utilisés (ce qui est un cas courant lorsque n_{mul} est grand devant K), il faut les traiter **conjointement**. Une mise à jour des modes non-utilisés *un par un* peut ne jamais résoudre le problème. Un mode est remplacé et au cycle précédent, un autre se retrouve non utilisé. Ce phénomène peut ensuite se répéter tout au long du déroulement de l'algorithme et ainsi produire des dictionnaires qui ne seront pas utilisés dans leur totalité. Pour éviter ce genre de boucle, une part d'aléatoire est aussi rajoutée.

Soit l l'index du mode couramment étudié. L'ensemble $\omega_l = \left\{ i \,|\, 1 \leq i \leq n_{\text{snap}},\ A^l(i) \neq 0 \right\}$ est ensuite calculé. Si ce dernier est vide, la procédure de remplacement est appliquée, sinon, la méthode de mise à jour du mode $\boldsymbol{d}_{1l}$ est appliquée.

Description de la procédure :
La première étape est de calculer les ensembles ω_l pour $1 \leq l \leq n_D$. Les index des ensembles ω_l vides sont ensuite regroupés pour former l'ensemble $T_\varnothing$. Cet ensemble contient $n_\varnothing \in \mathbb{N}^*$ éléments.

Dans ce cas :
- $A_\varnothing^T$, la matrice contenant les lignes de A indexées par $T_\varnothing$, est une matrice nulle de dimensions $n_\varnothing \times n_D$,
- $D_{1T_\varnothing}$, la matrice contenant les colonnes de D_1 indexées par $T_\varnothing$, est une matrice nulle de dimensions $n_S \times n_\varnothing$,
- $D_{2T_\varnothing}$, définie à partir de D_2 comme la matrice $D_{1T_\varnothing}$, est une matrice nulle de dimensions $n_{\boldsymbol{x}} \times n_\varnothing$.

Il faut proposer une matrice $D_{1T_\varnothing}$ qui fera en sorte que les modes $T_\varnothing$ soient utilisés au cycle suivant (ou, au moins, que le nombre de modes non utilisés diminue).

L'approche de remplacement retenue est très similaire à l'approche de mise à jour de D_1. La différence majeure est que la parcimonie du nouveau $A_\varnothing^T$ n'est plus imposée, afin de faciliter le **brassage des solutions**. Vu qu'un ou plusieurs modes ne sont pas utilisés, imposer une modification plus agressive de D_1 semble nécessaire.

La matrice erreur E_H associée à H et aux modes $T_\varnothing$ est calculée. Cette dernière est donnée par :

$$E_H = H - D_{2\setminus T_\varnothing} A^{\setminus T_\varnothing} = H - \sum_{i \notin T_\varnothing} \boldsymbol{d}_{2i} A^i. \tag{3.42}$$

Ensuite, bien qu'effectuer une restriction de E_H ne soit pas obligatoire, elle permet de réduire le temps de calcul et d'**ajouter de l'aléatoire** dans la méthode de remplacement. Ce faisant, d'un cycle à l'autre, des propositions de $D_{1T_\varnothing}$ peuvent être différentes, débloquant ainsi une situation délicate. Seules certaines colonnes de E_H sont considérées. Ces dernières sont indexées par l'ensemble T_ϵ et correspondent aux index des $n_\epsilon \in \mathbb{N}^*$ *snapshots* de H **les plus mal représentés**. L'erreur associée à un *snapshots* $l \in \mathbb{N}^*$ est donnée par $||\boldsymbol{h}_l - D_2 A_l||_2$. D'un cycle à l'autre, n_ϵ est choisi de manière aléatoire. Soit $E_{H,r}$ la restriction de E_H aux colonnes indexées par T_ϵ.

Une décomposition SVD réduite de $E_{H,r}$ permet ensuite d'obtenir la meilleure matrice de rang $n_\varnothing$ décrivant $E_{H,r}$. Cette décomposition est notée $U\Sigma V^T$ où $U \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_\varnothing}$, $\Sigma \in \mathbb{R}^{n_\varnothing \times n_\varnothing}$ et $V \in \mathbb{R}^{n_\epsilon \times n_\varnothing}$. Il devient clair qu'une nouvelle limitation apparaît de par le rang de $E_{H,r}$. Pour que cette mission puisse marcher, il faut que le rang de $E_{H,r}$ soit

supérieur ou égal à $n_\varnothing$. Si ce n'est pas le cas, seule une partie (de taille égale au rang de $E_{H,r}$) des modes non utilisés peut être mise à jour.

ΣV^T contient la représentation (pleine) des *snapshots* T_ϵ associés aux modes $T_\varnothing$. Elle est à l'origine de la modification des colonnes de D_1 recherchée. Pour effectuer cette dernière, la matrice erreur E_S associée à S et aux mode $T_\varnothing$ est calculée selon :

$$E_S = S - D_{1\setminus T_\varnothing} A^{\setminus T_\varnothing} = S - \sum_{i \notin T_\varnothing} \boldsymbol{d}_{1i} A^i. \tag{3.43}$$

Ensuite, la restriction de cette dernière aux colonnes indexées par T_ϵ est notée $E_{S,r}$. L'objectif est ensuite de trouver la matrice $\widetilde{D_{1T_\varnothing}}$ minimisant la norme $||E_{S,r} - \widetilde{D_{1T_\varnothing}} \Sigma V^T||_F$. Cette dernière est donnée par :

$$\widetilde{D_{1T_\varnothing}} = E_{S,r} \left(\Sigma V^T\right)^+, \tag{3.44}$$

Les modes de D_1 non utilisés sont enfin mis à jour selon :

$$D_{1T_\varnothing} \leftarrow E_{S,r} \left(\Sigma V^T\right)^+. \tag{3.45}$$

A partir de ce nouveau D_1, la représentation A est calculée à partir de l'étape *Sparse Coding*. D_2 est ensuite mis à jour à partir de A, comme lors de la première phase. L'erreur $||H - D_2 A||_F$ est ensuite calculée. Si cette dernière est plus faible que ϵ_{best}, les états actuels de D_1, D_2 et A sont stockés dans $D_{1\text{best}}, D_{2\text{best}}$ et A_{best} respectivement.

En pratique, lorsque cette procédure est utilisée, la totalité des modes sont correctement remplacés et permettent d'**exploiter pleinement le potentiel des dictionnaires** D_1 et D_2.

3.3.4 Résultats

Pour conclure sur GOBAL, quelques résultats illustrant cette approche vont être étudiés. Les réglages effectués sur l'algorithme GOBAL vont aussi être explicités afin de faciliter au mieux l'utilisation pratique de ce dernier. Pour pouvoir comparer cet algorithme avec SOBAL, les positions $\mathcal{C}_\text{f}$ et $\mathcal{C}_\text{opt}$ vues dans le chapitre 2 sont utilisées. Une fois de plus, les configurations relatives de $n_S = 1$ à $n_S = 5$ capteurs sont utilisées. La parcimonie de la représentation réduite est $K = n_S$. A chaque instant, autant de modes que de capteurs sont utilisés pour décrire ces mesures. Ensuite, un multiplicateur $n_{\text{mul}} \in \mathbb{N}^*$ permet de définir le nombre de modes n_D des deux dictionnaires D_1 et D_2. Trois multiplicateurs sont ici considérés : 1 (représentation pleine), 3 et 10. Ces situations font que les performances de GOBAL sont directement comparables à celles de SOBAL.

Vu que le problème *Pb.Estimation* est étudié, les matrices d'entraînement sont $H = Y$ et $S = CY$.

Soit $n_{\text{up}} \in \mathbb{N}^*$ le nombre de modes mis à jour lors d'un cycle. La méthode de réduction du nombre de n_{up} est définie selon un seuil.

Si $n_{\text{up}} > K$, une diminution *rapide* de n_{up} est retenue selon :

$$n_{\text{up}} \leftarrow \left\lceil \frac{3}{4} n_{\text{up}} \right\rceil, \tag{3.46}$$

où $\lceil \boldsymbol{w} \rceil$, $\boldsymbol{w} \in \mathbb{R}$, correspond au plus petit entier supérieur à $\boldsymbol{w}$.

Ensuite, lorsque $n_{\text{up}} \leq K$, une diminution *lente* est retenue à travers l'opération

$$n_{\text{up}} \leftarrow n_{\text{up}} - 1. \tag{3.47}$$

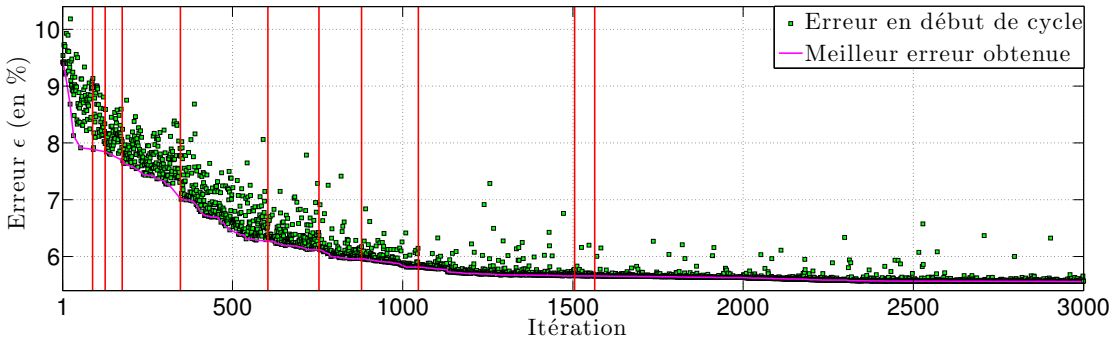

FIGURE 3.11 – GOBAL - Évolution de ϵ_r

Enfin, l'initialisation des paramètres de l'algorithme est faite de la manière suivante. Afin d'agir sur la totalité des modes au départ de l'algorithme, $n_{\text{up}} = n_D$. Ensuite, la seule initialisation nécessaire concerne le dictionnaire D_1. D_2 étant en permanence calculé de manière directe. D_1 est initialisé à partir du dictionnaire D_{ini}. Ce dernier est formé aléatoirement selon une distribution gaussienne réduite (centrée et de variance unité) puis normalisée selon les colonnes. Elle s'est manifestée efficace en pratique. De même, une initialisation à partir de *snapshots* de S est aussi possible.

Sur la figure 3.11, l'évolution du critère en fonction de l'index du cycle lors d'un lancer de l'algorithme est représentée. Ce lancer type est associé aux positions $\mathcal{C}_{\text{opt}_4}$ avec $n_{\text{mul}} = 10$. Le nuage de point représente l'erreur obtenue en début de cycle, juste après le calcul du meilleur D_2. La courbe en trait plein représente l'évolution de la meilleure solution obtenue jusqu'à cette itération. Enfin, les traits verticaux indiquent une réduction du nombre de modes mis à jour n_{up}.

Cette figure permet de se faire une idée plus claire du déroulement de l'algorithme. Cet algorithme converge par construction vers un minimum local.

L'erreur diminue fortement au démarrage de l'algorithme. Pendant ces premières itérations, il est très probable que certains modes ne sont pas utilisés. C'est principalement à ce stade qu'interviennent les méthodes de remplacement de modes.

Au fur et à mesure que n_{up} est réduit, le nuage de points devient moins épais, illustrant une variabilité plus faible de l'erreur de reconstruction.

La figure 3.12 présente les résultats de GOBAL dans le cas des capteurs $\mathcal{C}_{\text{f}}$, sous-optimalement placés. Pour chaque position des capteurs, plusieurs tailles de dictionnaires n_D sont considérées (voir la légende). Bien que cela surcharge un peu la figure, les résultats correspondant à SOBAL sont aussi représentés.

Tout d'abord, dans une configuration donnée des capteurs avec $n_S > 1$ et $n_{\text{mul}} > 1$, GOBAL fait toujours mieux que SOBAL pour une taille de dictionnaire et une parcimonie identiques. Tout comme SOBAL, GOBAL est capable de s'adapter à des capteurs mal placés. Découpler les dictionnaires, et ainsi rajouter plus de degrés de libertés, a permis de réduire l'erreur d'estimation. Bien qu'étant le résultat attendu, ceci n'était pas une garantie. SOBAL aurait très bien pu être optimal pour répondre à *Pb.Estimation*. Il convient de ne pas oublier qu'un grand intérêt de GOBAL est de pouvoir répondre à *Pb.GoalOriented* dans sa totalité.

Comme pour SOBAL, lorsqu'un seul capteur est utilisé, l'outil parcimonie (réglé via n_{mul}) est inefficace. Ceci provient encore de l'étape *Sensor Oriented Sparse Coding* qui

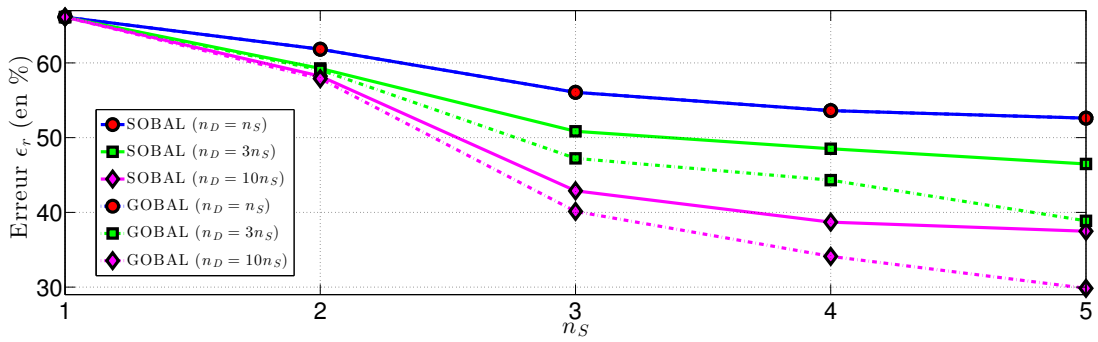

FIGURE 3.12 – GOBAL - Performances $\mathcal{C}_\mathrm{f}$

Nombres de capteurs n_S	**POD**	**SOBAL** $n_D = n_S$	**SOBAL** $n_D = 3n_S$	**SOBAL** $n_D = 10n_S$	**GOBAL** $n_D = n_S$	**GOBAL** $n_D = 3n_S$	**GOBAL** $n_D = 10n_S$
1	72.1%	66.1%	66.1%	66.1%	66.1%	66.1%	66.1%
2	137%	61.8%	59.3%	58.2%	61.8%	59.1%	57.9%
3	110%	56.1%	50.8%	42.9%	56.1%	47.2%	40.1%
4	103%	53.6%	48.5%	38.7%	53.6%	44.3%	34.1%
5	219%	52.6%	46.5%	37.5%	52.6%	38.9%	29.8%

TABLE 3.6 – GOBAL - Performances $\mathcal{C}_\mathrm{f}$

utilise OMP. Le dictionnaire D_1 ne possède que des modes composés par un scalaire (D_1 est un vecteur ligne). Vu le fonctionnement d'OMP, le mode ayant la norme la plus élevée (en valeur absolue) sera toujours retenu pour décrire les *snapshots*.

Enfin, il convient de remarquer l'amélioration notable de l'erreur d'estimation lorsque 5 capteurs sont utilisés. La parcimonie est bien mieux exploitée avec 2 dictionnaires pour cette configuration. Avec $n_\mathrm{mul} = 10$, SOBAL présente une erreur relative de 37.5% tandis que GOBAL présente une erreur relative de 29.8%.

La figure 3.13 présente le comportement de GOBAL dans le cas des capteurs placés de manière "optimale" (Configuration 3). Elle est analogue à la courbe précédente. La taille des dictionnaires est modifiée de la même manière.

La même remarque que pour le placement précédent peut être faite : lorsque plus d'un capteur est utilisé, GOBAL fait toujours mieux que SOBAL pour les positions étudiées. Ceci est encore le cas lorsque 5 capteurs ou plus sont utilisés. Pour illustrer ce point, le tableau 3.7 peut être examiné. Vu les essais réalisés, un lancer de l'algorithme GOBAL permet le plus souvent d'obtenir de meilleures performances que la *meilleure performance* SOBAL obtenue jusqu'à présent.

Ensuite, les courbes relatives à SOBAL et GOBAL lorsque le dictionnaire est plein ($n_\mathrm{mul} = 1$) sont confondues. Lorsque le dictionnaire est plein, ces deux approches fournissent des dictionnaires aux performances identiques (sur au moins 4 chiffres significatifs).

Un dernier point notable est le comportement de GOBAL lorsque 3 capteurs sont utilisés. Le gain obtenu en augmentant le multiplicateur (de 1 à 3) est relativement plus grand que toutes les autres positions. Cela veut peut-être dire qu'avec encore plus de lancers, des dictionnaires encore plus performants dans les autres configurations de capteurs pourraient être trouvés.

Enfin, afin de ne pas surcharger ce chapitre avec plus d'exemples d'applications de GOBAL, le lecteur est invité à se référer au chapitre 6. Plusieurs problèmes de type *Pb.GoalOriented* y sont traités. En particulier, un cas d'application synthétique est consi-

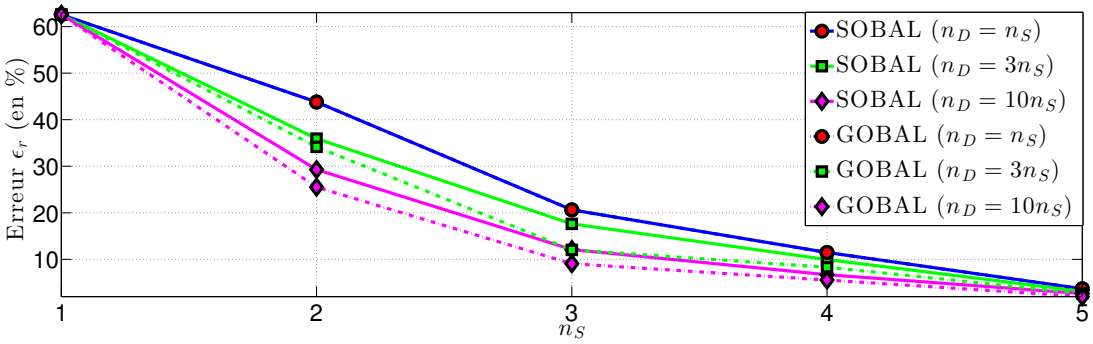

FIGURE 3.13 – GOBAL - Performances $\mathcal{C}_{\text{opt}}$

Nombres de capteurs n_S	**POD**	**SOBAL** $n_D = n_S$	**SOBAL** $n_D = 3n_S$	**SOBAL** $n_D = 10n_S$	**GOBAL** $n_D = n_S$	**GOBAL** $n_D = 3n_S$	**GOBAL** $n_D = 10n_S$
1	64.0%	62.6%	62.6%	62.6%	62.6%	62.6%	62.6%
2	46.6%	43.8%	35.9%	29.3%	43.8%	34.2%	25.5%
3	20.6%	20.6%	17.6%	12.1%	20.6%	12.0%	9.06%
4	12.3%	11.5%	9.56%	6.70%	11.5%	8.31%	5.56%
5	3.68%	3.68%	3.13%	2.65%	3.68%	2.67%	2.02%

TABLE 3.7 – GOBAL - Performances $\mathcal{C}_{\text{opt}}$

déré où, à partir des mesures $S = CY$, le champ $H = Y \odot Y$, où $\odot$ représente le produit matriciel de Hadamard (encore appelé produit de Schur). En d'autres termes, ce dernier effectue le produit, terme à terme, de matrices de mêmes dimensions. Dans le cas considéré $H = Y \odot Y$ correspond au champ Y où chaque élément est mis au carré. Bien évidemment, l'algorithme ne dispose pas de cette information sur la nature de H et doit élaborer D_1 et D_2 à partir des séquences d'entraînements seules.

3.4 Robustesse au bruit de mesure

Une question fondamentale qu'il reste à répondre est : comment ces méthodes se comportent face à des mesures bruitées ?

En pratique, la nature réelle des actionneurs fait qu'il y aura toujours du bruit de mesure : les mesures effectuées ne peuvent pas donner l'image exacte du champ de pression. Cependant, un problème plus profond réside dans le fait que toute la modélisation effectuée repose sur un modèle *imparfait*. Ce faisant, il y a toujours une différence entre ce qui est mesuré et ce qui est prévu par la théorie.

3.4.1 Modélisation

Le bruit de mesure va maintenant être modélisé.

Soit $B \in \mathbb{R}$ une variable aléatoire scalaire et réelle. Une réalisation de cette variable aléatoire est notée $b \in \mathbb{R}$. Ainsi, en notant $s \in \mathbb{R}$ la mesure idéale, non bruitée, produite par un capteur, la mesure réelle est représentée par la variable aléatoire $S_b \in \mathbb{R}$:

$$S_b = s + B \tag{3.48}$$

Une réalisation de cette variable aléatoire S_b est notée s_b et vérifie :

$$s_b = s + b \tag{3.49}$$

Une grande partie des articles s'attaquant au problème de reconstruction soulevé dans ce manuscrit considère un bruit de mesure particulier. Le bruit B est de plus supposé continu et suivant une loi gaussienne, dite encore loi normale, de moyenne $m \in \mathbb{R}$ et d'écart type $\sigma \in \mathbb{R}_+^*$. Cette hypothèse est notée :

$$B \sim \mathcal{N}(m, \sigma) \tag{3.50}$$

La densité de probabilité de cette variable aléatoire B est notée $f_B(.)$ et vérifie :

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R}_+ \\ b &\longmapsto f_B(b) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(b-m)^2}{\sigma^2}} \end{aligned} \tag{3.51}$$

Il convient maintenant de considérer la séquence des mesures effectuées sur la totalité des n_S capteurs.

Soient $B_1, \ldots, B_{n_S}$ des variables aléatoires, scalaires, continues et suivant chacune la loi normale $\mathcal{N}(m, \sigma)$ présentée ci-dessus. Ces variables aléatoires sont supposées **indépendantes dans leur ensemble**. En d'autres termes, la connaissance de la réalisation du bruit sur un capteur ne donne aucune information sur les réalisations des bruits sur les autres capteurs. Le bruit sur chaque capteur de dépend pas des autres.

Comme il a été vu jusqu'à présent, le vecteur de mesure à un instant donné est noté $\boldsymbol{s} \in \mathbb{R}^{n_S}$. Ce dernier correspond au cas non-bruité. Une réalisation du vecteur de mesure bruité est notée $\boldsymbol{s}_b \in \mathbb{R}^{n_S}$ et vérifie :

$$\boldsymbol{s}_b = \boldsymbol{s} + \boldsymbol{b}, \tag{3.52}$$

où $\boldsymbol{b} \in \mathbb{R}^{n_S}$ est une réalisation du vecteur aléatoire $B_s = \left(B_1^T \ldots B_{n_S}^T\right)^T$.

Ce vecteur aléatoire suit une loi gaussienne multivariée. La densité de probabilité associée est :

$$\begin{aligned} \mathbb{R}^{n_S} &\longrightarrow \mathbb{R}_+ \\ \xi &\longmapsto f_{B_s}(\boldsymbol{b}) = \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^{n_S}} e^{-\frac{1}{2}\frac{\sum_{i=1}^{n_S}(b_i-m)^2}{\sigma^2}} \end{aligned} \tag{3.53}$$

où b_i sont les n_S composantes de $\boldsymbol{b}$.

Dans le cas étudié sont considérés uniquement des mesures effectuées aux instants des *snapshots*. Cette condition avait permis de créer S, la matrice contenant toutes les mesures non bruitées effectuées sur la séquence d'entraînement Y.

La dernière hypothèse concernant le bruit est le fait que la connaissance de la réalisation du bruit de mesure pour l'instant i (index d'un *snapshot*) ne donne aucune information sur les possibles réalisations du bruits à d'autres instants. Pour définir cette propriété sans ambiguïtés, il est nécessaire de définir le signal aléatoire vectoriel suivant :

$$\left[B_{\boldsymbol{s}_1}, \ldots, B_{\boldsymbol{s}_{n_{\text{snap}}}}\right] \tag{3.54}$$

où $B_{\boldsymbol{s}_i}$ est un vecteur aléatoire ayant les mêmes propriétés statistiques que $B_{\boldsymbol{s}}$ présenté ci-dessus, $1 \leq i \leq n_{\text{snap}}$. L'hypothèse précédente consiste à dire que ce signal aléatoire est **blanc**. Ceci impose que la fonction d'autocorrélation de ce signal aléatoire soit représentée par une distribution de Dirac centrée en 0. Cette hypothèse de blancheur impose aussi que le signal soit stationnaire au sens strict. Les propriétés statistiques d'ordre 1 et 2 du signal S_B ne dépendent pas de l'origine des temps.

Maintenant que toutes les hypothèses ont été énoncées, il convient de faciliter la manipulation de ces mesures bruitées. La matrice des mesures bruitées est assimilable au signal aléatoire vectoriel précédent. Cette matrice s'écrit :

$$S_B = S + \Xi \tag{3.55}$$

où

$$\Xi = \left[B_{\boldsymbol{s}_1}, \ldots, B_{\boldsymbol{s}_{n_{\text{snap}}}}\right] \tag{3.56}$$

Ξ est une matrice aléatoire dont chaque entrée est une variable aléatoire suivant une loi gaussienne de moyenne m et d'écart type σ.
Une réalisation de Ξ est notée ξ. Une réalisation de S_B est noté S_b.

3.4.2 Dictionnaires robustes

L'objectif est maintenant de proposer une approche pour rendre les dictionnaires produits par les méthodes SOBAL et GOBAL robustes à des mesures bruités. Le cas suivant est considéré :

— Y est la matrice à estimer
— $S_b = S + \xi = CY + \xi$ est la matrice mesure
— ξ est une réalisation de la matrice bruit de mesure

Cependant, en pratique, la réalisation du bruit n'est pas connue (cela reviendrait à connaître la valeur de la mesure non-bruitée avant d'avoir effectué la mesure). Les dictionnaires doivent donc permettre une estimation minimisant l'espérance de l'erreur d'estimation :

$$\mathrm{E}\left[Y - \widehat{Y}\right] \tag{3.57}$$

où $\mathrm{E}[.]$ est l'opérateur espérance (sur toutes les variables aléatoires présentes dans l'expression entre crochet).

Il existe maintenant une approche qui permet, **sans modification des algorithmes**, de produire des dictionnaires robustes au bruit. En se rappelant que l'algorithme SOBAL généralisé et GOBAL permettent d'estimer H à partir de mesures faites sur Y, il suffit de considérer que :

— H contient le champ non-bruité à estimer, soit $H \leftarrow Y$
— S contient les mesures bruitées, soit $S \leftarrow CY + \xi$
— où ξ est créée artificiellement
— tout en rajoutant de la **redondance** dans la séquence Y

Pour illustrer ce point, considérons un seul *snapshot* $\boldsymbol{y}$ et la mesure non bruitée $\boldsymbol{s}$. En pratique, les mesures accessibles sont de la forme $\boldsymbol{s} + \boldsymbol{b}$, où $\boldsymbol{b}$ est une réalisation du vecteur aléatoire $B_{\boldsymbol{s}}$. Soient 3 réalisations différentes de $B_{\boldsymbol{s}}$ notés $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ et $\boldsymbol{b}_3$. Ces dernières sont créées de toute pièce à partir des informations *a priori* sur le bruit qui sera mesuré. L'objectif est que les dictionnaires produits permettent d'obtenir une estimée $\widehat{\boldsymbol{y}}$ proche de $\boldsymbol{y}$ à partir de $\boldsymbol{b}_1, \boldsymbol{b}_2$ et $\boldsymbol{b}_3$.

Afin d'alléger les notations, les deux outils suivants sont introduits :

— le produit de Kronecker $\otimes$,
— le vecteur ligne de taille $n \in \mathbb{N}^*$ contenant uniquement des termes 1, noté J_n.

Par exemple, le vecteur J_4 correspond au vecteur $[1\ 1\ 1\ 1]$.

En créant la séquence $H = J_3 \otimes \boldsymbol{y}$ et $S = J_3 \otimes \boldsymbol{s} + [\boldsymbol{b}_1\ \boldsymbol{b}_2\ \boldsymbol{b}_3]$, les estimateurs associés aux dictionnaires de type K-SVD sont rendus naturellement robustes au bruit.

Suivant la densité de probabilité utilisée pour générer les ξ, les estimateurs peuvent être rendus robustes à des bruits de mesures suivant des lois de probabilités quelconques.

L'approche retenue est maintenant formalisée puis illustrée à l'aide d'expériences numériques. En utilisant l'approche SOBAL généralisé et GOBAL, il est possible de rendre les dictionnaires robustes au bruit de mesure. Pour cela, il faut ajouter de la redondance dans la séquence d'entraînement Y. Il faut dupliquer les *snapshots* qui doivent être rendus robustes. Ensuite, les mesures associées sont aussi dupliquées puis artificiellement bruitées selon la loi de probabilité susceptible de décrire le bruit agissant sur les capteurs.

Ainsi, en considérant une redondance $n_r \in \mathbb{N}^*$:

— $H = J_{n_r} \otimes Y$ (Y est répétée n_r fois)

— $S = CH + \xi$

— où ξ est la réalisation d'une matrice aléatoire de taille $n_S \times n_{\text{snap}} n_r$ dont chaque composante suit une **même** loi de probabilité $f(.)$

Il convient de noter que l'hypothèse de bruit gaussien n'est pas nécessaire ici, contrairement à beaucoup d'approches.

Enfin, toute cette approche revient à minimiser *a posteriori* l'espérance de l'erreur d'estimation $||Y - \widehat{Y}||_F$. L'estimateur obtenu correspond aussi à un estimateur de Bayes dont le risque est de nature quadratique.

3.4.3 Illustration numérique

L'exemple de l'écoulement autour du cylindre est une nouvelle fois utilisé. La disposition $\mathcal{C}_{\text{opt}_3}$ obtenue avec SensorSpace (voir le chapitre 5) est utilisée. Y est associée à la séquence d'entraînement $\mathcal{S}_1$. GOBAL utilise ici un multiplicateur de $n_{\text{mul}} = 3$.

En pratique, la redondance est augmentée jusqu'à obtenir des résultats satisfaisants. Il s'est avéré qu'une redondance de 3 est adaptée pour la séquence retenue.

Le **bruit de mesure** pour chaque capteur est considéré gaussien de moyenne nulle (bruit centré) et d'écart type $\sigma_b \in \mathbb{R}_+$.

Le **bruit simulé**, pour la création des dictionnaires, suit une loi gaussienne de moyenne nulle et d'écart type $\sigma_n \in \mathbb{R}_+$.

Des écarts-types différents sont considérés. Différents dictionnaires sont produits suivant la valeur de σ_n. Ensuite, 100 essais d'estimations sont réalisés sur la totalité des *snapshots* avec différentes réalisations de σ_b. La moyenne des erreurs d'estimation obtenues sur la séquence totale permet d'approximer numériquement l'espérance $\mathrm{E}\left[H - \widehat{H}\right]$.

Les résultats se trouvent dans le tableau 3.8.

σ_n^2	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.01$	$\sigma_b^2 = 0.02$	$\sigma_b^2 = 0.05$	$\sigma_b^2 = 0.1$	$\sigma_b^2 = 0.5$
GOBAL	12.0%	12.3%	12.5%	13.6%	20.6%	94.3%
GOBAL (0.1)	13.4%	13.4%	13.5%	14.00%	15.5%	39.3%
GOBAL (0.5)	23.0%	23.0%	23.0%	23.0%	23.2%	29.4%

Table 3.8 – GOBAL - Performances dans le cas bruité

La première ligne correspond à l'algorithme GOBAL utilisé sur les données non bruités. Les deux lignes suivantes correspondent aux résultats fournis par l'algorithme GOBAL entraîné sur des mesures bruités.

Il faut remarquer la robustesse inhérente à GOBAL. Sans aucune modification, les dictionnaires produits sont capables de fournir de très bon résultats dans un environnement bruité. Cependant, si la variance du bruit attendue est $\sigma_b^2 = 0.1$, les dictionnaires

appropriés (deuxième ligne) sont capables d'effectuer une erreur d'estimation plus faible dans ce contexte (case verte). Le même comportement est remarqué pour $\sigma_b^2 = 0.5$ et les dictionnaires GOBAL associés.

GOBAL peut donc être utilisé dans un contexte bruité et, si besoin, peut incorporer des informations sur le bruit de mesure afin de produire des dictionnaires plus adaptés.

3.5 Comportement face à une séquence originale

Avant de pouvoir conclure ce chapitre, il reste à discuter du comportement de ces algorithmes lorsqu'ils sont appliqués à des *snapshots* qui ne sont pas inclus dans la séquence d'entraînement.

C'est ici une grande limitation des approches se basant sur des données d'entraînements seulement. Les performances de tels algorithmes ne sont aussi bonnes que ce que la séquence d'entraînement leur permet d'atteindre.

La **nouvelle** séquence test $\mathcal{S}_\mathrm{t}$ utilisée ici est très courte (50 *snapshots* pris toutes les $0.4s$). Elle est obtenue à partir de l'utilisation de commandes sinusoïdales alternées seulement. Contrairement aux anciennes séquences tests ($\mathcal{S}_\mathrm{l}$ et $\mathcal{S}_\mathrm{c}$), les commandes constantes ne sont plus appliquées. De plus, la fréquence de la sinusoïde utilisée pour créer la commande n'est plus égale à la fréquence de détachement de tourbillons. La figure 3.14 contient les commandes appliquées et l'évolution du coefficient de traînée associé.

L'estimation du champ de pression associé à $\mathcal{S}_\mathrm{t}$ va être effectuée à partir des dictionnaires POD et GOBAL obtenues à partir de la séquence $\mathcal{S}_\mathrm{l}$. Les positions $\mathcal{C}_\mathrm{opt}$ des capteurs sont utilisées.

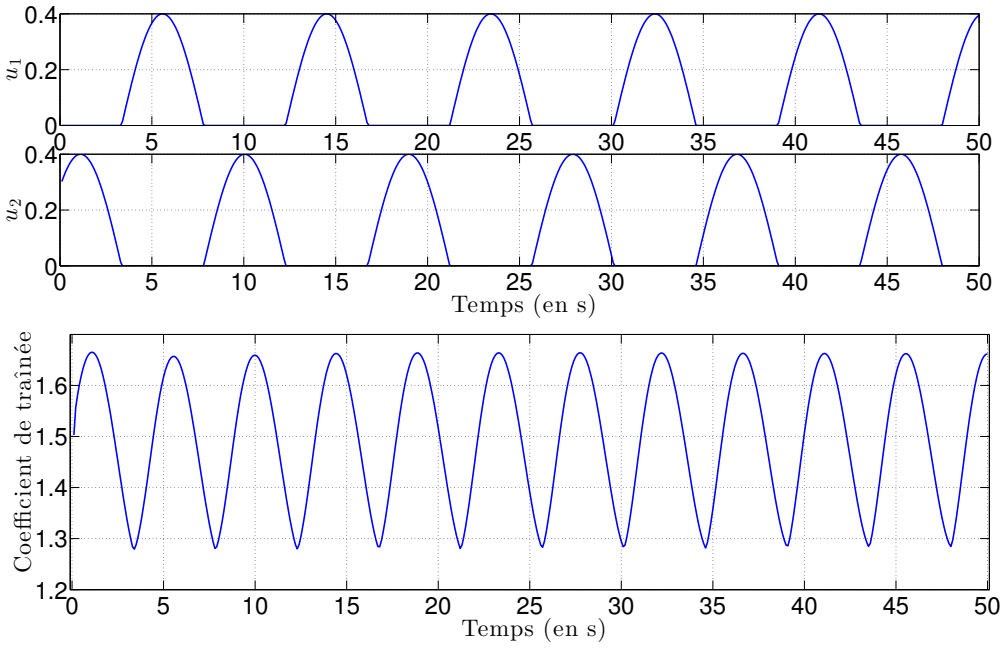

FIGURE 3.14 – Séquence Test - commande (haut), coeff. de traînée (bas)

Le tableau 3.9 contient les résultats de cette étude. Les cases en rouge indiquent que l'approche GOBAL a donné de moins bons résultats que l'approche POD.

Nombres de capteurs n_S	**POD**	**GOBAL** $n_D = n_S$	**GOBAL** $n_D = 3n_S$	**GOBAL** $n_D = 10n_S$
1	76.6%	75.5%	72.5%	72.5%
2	51.5%	49.6%	47.0%	71.4%
3	33.2%	33.8%	25.6%	22.2%
4	18.5%	17.5%	20.1%	19.2%
5	5.27%	5.27%	6.29%	6.01%

TABLE 3.9 – GOBAL - Peformance $\mathcal{C}_{\text{opt}}$ - séquence test $\mathcal{S}_{\text{t}}$

Lorsque $n_{\text{mul}} = 3$ et $n_{\text{mul}} = 4$, l'approche GOBAL est moins performante que l'approche POD si plus de 3 capteurs sont utilisés. L'approche POD présente une plus grande robustesse. Plus le nombres de modes utilisés par GOBAL est grand, plus cette approche risque de ne pas donner de résultats corrects sur une séquence originale.

Un nombre de modes n_D plus faible (vérifiant $n_S \leq n_D < 3n_S$ par exemple) pourrait encore permettre à GOBAL de faire mieux que l'approche POD. Ceci est visible lorsque $n_D = n_S$ et lorsque $n_S = 1$ ou $n_S = 2$. L'approche GOBAL est capable de donner de meilleurs résultats que l'approche POD surtout lorsque la **parcimonie utilisée est faible**.

Cependant, dans le cas 3 capteurs, même un multiplicateur $n_{\text{mul}} = 10$ est exploitable. Certaines positions des capteurs sont aussi plus adaptées à la séquence $\mathcal{S}_{\text{t}}$ que d'autres.

Cette dernière expérience permet d'insister sur l'importance de la séquence d'entraînement choisie. Bien qu'ici $\mathcal{S}_{\text{t}}$ ressemble aux séquence précédentes $\mathcal{S}_{\text{l}}$, $\mathcal{S}_{\text{c}}$, elle contient des informations qui n'ont pas pu être décrites correctement par les dictionnaires extraits de $\mathcal{S}_{\text{l}}$.

3.6 Conclusion

Ce chapitre a permis de présenter une nouvelle méthode pour répondre au problème *Pb.Estimation*. L'approche SOBAL conçue permet de créer une base adaptée à une position donnée des capteurs et qui est capable de fonctionner en pratique à partir d'un très faible nombre de mesures.

Dans des situations où les capteurs sont déjà fixés, une telle approche, bien que plus coûteuse que l'approche POD dans la phase hors-ligne, permet d'améliorer les performances d'estimation. Il ne faut pas oublier qu'il n'est pas toujours possible de placer les capteurs dans les zones les plus informatives. Dans ces cas, une telle approche adaptative peut améliorer les performances d'estimation.

L'approche conçue est particulièrement adaptée lorsque très peu de capteurs sont disponibles. Ceci provient du fait que le problème est considéré dans sa totalité au lieu de ne se concentrer que sur l'obtention d'une représentation réduite qui peut être inaccessible en pratique ou limitée de par le nombre de modes retenus (pour l'approche POD). Il est aussi remarquable que l'erreur d'estimation effectuée diminue très rapidement lorsque de nouveaux capteurs sont ajoutés (moyennant un placement *intelligent* de ces derniers). Dans ces cas, il est envisageable d'utiliser l'approche classique POD puisque elle est bien plus rapide à mettre en œuvre.

Ensuite, l'algorithme SOBAL a pu être généralisé pour agir sur un nouveau problème *Pb.GoalOriented*. Un cas simple de ce problème où une relation de nature matricielle linéaire lie S et H a été considéré et est solvable par cet algorithme SOBAL généralisé. Enfin, le problème *Pb.GoalOriented* a été considéré dans le cas général et l'algorithme

GOBAL a été proposé pour le résoudre. Les performances de ce dernier dépendent très fortement de la nature de S et H. Il est notable qu'un tel algorithme manipulant deux dictionnaires permet d'obtenir de meilleurs résultats que SOBAL dans le cadre du problème *Pb.Estimation*.

Le chapitre suivant va introduire une nouvelle dimension à la résolution de ce problème et va permettre d'améliorer de manière notable la résolution de *Pb.Estimation* et surtout de *Pb.GoalOriented* qui en est la généralisation. Ceci va se faire à travers la notion de classification. En d'autres termes, le problème *Pb.GoalOriented* va être découpé en sous-problèmes où une approche du type GOBAL, qui produit une estimation linéaire, est bien plus adaptée à le résoudre. Cette approche est similaire à la linéarisation par morceau en analyse.

Bibliographie

[1] Claude E. Shannon. Communication In The Presence Of Noise. In *Proceedings of Institute of Radio Engineers*, volume 37, pages 10–21, 1949.

[2] Emmanuel J. Candès and Terence Tao. Near-Optimal Signal Recovery From Random Projections : Universal Encoding Strategies ? *IEEE Transactions on Information Theory*, 52(12) :5406–5425, December 2006.

[3] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8) :1207–1223, August 2006.

[4] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4) :1289–1306, April 2006.

[5] Saad Qaisar, Rana Muhammad Bilal, Wafa Iqbal, Muqaddas Naureen, and Sungyoung Lee. Compressive Sensing : From Theory to Applications , A Survey. *Communications and Networks, Journal of*, 15(5) :443–456, 2013.

[6] T. Tony Cai, Guangwu Xu, and Jun Zhang. On Recovery of Sparse Signals Via l1 Minimization. *IEEE Transactions on Information Theory*, 55(7) :3388–3397, July 2009.

[7] T. Tony Cai, Lie Wang, and Guangwu Xu. New Bounds for Restricted Isometry Constants. *IEEE Transactions on Information Theory*, 56(9) :4388–4394, September 2010.

[8] Michael Elad. *Sparse and Redundant Representations.* Springer New York, New York, NY, 2010.

[9] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM Review*, 51(1) :34–81, February 2009.

[10] David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1) :6–18, January 2006.

[11] Yonina C. Eldar and Gitta Kutyniok, editors. *Compressed Sensing.* Cambridge University Press, Cambridge, 2012.

[12] Robert Tibshirani. Regression Selection and Shrinkage via the Lasso. *Journal of the Royal Statistical Society B*, 58 :267–288, 1994.

[13] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3) :265–274, 2009.

[14] Daniele Dessi. Load field reconstruction with a combined POD and integral spline approximation technique. *Mechanical Systems and Signal Processing*, 46(2) :442–467, June 2014.

[15] S. Kho, C. Baker, and R. Hoxey. POD/ARMA reconstruction of the surface pressure field around a low rise structure. *Journal of Wind Engineering and Industrial Aerodynamics*, 90(12-15) :1831–1842, December 2002.

[16] Xiu Yang, Daniele Venturi, Changsheng Chen, Chryssostomos Chryssostomidis, and George Em Karniadakis. EOF-based constrained sensor placement and field reconstruction from noisy ocean measurements : Application to Nantucket Sound. *Journal of Geophysical Research*, 115(C12) :C12072, December 2010.

[17] Mihailo R. Jovanovich, Peter J. Schmid, and Joseph W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2), 2014.

[18] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656(July 2010) :5–28, July 2010.

[19] Clarence W. Rowley. Model Reduction For Fluids Using Balanced Proper Orthogonal Decomposition. *International Journal of Bifurcation and Chaos*, 15(03) :997–1013, March 2005.

[20] K Willcox and J Peraire. Balanced Model Reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, 40(11) :2323—-2330, 2002.

[21] Michal Aharon, Michael Elad, and Alfred M. Bruckstein. K-SVD : An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11) :4311–4322, November 2006.

[22] Stéphane Mallat. *Une exploration des signaux en ondelettes*. Ecole Polytechnique, 2011.

[23] Sylvain Lesage, Rémi Gribonval, Frédéric Bimbot, and Laurent Benaroya. Learning unions of orthonormal bases with thresholded singular value decomposition. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume V, pages 293–296. IEEE, 2005.

[24] Deyu Meng, Qian Zhao, Yee Leung, and Zongben Xu. Learning dictionary from signals under global sparsity constraint. *Neurocomputing*, 119 :308–318, November 2013.

[25] James B. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[26] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Technical report, Technion, Haifa, Israël, 2008.

[27] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Double sparsity : Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3 PART 2) :1553–1564, March 2010.

[28] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hokpinks University Press, Baltimore, MD, 3rd editio edition, 1996.

[29] Kuang-Yu Chang, Cheng-Fu Lin, Chu-Song Chen, and Yi-Ping Hung. Single-Pass K-SVD for Efficient Dictionary Learning. *Circuits, Systems, and Signal Processing*, 33(1) :309–320, July 2013.

[30] Zhuolin Jiang, Zhe Lin, and Larry S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *CVPR 2011*, pages 1697–1704. IEEE, June 2011.

[31] Paul Ruvolo and Eric Eaton. Online Multi-Task Learning via Sparse Dictionary Optimization. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, number 1, 2014.

[32] Qiang Zhang and Baoxin Li. Discriminative K-SVD for dictionary learning in face recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2691–2698. IEEE, June 2010.

[33] Vasilios N. Katsikis, Dimitrios Pappas, and Athanassios Petralias. An improved method for the computation of the Moore-Penrose inverse matrix. *Applied Mathematics and Computation*, 217(23) :9828–9834, 2011.

CHAPITRE 4
Parcimonie et Classification

Chapitre 4

Parcimonie et Classification

Contenu

Ce chapitre a pour objectif de proposer une méthode encore **plus performante** que GOBAL pour répondre au problème *Pb.GoalOriented*. GOBAL repose sur SOBAL qui est lui-même basé sur la K-SVD. L'utilisation de représentations réduites creuses et de l'intégration explicite des mesures disponibles en pratique sont les sources de ces bonnes performances.

Afin d'aller plus loin dans la résolution de ce problème, la notion de **classes** va être exploitée. L'idée principale est de **regrouper les *snapshots*** d'une manière efficace afin d'engendrer des dictionnaires **plus adaptés**. Le gain ne doit pas provenir d'une augmentation du nombre de modes mais d'une meilleure gestion de ces derniers.

La première partie de ce chapitre permet d'illustrer la **classification**. Les outils principaux vont être présentés ainsi que leurs possibles interactions et intérêts pour répondre au problème *Pb.GoalOriented*.

Ensuite, la seconde section est dédiée à un nouvel algorithme permettant d'effectuer une classification adaptée à l'utilisation future de GOBAL. Différentes variantes de cet algorithme seront proposées.

Enfin, la méthode centrale de ce chapitre va être décrite. Elle allie la nouvelle méthode de classification et l'algorithme GOBAL. Cette combinaison n'est pas triviale et de nouveaux problèmes de *structure* sont soulevés. Une seule approche pour déterminer la *structure* sera utilisée malgré le fait qu'elle ne soit pas optimale. Cette dernière est rapidement applicable quel que soit le problème considéré.

4.1 La classification

4.1.1 Motivations

La classification est une action possiblement complexe. Elle peut être effectuée intuitivement par certains être vivants principalement grâce à leurs expériences (culture, connaissance, abstraction, ...). Elle consiste à regrouper des éléments qui possèdent certains points communs. Une telle opération est souvent subjective et est aussi le fruit d'une certaine expérience personnelle.

La classification peut être extrêmement simple ou complexe à traduire sous une forme algorithmique. Classer un ensemble de nombre en deux groupes, nombres strictement positifs ou non, est une action triviale. Cependant, séparer des chiens et des chats, action qui semble être une évidence pour un être humain de quelques années seulement, est une tâche très compliquée à traduire pour une machine. Derrière cette opération se cache une mise en situation et un apprentissage. Ce conditionnement fait qu'il est parfois délicats de traduire exactement le raisonnement logique qui permet de classer des objets. Certains critères sont prioritaires devant d'autres et peuvent être de nature subjective et ainsi très délicat à traduire sous forme algorithmique.

La classification cherche à grouper des éléments (de nature possiblement très variée) selon un critère. Étant donné l'importance de l'apprentissage pour l'opération de classification effectuée par l'être humain, il est naturel que le domaine du Machine Learning est un des plus friands de tels problèmes. La difficulté de produire un algorithme performant en classification est déviée vers la difficulté d'obtenir un algorithme capable d'apprendre un tel schéma. Suivant les éléments à classer, en particulier en traitement de l'image, des étapes supplémentaires doivent être effectuées comme la détection de contours ou de formes. Ces informations sont ensuite mises sous une forme exploitable par un tel algorithme d'apprentissage.

Dans le cadre de cette thèse, une approche **Machine Learning** de la classification est retenue. Une des contraintes est de ne plus introduire d'*a priori* physique (provenant de l'utilisateur) une fois l'obtention des séquence d'entraînement. De ce fait, les classes et leur nombre **ne sont pas connus à l'avance**. Avec cette contrainte supplémentaire, l'action de classification est connue sous le nom de classification **non-supervisée** ou encore de **clustering**. L'objectif d'un tel algorithme de classification sera de grouper les *snapshots* qui se ressemblent mais de faire en sorte que chaque classe est la plus différentes des autres afin d'améliorer les performances d'estimation globale. Les notions de ressemblances et de différences sont le plus souvent traitées à l'aide d'une métrique sur l'espace des éléments à classer. Au lieu de n'utiliser que les séquence (H, S) (voir 3.3) pour obtenir les éléments définissant les estimateurs (dictionnaires), ces séquences sont découpées afin de permettre l'obtention de plusieurs estimateurs **adaptés**.

Afin de permettre une **comparaison propre** avec les approches SOBAL et GOBAL précédentes, le nombre de "modes" exploités par cette nouvelle approche devra être similaire aux anciennes. Cette nouvelle approche de classification se base uniquement sur les mêmes informations (H, S) que les algorithmes précédents.

Le problème de classification non-supervisée associé à la résolution de $Pb.GoalOriented$ est maintenant formalisé.

Soit $n_K \in \mathbb{N}^*$ le nombre de classes distinctes utilisées. n_K est impérativement supérieur

à 1 afin que la classification puisse possiblement conduire à une synthèse d'estimateurs plus performants. Afin de simplifier le problème, le nombre de classes est imposé par l'utilisateur. Dans la suite, des conseils pour effectuer ce choix seront donnés.

Soit $\mathcal{K}(.)$ l'application de $\mathbb{R}^{n_S} \to \mathbb{N}^*$ qui permet d'effectuer la classification. Elle agit ainsi sur un élément de S, possédant n_S composantes, et y associe un entier $k \in \mathbb{N}^*$ indiquant l'index de classe. Une fois une telle application connue, les *snapshots* de S et de H associés à la classe k peuvent être regroupés pour former S_k et H_k respectivement. Les index de ces *snapshots* sont contenus dans l'ensemble $T_{\text{class}k}$.

$\mathcal{K}(.)$ agit uniquement sur un élément de S afin de trouver l'index de la classe du *snapshot* étudié. Ceci est une contrainte nécessaire afin qu'un tel outil puisse être utilisé en temps réel. S et H sont utilisées conjointement, et dans leur totalité, afin de produire le classificateur $\mathcal{K}(.)$. Cependant, ce dernier ne peut plus avoir accès à ces séquences lors du fonctionnement en temps réel.

Chaque classe k est associée à un estimateur $\Psi_k(.)$ défini sur $\mathbb{R}^{n_S} \to \mathbb{R}^{n_{\boldsymbol{h}}}$. Si $\boldsymbol{s}_i$ appartient à la classe k, alors $\widehat{\boldsymbol{h}}_i$ est donnée par :

$$\widehat{\boldsymbol{h}}_i = \Psi_k(\boldsymbol{s}_i). \tag{4.1}$$

Cet estimateur agit sur les *snapshots* de S_k afin de produire une estimée $\widehat{H}_k$ de H_k. La matrice $\widehat{H} \in \mathbb{R}^{n_{\boldsymbol{h}} \times n_{\text{snap}}}$ est donnée par :

$$\widehat{H} = \left[\widehat{\boldsymbol{h}}_1, \ldots, \widehat{\boldsymbol{h}}_{n_{\text{snap}}}\right] \tag{4.2}$$

Si les *snapshots* de H sont ré-ordonnés afin d'être regroupés par classes, la matrice H_{ord} résultante est obtenue selon :

$$H_{ord} = [H_1 \ldots H_{n_K}] \tag{4.3}$$

L'estimée de la séquence totale H_{ord} est définie selon :

$$\widehat{H}_{ord} = \left[\widehat{H}_1 \ldots \widehat{H}_{n_K}\right], \tag{4.4}$$

où $\widehat{H}_k$ est formée par la concaténation des images des vecteurs de S_k par l'application $\Psi_k(.)$.

Il ne reste plus qu'à formuler l'objectif des classes. En d'autres termes, il faut définir le point commun des éléments d'une même classe afin de pouvoir espérer les former. Les classes doivent générer des estimateurs $\Psi_k(.)$ qui permettent conjointement de former une matrice $\widehat{H}$ qui minimise l'erreur relative $\epsilon_r \in \mathbb{R}_+$ où

$$\epsilon_r = ||H_{ord} - \widehat{H}_{ord}||_F / ||H_{ord}||_F. \tag{4.5}$$

Vu la norme utilisée, cette erreur peut s'exprimer de manière équivalente avec :

$$\epsilon_r = ||H - \widehat{H}||_F / ||H||_F. \tag{4.6}$$

Ceci est aussi l'occasion de définir l'erreur relative par classe notée $\epsilon_{rk} \in \mathbb{R}_+$. Cette dernière vérifie

$$\epsilon_{rk} = ||H_k - \widehat{H}_k||_F / ||H_k||_F. \tag{4.7}$$

Ces classes sont ainsi dépendantes des applications utilisées pour effectuer l'estimation. Cependant, le problème est rendu d'autant plus délicat que les estimateurs sont eux aussi

dépendants des classes créées. En effet, ces estimateurs doivent être adaptés à la séquence associée afin de pouvoir réduire efficacement l'erreur relative ϵ_r.

Le problème de classification/estimation, nommé *Pb.Class*, est résumé ci-dessous

Données : H, S
Objectif : Trouver les applications $\mathcal{K}(.)$ et $\Psi_k(.)$, $1 \leq k \leq n_K$.
$\mathcal{K}(.)$ doit séparer les séquences d'apprentissages en n_K classes distinctes représentées par (H_k, S_k) avec $1 \leq k \leq n_K$.
Ces classes doivent permettre la création de n_K estimateurs représentés par les applications $\Psi_k(.)$ qui elles-mêmes engendrent les $\widehat{H}_k$ qui doivent permettre de minimiser ϵ_r.

Vu les performances de GOBAL exhibées dans le chapitre précédent, ce dernier semble être un candidat parfait pour remplir le rôle de $\Psi_k(.)$. Cependant, vu le couplage existant entre $\Psi_k(.)$ et $\mathcal{K}(.)$, un algorithme aussi complexe que GOBAL est susceptible de rendre l'obtention de $\mathcal{K}(.)$ très délicate (et surtout, très lente à converger). Le choix est fait d'utiliser un estimateur $\Psi_k(.)$ plus simple que GOBAL lors de la recherche de $\mathcal{K}(.)$. Une fois $\mathcal{K}(.)$ obtenue, l'algorithme GOBAL pourra être utilisé par classe afin d'obtenir le couple de dictionnaire représentant $\Psi_k(.)$.

L'attention se porte ainsi vers l'obtention de $\mathcal{K}(.)$. Avant de présenter l'approche retenue, il est nécessaire de parcourir les approches les plus répandues en classification, puisque l'une d'entre elles sera activement utilisée par la suite.

4.1.2 Approches usuelles

Le clustering est utilisé dans de nombreux domaines comme par exemple le **Data Mining** [1]. Ce dernier regroupe l'ensemble des problèmes d'extraction de caractéristiques (inconnues) d'un grand jeu de données.

Dans [2], des structures musicales de divers titres sont extraites. Ceci est un exemple d'extraction de forme (Pattern/Shape Extraction).

Dans [3], des pixels sont regroupés afin d'extraire les caractéristiques fondamentales d'images fixes et mobiles (vidéo). Ceci est un exemple de segmentation.

De très nombreuses méthodes existent pour répondre aux problèmes de clustering ([4]). Sans être exhaustif, il convient de citer les algorithmes suivants, actuellement très utilisés :

- SOM : Self Organising Map [5], qui permet la création d'un réseau de neurones artificiel
- ELM : Extreme Learning Machine [6, 7], qui permet la création d'un réseau de neurones à couche dont une est inconnue
- LVQ : Learning Vector Quantization [8], qui consiste à regrouper des vecteurs possédant certaines caractéristiques
- K-Means [9], qui consiste à regrouper des vecteurs à l'aide de l'utilisation d'une norme euclidienne
- KNN : K-Nearest Neighbours [10], une modification de l'*Update Step* de K-means (voir 4.1.2)
- FCM : Fuzzy C-means, une modification de K-means où les classes sont plus souples
- FOCM : Fuzzy Ordered C-means [11], une modification de FCM pour la rendre plus robuste au bruit

Enfin, plusieurs de ces méthodes de clustering peuvent être utilisées conjointement, comme le montre [12].

Parmi cette myriade de méthodes, l'approche K-means est retenue. La raison princi-

pale est que cette dernière est très simple à mettre en œuvre tout en étant efficace. Cette particularité la rend ainsi facilement modifiable pour les besoins de ce travail de thèse. Son principal inconvénient est le choix du nombre de classes, mais de nombreux travaux de recherche existent sur ce point (voir par exemple [13]). Les paragraphes suivants vont détailler le fonctionnement de K-means.

Enfin, il convient de noter que la contribution majeure de ce chapitre réside dans l'utilisation et l'agencement des différentes approches. Chaque brique composant la méthode peut ainsi être améliorée si besoin. Une meilleure approche pour la classification peut alors être utilisée.

K-means

K-means est un algorithme utilisé pour faire de la classification de données. Bien qu'ayant été proposé en 1967 par JB Macqueen [14], cet algorithme est encore d'actualité grâce à son rapport **efficacité/simplicité** exemplaire. A partir d'une séquence d'entraînement Y, l'algorithme K-means va créer $n_K \in \mathbb{N}^*$ classes, chacune caractérisée par un vecteur appelé le représentant de la classe $\boldsymbol{d}_i \in \mathbb{R}^{n_{\boldsymbol{x}}}, 1 \leq i \leq n_K$. Soit $K \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_K}$ la matrice contenant les différents représentants $\boldsymbol{d}_i$. Chaque élément de Y est ainsi associé à un représentant. Ces derniers sont choisis de façon à approximer le mieux possible (au sens de la norme 2) les vecteurs de Y.

L'algorithme K-means se décompose en 2 étapes : l'affectation (***Assignement Step***) et la mise à jour des représentants (***Update Step***). A partir des représentants contenus dans K, la distance (norme 2) entre chaque *snapshot* et chaque $\boldsymbol{d}_i$ est calculée. Un *snapshot* est dit appartenir à la classe du représentant *le plus proche* de ce dernier. Ceci constitue la première étape.

Ensuite, chaque vecteur de K est **mis à jour** en tant que moyenne des *snapshots* appartenant à la classe traitée. Cette étape très simple constitue l'*Update Step*.

Le processus est ensuite répété jusqu'à ce qu'un nombre d'itérations particulier $n_{\text{it}} \in \mathbb{N}^*$ soit atteint ou que l'erreur de classification effectuée (calculée à partir des distances entre chaque *snapshot* et son représentant) soit suffisamment faible.

Pour un n_K donné, le paramètre ayant le plus d'impact sur les performances de la classification trouvée est le dictionnaire initial K_{ini}. **Le choix initial des représentants est de la plus haute importance** lors de l'utilisation de l'algorithme K-means. L'algorithme converge naturellement vers un minimum local. Bien évidemment, il existe des techniques pour faciliter la mise à jour des représentants et ainsi balayer plus de configurations possibles. Tout comme l'algorithme SOBAL et GOBAL, l'algorithme K-means est à appliquer plusieurs fois, pour différentes conditions initiales, afin d'augmenter les chances d'obtenir une solution satisfaisante. **Le choix du nombre de classes** est aussi un obstacle en pratique. Le pseudo-code de l'algorithme se trouve dans l'algorithme 6.

4.2 GOC : Goal-Oriented Classifier

Cette section va permettre d'illustrer l'une des contributions de ce chapitre : un algorithme permettant d'obtenir une application $\mathcal{K}(.)$ pour répondre à *Pb.Class*. Ce dernier est nommé **Goal-Oriented Classifier** (GOC) mais n'est pas aussi général que son nom l'indique. GOC se base sur l'algorithme K-means présenté précédemment. L'application

Algorithm 6 K-means Pseudo-Code

Require: Y, K_{ini}, n_K, n_{it}
1: **Initialisation** :
2: $K \leftarrow K_{\text{ini}}$
3: $T_{\text{class}} \leftarrow$ vecteur nul de taille n_{snap}
4:
5: **for** $j \in [1 \dots n_{\texttt{it}}]$ **do**
6: **Classification** :
7: **for** $i \in [1 \dots n_{\text{snap}}]$ **do**
8: $T_{\text{class}i} \leftarrow \arg\min_k ||\boldsymbol{y}_i - \boldsymbol{d}_k||_2, 1 \leq k \leq n_K$
9: **Mise à jour** :
10: **for** $k \in [1 \dots n_K]$ **do**
11: id $\leftarrow$ index des *snapshots* de la classe k
12: $Y_k \leftarrow$ restriction de Y aux colonnes indexées par id
13: $\boldsymbol{d}_k \leftarrow$ moyenne des colonnes de Y_k
14:
15: **Sortie** : K

$\mathcal{K}(.)$ est ainsi choisie non-linéaire et repose sur l'utilisation de représentants $\boldsymbol{d}_k \in \mathbb{R}^{n_S}$, avec $1 \leq k \leq n_K$. L'appartenance à une classe est obtenue par un **test de proximité**. Le représentant le plus proche d'un *snapshot*, selon la norme 2, associe son indice de classe à ce *snapshot*. La procédure de classification est ainsi très simple et se base sur une norme bien connue, tout comme K-means. Cette dernière peut être remplacée par une autre norme selon les besoins de l'utilisateur, par exemple la norme 1 ou une norme vectorielle pondérée. L'application non-linéaire $\mathcal{K}(.)$ est représentée par la matrice $K \in \mathbb{R}^{n_S \times n_K}$. L'affectation (ou l'*Assignement Step*) de K-means est inchangée.

Cependant, afin de pouvoir remplir l'objectif demandé, il est impératif de **modifier l'*Update Step* de K-means**. Ce dernier doit exploiter $\Psi(.)$ ainsi que l'objectif final de devoir minimiser ϵ_r. La nature couplée de *Pb.Class* rend cette exploitation délicate. De plus, étant donné que GOBAL est choisi pour remplir le rôle $\Psi(.)$, l'évaluation récursive d'un tel algorithme est à prohiber.

Il devient crucial d'introduire un **estimateur simplifié** afin d'éviter la manipulation de l'algorithme GOBAL lors de la recherche de K. Il faut qu'un tel estimateur simplifié indique si le regroupement actuel des *snapshots* engendrera des estimateurs performants de type GOBAL une fois que la totalité de la séquence est traitée. La méthode de **mise à jour des représentants** va être modifiée en exploitant cette information.

4.2.1 Estimateur simplifié

L'*Update Step* doit permettre de modifier les représentants des classes pour remplir l'objectif recherché. Bien que l'algorithme GOBAL sera utilisé sur chaque classe obtenue, utiliser un tel algorithme lors de la recherche des classes est trop complexe.

Un nouvel estimateur $\Psi(.)$ va alors être proposé. Ce dernier doit être plus léger que GOBAL et doit permetre de s'approcher de ses propriétés d'estimation.

On peut se poser la question en considérant le problème *Pb.Estimation* : pourquoi la K-SVD a pu fournir de si bons résultats ? Pourtant cette méthode, comme l'approche POD, fait le lien entre les mesures $\boldsymbol{s}$ et le champ $\boldsymbol{h}$ à être reconstruit de manière linéaire.

Il a été vu qu'une spécificité de la K-SVD est que le support de la représentation réduite associée à la mesure $\boldsymbol{s}$ est creux.

Soit $T_{\boldsymbol{s}}$ le support de la représentation réduite associé à $\boldsymbol{s}$. Ce dernier est obtenu grâce à l'algorithme OMP qui exploite la corrélation entre les colonnes du dictionnaire CD et les mesures $\boldsymbol{s}$ (plus particulièrement, les résidus successif, voir le pseudo-code 1). La représentation réduite est alors donnée par :

$$\boldsymbol{a} = (CD_{T_{\boldsymbol{s}}})^{+}\boldsymbol{s}, \tag{4.8}$$

où $D_{T_{\boldsymbol{s}}}$ est la restriction de D aux colonnes indexées par $T_{\boldsymbol{s}}$. Le champ recherché est ensuite estimé par :

$$\widehat{\boldsymbol{h}} = D_{T_{\boldsymbol{s}}}(CD_{T_{\boldsymbol{s}}})^{+}\boldsymbol{s}. \tag{4.9}$$

Cela revient alors à utiliser l'opérateur :

$$\Psi\left(\boldsymbol{s}\right) = D_{T_{\boldsymbol{s}}}(CD_{T_{\boldsymbol{s}}})^{+}\boldsymbol{s} = R_{T_{\boldsymbol{s}}}\boldsymbol{s}, \tag{4.10}$$

avec $R_{T_{\boldsymbol{s}}} = D_{T_{\boldsymbol{s}}}(CD_{T_{\boldsymbol{s}}})^{+}$.

Cette relation peut aussi être obtenue dans le cas de l'utilisation de la paire (D_1, D_2) dans le cas de GOBAL. En nommant ce support de la même manière que dans le développement ci-dessus, l'estimateur obtenu est :

$$R_{T_{\boldsymbol{s}}} = D_{1T_{\boldsymbol{s}}}(CD_{2T_{\boldsymbol{s}}})^{+} \ . \tag{4.11}$$

Ainsi, les différentes approches basées sur la K-SVD peuvent être vues comme l'utilisation de **plusieurs estimateurs linéaires** dépendant du choix du support de la représentation réduite. Comme il a été vu dans le chapitre 3, plus de 50 supports distincts peuvent être générés lors d'une utilisation de tels algorithmes. Cela revient alors à utiliser de nombreux estimateurs linéaires différents, définis à partir de la mesure effectuée. Ces estimateurs sont aussi **couplés** puisque un même mode peut intervenir dans de nombreux supports différents. Une telle richesse est aussi, comme vu dans le chapitre précédent, à la source de la difficulté majeure d'obtenir des dictionnaires D_1 et D_2 performants.

L'idée retenue pour proposer un estimateur simplifié est ainsi de limiter le nombre de supports possibles, autrement dit, le nombre d'estimateurs possibles et, en contrepartie, de se défaire de la contrainte de structure sur l'estimateur (exploitant les modes de D_1 et D_2).

Le choix du support n'est plus effectué par un algorithme à contrainte-ℓ_0 type OMP. C'est l'*Assignement Step* de K-means (de GOC) qui est utilisé. **Cela revient à utiliser un estimateur linéaire par classe**.

Les représentants des classes se situent dans la matrice $K \in \mathbb{R}^{n_S \times n_K}$. À partir de ces derniers, chaque *snapshot* peut être associé à une classe k à l'aide d'un test de proximité.

Les *snapshots* de H et S associés à chaque classe sont regroupés dans la paire (H_k, S_k). Enfin, l'estimateur linéaire adapté à cette classe doit permettre de minimiser la norme

$$||H_k - R_k S_k||_F \tag{4.12}$$

Un tel estimateur peut être défini par

$$R_k = H_k\left(S_k\right)^{+} \tag{4.13}$$

Ainsi, l'estimateur $\Psi\left(.\right)$ utilisé par GOC est choisi tel que :

$$\boxed{\Psi\left(\boldsymbol{s}\right) = R_k\boldsymbol{s} \text{ où } k \text{ est la classe de } \boldsymbol{s}} \tag{4.14}$$

4.2.2 Algorithme

Avant de présenter l'algorithme GOC, il convient de présenter le cas simple où aucune mise à jour complexe des classes n'est effectuée. Ceci est intéressant pour avoir un meilleur ressenti de l'intérêt de GOC.

Utilisation directe - GOC simplifié

Cette approche revient à utiliser K-means de manière directe sur S afin d'obtenir les n_K classes. Les classes ont donc été choisies afin de minimiser l'erreur (en norme 2) entre les *snapshots* de la classe et le représentant associé. Si les mesures se ressemblent, il devient naturel de vouloir les exploiter de la même manière, c'est-à-dire d'utiliser un même estimateur pour une classe. Chaque classe ainsi créée est affectée à un estimateur $R_k = H_k (S_k)^+$.

Cependant, une faible erreur de classification sur S n'implique pas une très faible erreur sur l'estimation de H. Étant donné que le meilleur estimateur linéaire est utilisé par classe, la faute vient de la classification.

Par exemple,

— une erreur de classification relative de 17% sur S peut être liée à une erreur d'estimation relative de 3% sur H et

— une erreur de classification relative de 7% sur S peut être liée à une erreur d'estimation relative de 6% sur H

Cependant, suivant l'initialisation de K-means et les mises à jour effectuées des représentants, il est possible de trouver de bonnes classifications pour l'estimation de H. De nombreuses relances de l'algorithme K-means sont nécessaires.

Enfin, plus les champs H et S sont de natures différentes, moins cette approche est valide.

Le pseudo-code de cette approche se trouve dans l'algorithme 7.

Algorithm 7 GOC Simplifié Pseudo-Code

Require: S, H, K_{ini}, n_K, n_{it}

1: **Classification** :
2: Appliquer n_{it} itérations K-means à S
3: (ce dernier est initialisé avec K_{ini})
4: Le dictionnaire K est ainsi obtenu
5: **Estimation** :
6: **for** $j \in [1 \dots n_K]$ **do**
7: id $\leftarrow$ index des *snapshots* de la classe j
8: $S_j \leftarrow$ restriction de S aux colonnes indexées par id
9: $H_j \leftarrow$ restriction de H aux colonnes indexées par id
10: $R_j \leftarrow H_j (S_j)^+$
11:
12: **Sortie** : K et les R_j,$1 \leq j \leq n_K$

Approche GOC

L'approche GOC a pour objectif d'améliorer la probabilité d'obtenir une meilleure classification dans l'objectif d'estimer H avec le moins d'erreur possible. De plus, il faut que GOC puisse fonctionner dans le cas général du problème *Pb.GoalOriented*.

Il faut donc regrouper les éléments de S afin que l'erreur sur H soit la plus faible possible. Ceci est fait à l'aide d'un schéma cyclique. **Une nouvelle classification est réalisée au début de chaque cycle**.

Un cycle basique est maintenant décrit. Ce dernier commence par **classer les éléments** de S en n_K classes à partir de K. Ceci est fait comme le *Assignement Step* de K-means.

Au lieu de directement raffiner les représentants avec la *Update Step*, **les estimateurs** R_j **de chaque classe sont calculés** selon $R_j = H_j\,(S_j)^+$, pour $1 \leq j \leq n_K$.

Chaque paire $(\boldsymbol{h}, \boldsymbol{s})$ va ensuite être traitée. Le *snapshot* $\boldsymbol{s}$ est appliqué à chaque estimateur R_j afin d'obtenir une estimée $\widehat{\boldsymbol{h}}_j = R_j\boldsymbol{s}$. La classe permettant de minimiser la norme $||\boldsymbol{h} - \widehat{\boldsymbol{h}}_j||_2$ devient la nouvelle classe de $\boldsymbol{s}$. Ainsi, lors de chaque cycle, **un remaniement des classes est effectué** autre que dans l'*Assignement Step*.

Une telle modification de la classe a lieu lorsqu'une classe disposant d'un estimateur plus adapté pour estimer correctement $\boldsymbol{h}$ est disponible malgré la proximité de $\boldsymbol{s}$ avec le représentant de la classe initiale.

Enfin, il faut maintenant **mettre à jour les représentants des classes** afin d'espérer que cette nouvelle classification soit faite à partir de S seule. Ceci est fait comme l'étape *Update Step*, c'est à dire, par classe, le nouveau représentant est donné par la moyenne des éléments de S appartenant à la classe.

La distance moyenne entre les représentants de K et les *snapshots* de S de la classe correspondante augmente. En contrepartie, l'erreur d'estimation faite sur H diminue. Ceci est l'effet recherché.

Tout comme les algorithmes basés sur la K-SVD, le nombre de représentants des classes modifié est directement lié avec la faculté de l'algorithme à se défaire des minima locaux au détriment d'une convergence plus chaotique (due à un remaniement brutal des représentants). Enfin, la matrice K permettant l'erreur sur H la plus faible est retenu à la fin des $n_{\text{it}} \in \mathbb{N}^*$ cycles effectués par l'algorithme.

Un critère d'arrêt plus raffiné peut être choisi comme le fait d'effectuer une erreur satisfaisante sur l'estimation de H ou le fait de remarquer que l'erreur ne diminue plus suffisamment rapidement. Comme la majorité des algorithmes conçus lors de cette étude, la nature *greedy* de ce dernier fait qu'il n'est pas garanti que l'erreur diminue à chaque cycle (en particulier quand l'erreur est déjà faible).

Le pseudo-code de l'algorithme GOC se trouve dans l'algorithme 8.

Remarques techniques

Tout comme l'algorithme K-SVD, SOBAL et GOBAL, il faut considérer le cas qui survient lorsqu'une classe (un mode) **n'est pas utilisée**. Il suffit juste de vérifier, après chaque étape Affectation *Goal-Oriented* que chaque classe est bien utilisée. Si ce n'est pas le cas, le représentant de la classe en question est remplacé par une colonne de S, *choisie aléatoirement* afin de permettre à l'algorithme de continuer normalement.

Pour permettre un meilleur brassage des solutions, il a été choisi de ne mettre à jour que 75% des représentants lors d'un cycle.

Pour éviter le phénomène d'*overfitting*, il convient de ne pas utiliser trop d'estimateurs, soit trop de classes n_K. Un problème majeur auquel sont confrontés les utilisateurs de

Algorithm 8 GOC Pseudo-Code

Require: S, H, K_{ini}, n_K, n_{it}

1: **Initialisation** :
2: $\boldsymbol{d}_k \leftarrow$ la $k^{\text{ème}}$ colonne de K_{ini}
3: T_{class} et T_{go} sont des ensembles de n_{snap} éléments
4:
5: **for** $i \in [1 \dots n_{\texttt{it}}]$ **do**
6: **Classification** :
7: **for** $j \in [1 \dots n_{\text{snap}}]$ **do**
8: Calculer la norme $||\boldsymbol{s}_j - \boldsymbol{d}_k||_2$ pour chaque classe k
9: $T_{\text{class}}\{j\} \leftarrow$ numéro de la classe qui minimise la norme précédente
10: **Estimation** :
11: **for** $j \in [1 \dots n_K]$ **do**
12: id $\leftarrow$ indices des *snapshots* de la classe j (obtenue par T_{class})
13: $S_j \leftarrow$ restriction de S aux colonnes indexées par id
14: $H_j \leftarrow$ restriction de H aux colonnes indexées par id
15: $R_j \leftarrow H_j\,(S_j)^+$
16: **Sauvegarde** :
17: Sauvegarder K et les R_j si l'erreur totale sur l'estimation de H diminue
18: **Affectation *Goal Oriented*** :
19: **for** $j \in [1 \dots n_{\text{snap}}]$ **do**
20: Calculer la norme $||\boldsymbol{h}_j - \widehat{\boldsymbol{h}}_j^k||_2$ pour chaque classe k, où $\widehat{\boldsymbol{h}}_j^k = R_k \boldsymbol{s}_j$
21: $T_{\text{go}}\{j\} \leftarrow$ numéro de la classe qui minimise la norme précédente
22: **Mise à jour des représentants** :
23: **for** $j \in [1 \dots n_K]$ **do**
24: id $\leftarrow$ indices des *snapshots* de la classe j (obtenue par T_{go})
25: $S_j \leftarrow$ restriction de S aux colonnes indexées par id
26: $\boldsymbol{d}_j \leftarrow$ la moyenne des éléments de S_j
27:
28: **Sortie** : K et les R_j,$1 \leq j \leq n_K$

l'algorithme K-means est le choix du nombre de classes. On cherche à utiliser le nombre minimal de classes possibles (afin de ne pas faire d'*overfitting*) qui permet cependant de faire une erreur satisfaisante sur la séquence d'entraînement Vu que cette étape de conception se fait off-line, le nombre optimal de classe peut être obtenu par *tests successifs* aux prix d'une phase de conception plus longue.

Extensions

Avant de présenter les résultats d'estimation reliée à GOC, quelques améliorations peuvent être faites.

Tout d'abord, plusieurs méthodes peuvent être considérées pour la mise à jour des classes. Pour chaque *snapshot* $\boldsymbol{s}$, le choix de tester la compatibilité de ce dernier avec chaque classe a été fait. Cette approche est non-optimale et peut être accélérée en ne testant la reconstruction qu'avec les estimateurs des classes **voisines**. Si le *snapshot* $\boldsymbol{s}$ est trop loin du représentant avec l'estimateur optimal, ce *snapshot* n'y sera jamais affecté à cause de la norme utilisée pour effectuer l'*Assignement Step*.

De ce fait, il existe sûrement des **normes bien plus adaptées** pour effectuer cette étape d'affectation. Ceci constitue une piste d'amélioration notable de GOC. Enfin, un point qui sera discuté plus en détail dans la conclusion de ce manuscrit est l'utilisation d'une approche type Kernel pour effectuer l'affectation (7.2).

Ensuite, au prix d'une augmentation de la taille des matrices, il est possible d'utiliser des estimateurs plus performants. Utiliser un **estimateur affine** par classe donne ainsi des résultats bien plus impressionnants, surtout dans le contexte peu de capteurs. Pour chaque classe, en plus de R_j, une certaine composante continue $B_j \in \mathbb{R}^{n_{\boldsymbol{h}}}$ va être déterminée.

L'estimateur utilisé pour la classe j est alors de la forme :

$$\Psi(\boldsymbol{s}) = R_j \boldsymbol{s} + B_j \tag{4.15}$$

La recherche de l'estimateur se fait ainsi en ajoutant une ligne de 1, notée $\mathbf{1}$, à chaque matrice S_j, $1 \leq j \leq n_K$. La matrice $S_{j_{\text{ext}}}^T = \left[S_j^T \mathbf{1}\right]$ est ainsi définie. L'estimateur de la classe j obtenu $H_j \left(S_{j_{\text{ext}}}\right)^+$ peut ensuite être découpé pour obtenir la partie linéaire et la partie constante. En d'autres termes :

$$[R_j B_j] = H_j \left(S_{j_{\text{ext}}}\right)^+ \tag{4.16}$$

Les n_S premières colonnes de $H_j \left(S_{j_{\text{ext}}}\right)^+$ forment la matrice R_j et les colonnes restantes forment B_j.

Dans la même optique, un estimateur **polynomial** peut aussi être obtenu d'une manière aussi aisée. Il suffit de modifier à chaque fois les matrices S_k pour inclure les nouvelles informations disponibles, telle que les mesures mis au carrés. Bien que cette modification réduit naturellement l'erreur faite sur H, l'erreur faite sur une séquence non contenue dans la séquence d'entraînement devient, dans les cas étudiés, non négligeable. La phénomène d'overfitting se fait ainsi grandement ressentir. Cette approche polynomiale ne sera utilisée qu'une seule fois dans le chapitre 6.

4.2.3 Résultats

Cette section permet d'illustrer les performances de GOC. Pour éviter d'alourdir le manuscrit, seule la configuration des capteurs $\mathcal{C}_{\text{opt}}$ est ici retenue. Le placement $\mathcal{C}_{\text{f}}$ sera à nouveau étudié dans la suite de ce chapitre lorsque l'algorithme principal sera présenté.

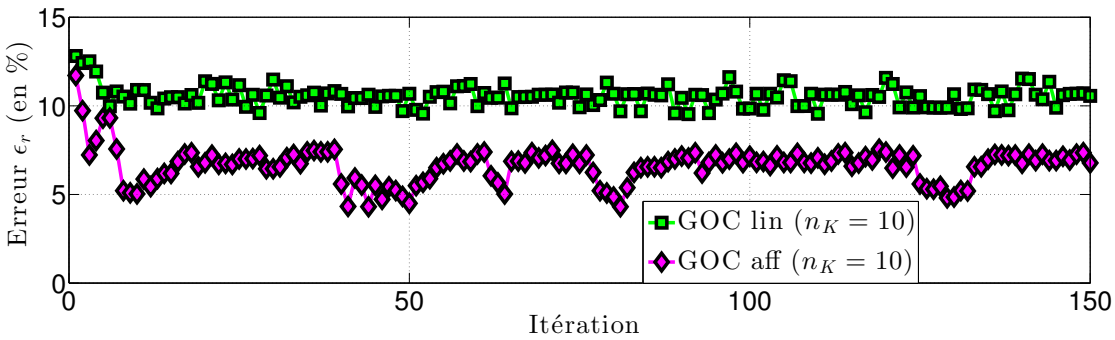

FIGURE 4.1 – GOC - Évolution de ϵ_r

Les représentants initiaux des classes sont choisis aléatoirement parmi les éléments de S. Chaque solution, composée des représentants K et des estimateurs R_j, $1 \leq j \leq n_K$, est celle donnant les meilleures performances d'estimation sur H après 50 lancers de l'algorithme. Chaque lancer est composé de 100 itérations ($n_{\text{it}} = 100$).

Tout d'abord, un exemple d'évolution de l'erreur lors d'un lancer de $n_{\text{it}} = 150$ itérations est illustré sur la figure 4.1. L'évolution chaotique est due à la mise de jour de 75% des représentants des classes par itération. Ceci permet un balayage efficace de l'espace des solutions. Plus de stabilité peut être obtenue en mettant à jour un représentant à la fois, mais le risque de *passer à côté* d'une solution satisfaisante est grand. Sur cette figure, $n_S = 3$ capteurs sont utilisés. L'approche estimateur affine permet bien évidemment de faire moins d'erreur que l'approche linéaire.

GOC est comparé à l'approche POD et à l'approche SOBAL. Quelques précautions sont nécessaires afin de pouvoir comparer ces trois approches dans des situations effectivement comparables.

Concernant la méthode POD, il faut utiliser autant de modes POD que de capteurs soit $n_D = n_S$.

Comme il a été vu dans le chapitre précédent, il faut utiliser une parcimonie SOBAL égale au nombre de capteurs afin d'être dans une situation comparable avec l'approche POD. Ceci impose $K = n_S$ pour SOBAL. Les performances de cette approches peuvent être améliorées en augmentant la valeur du multiplieur n_{mul}, définissant la taille finale du dictionnaire via $n_{\text{mul}}K = n_{\text{mul}}n_S$. Le dictionnaire SOBAL est donc de taille $(n_{\boldsymbol{x}} \times n_{\text{mul}}n_S)$.

Maintenant, afin de pouvoir comparer GOC à SOBAL, le choix a été fait d'avoir la concaténation des matrices d'estimateurs de taille identique au dictionnaire SOBAL. Étant donné que GOC utilise n_K estimateurs de taille $n_{\boldsymbol{x}} \times n_S$, la matrice résultant de cette concaténation d'estimateurs est de taille $n_{\boldsymbol{x}} \times n_K n_S$.

Le nombre de classes doit vérifier : $(n_{\boldsymbol{x}} \times n_K n_S) = (n_{\boldsymbol{x}} \times n_{\text{mul}} n_S)$ d'où

$$n_K = n_{\text{mul}} \tag{4.17}$$

Cependant, il convient de noter que ce cadre **favorise** l'algorithme GOC. Il faut prendre en compte le nombre de modes utilisés pour effectuer la classification. Par exemple, dans le cas $n_S = 1$, l'approche GOBAL utilise un dictionnaire D_2 contenant 20 modes. L'approche GOC avec $n_K = 10$ présente un estimateur contenant 20 colonnes, **mais nécessite en plus** une matrice de classification comportant 10 éléments. Ces derniers sont

Nombres de capteurs n_S	**POD**	**GOBAL** $n_D = n_S$	**GOBAL** $n_D = 3n_S$	**GOBAL** $n_D = 10n_S$	**GOC** $n_K = 1$	**GOC** $n_K = 3$	**GOC** $n_K = 10$
1	64.0%	62.6%	62.6%	62.6%	62.5 %	56.9%	55.2%
2	46.6%	43.8%	34.2%	25.5%	43.8%	31.6%	20.87%
3	20.6%	20.6%	12.0%	9.06%	20.6 %	13.4%	9.60%
4	12.3%	11.5%	8.31%	5.56%	11.5%	8.69%	5.00%
5	3.68%	3.68%	2.67%	2.02%	3.68%	2.75%	1.23%

TABLE 4.1 – GOC - Performances $\mathcal{C}_{\text{opt}}$

certes de tailles plus faibles mais jouent en faveur de GOC.

Le tableau 4.1 permet de comparer l'erreur d'estimation relative ϵ_r obtenue avec les approches POD, GOBAL et GOC (estimateur linéaire). Les cases bleues indiquent que l'approche GOC a permis d'effectuer une erreur inférieure à l'approche GOBAL correspondante ($n_{\text{mul}} = n_K$). Dans tous les cas considérés, l'approche GOC permet d'obtenir une erreur d'estimation proche de l'approche GOBAL avec un algorithme bien plus simple à mettre en œuvre. Cela dit, l'utilisation de trop nombreuses classes ($n_K = 10$) peut causer des problèmes d'*overfitting* et ainsi réduire les performances sur des séquences originales.

La figure 4.2 contient 3 graphes permettant de comparer les approches POD, SOBAL et GOC (estimateur linéaire et affine). Chaque graphe correspond à un nombre donné de classes utilisées. L'erreur représentée par ϵ_r est l'erreur de reconstruction relative, en norme de Frobenius, sur la totalité de la séquence d'entraînement :

$$\epsilon_r = ||H - \widehat{H}||_F / ||H||_F \times 100. \tag{4.18}$$

La taille des estimateurs affines est plus grande que celle des estimateurs linéaires. Il faudrait ainsi réduire le nombre de classes lorsque cette approche est utilisée. Cette courbe est affichée pour information seulement. Elle permet de ressentir que GOC permet d'effectuer ici une meilleure estimation que SOBAL. Utiliser plusieurs estimateurs linéaires est ainsi une approche valide et justifie l'utilisation des classes.

Un comportement particulièrement notable est présent dans le cas $n_K = 1$ / $n_{\text{mul}} = 1$. Les approches SOBAL et GOC linéaire produisent des résultats extrêmement similaires (identiques sur 5 chiffres significatifs). Ensuite, plus le nombre de classes utilisées est grand, meilleurs sont les résultats. Ceci est attendu vu les performances obtenues avec SOBAL lors de l'augmentation de n_{mul}.

L'utilisation d'un estimateur affine est plus efficace qu'un estimateur linéaire.

La figure 4.3 compare ces 3 approches pour une même configuration donnée des capteurs et lorsque le nombre de classes augmente. La position des 3 capteurs n'est pas modifiée. Ce graphique illustre clairement l'avantage d'utiliser plus de classes ainsi qu'un estimateur affine.

Compte tenu des performances de GOC et GOBAL, il semble très prometteur de fusionner les deux approches.

4.3 Classed Based GOBAL - CB.GOBAL

La **contribution principale** de ce chapitre est maintenant présentée. Cette dernière correspond à l'algorithme Classed Based GOBAL (ou encore, GOBAL Structuré), nommé ci-après CB.GOBAL. Cette méthode se trouve dans la même catégorie que GOBAL et permet de répondre au même problème *Pb.GoalOriented*.

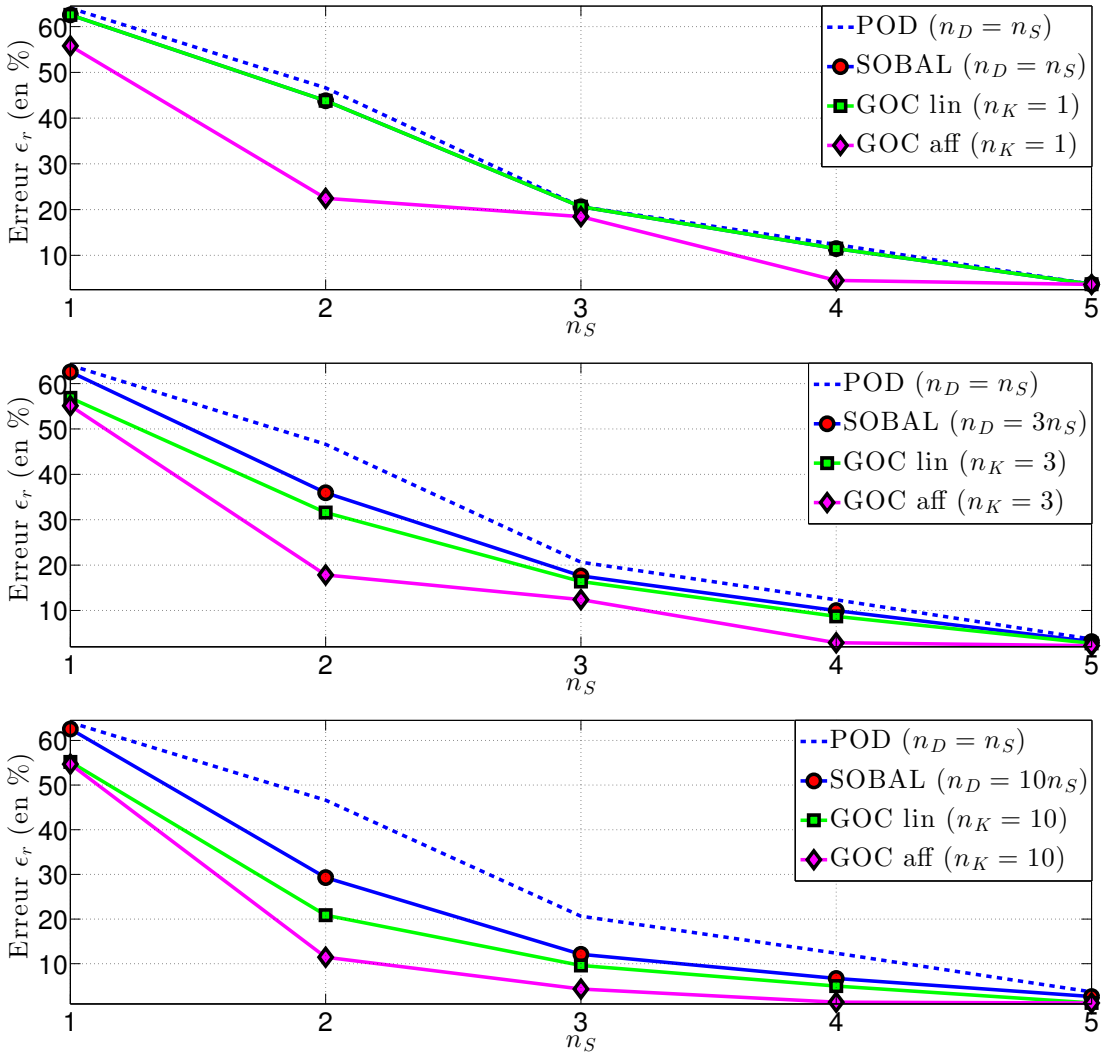

FIGURE 4.2 – GOC - Performances $\mathcal{C}_{\text{opt}}$

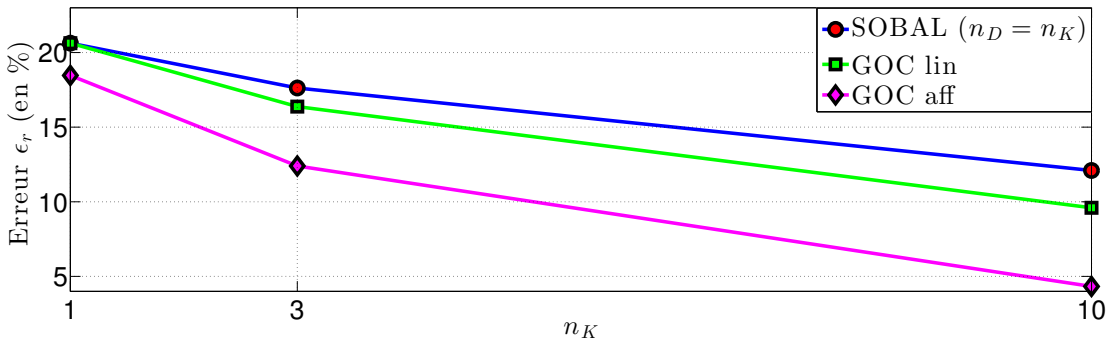

FIGURE 4.3 – GOC - Performances $\mathcal{C}_{\text{opt}_3}$ - Influence de n_K

L'approche GOBAL bénéficie fortement d'un multiplieur n_{mul} élevé. Cependant, bien qu'augmenter n_{mul} permet d'effectuer une erreur d'estimation sur H de plus en plus faible, cette erreur ne diminue pas suffisamment rapidement.

Au lieu de créer des représentations de plus en plus creuses exploitant les mêmes modes, il semble intéressant de créer des **représentations moins creuses mais plus variées** sur leur utilisation modale (via les classes).

Ces idées sont à l'origine de l'algorithme CB.GOBAL. Ce dernier utilisera l'algorithme GOC pour obtenir les classes et GOBAL pour obtenir les estimateurs **pour chaque classe**.

4.3.1 Choix de la structure

Le problème *Pb.GoalOriented* est considéré. Les séquences S et H sont disponibles.

Une fois le nombre de classes n_K choisi, l'algorithme GOC est utilisé afin d'obtenir la matrice de sélection de classes K. Les séquences (S, H) peuvent ensuite être réorganisées par classes pour former les n_K paires (S_k, H_k).

L'objectif est maintenant d'appliquer GOBAL pour chaque couple (S_k, H_k). Chaque classe k se trouve associée à une paire de dictionnaires GOBAL contenant $n_{Dk} \in \mathbb{N}^*$ modes :

— $D_{1k} \in \mathbb{R}^{n_S \times n_{Dk}}$
— $D_{2k} \in \mathbb{R}^{n_{\boldsymbol{h}} \times n_{Dk}}$

où K est la parcimonie de chaque représentation creuse. Cette dernière est choisie égale à $K = n_S$ pour chaque classe, afin d'exploiter le plus d'information possible.

Il est maintenant possible de définir un nombre total de modes GOBAL. Ce dernier est noté $n_T = \sum_{i=1}^{n_K} n_{Dk}$. Enfin le choix a été fait d'ajouter le nombre de classes (le nombre de représentants dans K) à n_T, bien que ces modes $\boldsymbol{d}_k$ soient de nature différentes, afin de former le nombre de modes équivalents utilisé par CB.GOBAL. Ce dernier est noté $n_{T.\text{st}}$ et vaut $n_{T.\text{st}} = n_T + n_K = \sum_{i=1}^{n_K} n_{Dk} + n_K$.

La difficulté principale à surmonter est le choix de n_{Dk}, le nombre de modes utilisés par la classe k. L'ensemble $\mathrm{S}_t = (n_{D1}, n_{D2}, \ldots, n_{Dn_K})$ forme ***la structure*** qui sera utilisée par l'estimateur CB.GOBAL.

En effet, **certaines classes sont plus aptes à profiter de la parcimonie que d'autres**. Il est donc essentiel d'exploiter cette information pour pouvoir obtenir de meilleures propriétés d'estimation. Ensuite, dans l'optique de pouvoir comparer cette approche avec l'approche GOBAL, il faut aussi que $Kn_{\text{mul}} = n_{T.\text{st}}$, où n_{mul} est le multiplicateur utilisée par GOBAL. Ainsi, pour un n_{mul} donné, il faut que la *structure* CB.GOBAL vérifie :

$$\sum_{i=1}^{n_K} n_{Dk} + n_K = Kn_{\text{mul}} \tag{4.19}$$

Un critère approché de *l'aptitude à la parcimonie* d'un couple (S_k, H_k) n'a pas pu être obtenu lors de ce travail de thèse. L'objectif à atteindre était de trouver quelles classes étaient plus aptes à tirer profit d'un grand n_{Dk}.

Finalement, une approche plus lourde d'un point de vue calculatoire a été retenue. **La *structure* $\mathbf{S}_t$ est obtenue de manière itérative**. Cette dernière possède n_K éléments.

Soit S_{t0} la *structure* initiale. Cette dernière est définie par $\mathrm{S}_{t0} = (K, K, \ldots, K)$. Ceci signifie qu'à l'état initial, les *snapshots* de chaque classe sont décrits par des représentations pleines. Soit $n_{\mathrm{inc}} \in \mathbb{N}^*$ l'incrément. L'itération i est associée à la *structure* S_{ti}.

Pour obtenir la *structure* à l'itération $i+1$, l'algorithme GOBAL est appliqué à chaque classe avec pour nombre de modes $n_{Dk}+n_{\mathrm{inc}}$, où n_{Dk} provient de S_{ti}. Ensuite, pour chaque classe, l'erreur d'estimation totale est calculée lorsque l'estimateur d'une seule classe est mis à jour. Le nombre de modes de la classe qui a permis de réduire le plus possible l'erreur d'estimation totale est retenu. Soit $j \in \mathbb{N}^*$ cette classe. La *structure* retenue pour l'itération $i+1$ est $\mathrm{S}_{ti+1} = (n_{D1}, \ldots, n_{Dj} + n_{\mathrm{inc}}, \ldots, n_{Dn_K})$.

Cette démarche s'arrête lorsque $\sum_{i=1}^{n_K} n_{Di}$ atteint la valeur max définie par l'utilisateur. Dans le cas présent, étant donné que CB.GOBAL va être comparé à GOBAL avec un multiplieur n_{mul}, le critère d'arrêt est : $\sum_{i=1}^{n_K} n_{Di} + n_K = Kn_{\mathrm{mul}}$.

Exemple d'obtention d'une *structure*

Cette approche est maintenant illustrée par un **exemple**. CB.GOBAL va être comparé à GOBAL avec un multiplieur $n_{\mathrm{mul}} = 10$. Le nombre de capteurs disponibles est de 2 et ces derniers ont déjà été placés. Le nombre de classes est choisi égal à $n_K = 4$. L'incrément est choisi égal à $n_{\mathrm{inc}} = 1$.

La *structure* initiale est $\mathrm{S}_{t0} = (2, 2, 2, 2)$.

Ensuite, GOBAL est appliqué sur chaque classe avec un nombre de modes de $2+n_{\mathrm{inc}} = 3$. La classe 2 est celle qui permet de minimiser l'erreur d'estimation globale. De ce fait, la *structure* devient $\mathrm{S}_{t1} = (2, 3, 2, 2)$.

Les classes 1, 3 et 4 ont déjà été testées avec un nombre de modes de $2 + n_{\mathrm{inc}} = 3$. De ce fait, seule la classe 2 doit être testée avec $n_{D2} = 3 + n_{\mathrm{inc}} = 4$. Maintenant, c'est le fait d'utiliser la classe 4 avec un $n_{D4} = 3$ qui permet de diminuer l'erreur d'estimation totale. La *structure* est $\mathrm{S}_{t2} = (2, 3, 2, 3)$.

Cette démarche est répétée jusqu'à ce que $\sum_{i=1}^{n_K} n_{Di} = Kn_{\mathrm{mul}} - n_K = 2 * 10 - 4 = 16$. Dans le cas présent, une *structure* finale possible est $\mathrm{S}_{t\mathrm{fin}} = (3, 4, 6, 3)$.

Remarques Techniques

Utiliser un **incrément** de 1 n'est pas toujours conseillé. Il est très utile d'utiliser un incrément $n_{\mathrm{inc}} > 1$ au début de la recherche de la *structure* puis de diminuer ce dernier lorsque l'on se rapproche du critère d'arrêt. Ceci est encore plus vrai lorsque peu de classes sont utilisées.

Le choix du **nombre de classes** est encore un problème à résoudre. Ce dernier dépend des séquences considérées et des capteurs. Pour certaines configurations, une erreur d'estimation totale plus faible peut être obtenue avec de nombreuses classes et peu de parcimonie.

4.3.2 Algorithme

L'algorithme CB.GOBAL peut maintenant être formalisé. Contrairement aux autres algorithmes présentés jusqu'ici, toutes les *briques de bases* ont été présentées. Il ne reste plus qu'à les assembler.

Sans plus de précisions, le pseudo-code se trouve dans l'algorithme 9.

Algorithm 9 CB.GOBAL Pseudo-Code

Require: S, H, n_K, n_{inc}, K , $n_{T.\text{st}}$, n_{it}

1: **Goal Oriented Classification**
2: $K_{\text{ini}} \leftarrow n_K$ *snapshots* distincts aléatoires de S
3: $K \leftarrow \text{GOC}\,(S, H, K_{\text{ini}}, n_K, n_{\text{it}})$
4: (étape répétée jusqu'à ce que l'erreur effectuée par la classification GOC soit satisfaisante pour l'utilisateur)
5:
6: **Initialisation** :
7: $\text{S}_t \leftarrow (n_{D1}, \ldots, n_{Dn_K}) = (K, \ldots, K)$
8: **for** $k \in [1 \ldots n_K]$ **do**
9: $(S_k, H_k) \leftarrow$ restriction de (S, H) aux *snapshots* de la classe k
10: $D_{\text{ini}k} \leftarrow n_{Dk}$ *snapshots* distincts aléatoires de H
11: $(D_{1k}, D_{2k}) \leftarrow \text{GOBAL}\,(S_k, H_k, n_{Dk}, K, D_{\text{ini}k}, n_{\text{it}})$
12:
13: **Recherche de la *structure*** :
14: **while** $\sum_{i=1}^{n_K} n_{Di} < n_{T.\texttt{st}} - n_K$ **do**
15:
16: **for** $l \in [1 \ldots n_K]$ **do**
17: $\text{S}_{ttemp,l} \leftarrow \text{S}_t$, $\text{S}_{ttemp,l}\,[l] \leftarrow \text{S}_{ttemp}\,[l] + n_{\text{inc}}$
18: $D_{\text{ini}} \leftarrow \text{S}_{ttemp}\,[l]$ *snapshots* distincts aléatoires de H
19: $(D_{1temp,l}, D_{2temp,l}) \leftarrow \text{GOBAL}\,(S_l, H_l, \text{S}_{ttemp}\,[l]\,, K, D_{\text{ini}}, n_{\text{it}})$ (si $n_D = \text{S}_{ttemp}\,[l]$ n'a jamais été testé)
20: $\epsilon_l \leftarrow ||H - \widehat{H}||_F$ où $\widehat{H}$ est obtenue à l'aide des dictionnaires (D_{1k}, D_{2k}) pour $1 \leq k \leq n_K, k \neq l$ et $(D_{1temp,l}, D_{2temp,l})$ pour la classe l
21:
22: le l minimisant ϵ_l est retenu
23: $\text{S}_t \leftarrow \text{S}_{ttemp,l}$
24: $(D_{1l}, D_{2l}) \leftarrow (D_{1temp,l}, D_{2temp,l})$
25: Mise à jour des n_{Di} à partir des éléments de S_t
26:
27: **Sortie** : K, S_t, (D_{1k}, D_{2k}),$1 \leq k \leq n_K$

4.3.3 Résultats

Les résultats obtenus à l'aide de CB.GOBAL sont maintenant illustrés. Les positions $\mathcal{C}_{\text{f}}$ et $\mathcal{C}_{\text{opt}}$ des capteurs sont utilisées pour caractériser les performances de CB.GOBAL.

Dans un premier temps, les meilleurs résultats obtenus au fil des expériences sont affichés. Ceci permet de comparer simplement CB.GOBAL à GOBAL. Le nombre de classes n_K utilisées par GOC est ainsi adapté à chaque position des capteurs. Lors des tests, n_K a été choisi entre 3 et 6. Vu le nombre de tests requis, provenant du choix du nombre de classes et des possibilités de *structures* distinctes, seul le cas $n_{\text{mul}} = 10$ est étudié. CB.GOBAL utilise ainsi $n_{T.\text{st}} = n_S n_{\text{mul}}$ modes équivalents (incluant les n_K modes permettant la classification) et est ainsi comparé à GOBAL avec $n_{\text{mul}} = 10$.

Les tableaux illustrant les performances de CB.GOBAL contiennent aussi :

— le nombre de classes retenues n_K,
— l'erreur d'estimation obtenue avec l'approche GOC,

Nombres de capteurs n_S	**POD**	**GOBAL** $n_D = 10n_S$	n_K	**GOC**	**CB.GOBAL**	**Structure**
1	72.1%	66.1%	5	58.6%	58.6%	-
2	137%	57.9%	4	53.1%	48.8%	$(5, 4, 4, 3)$
3	110%	40.1%	5	47.3%	38.7%	$(4, 6, 6, 4, 5)$
4	103%	34.1%	5	34.4%	26.1%	$(8, 4, 6, 9, 8)$
5	219%	29.8%	5	30.4%	20.8%	$(7, 9, 9, 9, 11)$

TABLE 4.2 – CB.GOBAL - Performances $\mathcal{C}_{\mathrm{f}}$

Nombres de capteurs n_S	**POD**	**GOBAL** $n_D = 10n_S$	n_K	**GOC**	**CB.GOBAL**	**Structure**
1	64.0%	62.6%	5	55.9%	55.9%	-
2	46.6%	25.5%	5	27.3%	21.6%	$(2, 3, 3, 4, 3)$
3	20.6%	9.06%	4	13.4%	7.68%	$(8, 8, 6, 4)$
4	12.3%	5.56%	4	7.28%	3.63%	$(6, 7, 13, 10)$
5	3.68%	2.02%	5	2.25%	1.51%	$(14, 5, 5, 7, 14)$

TABLE 4.3 – CB.GOBAL - Performances $\mathcal{C}_{\mathrm{opt}}$

— la *structure* finale utilisée

Dans une dernière sous-section, des compléments sur les *structures* obtenues seront donnés. L'objectif premier est de montrer que les *structures* produites par cette approche sont satisfaisantes.

Positions $\mathcal{C}_{\mathbf{f}}$

Le tableau 4.2 contient les meilleurs résultats obtenus avec l'approche CB.GOBAL sur les positions $\mathcal{C}_{\mathrm{f}}$. La *structure* n'importe pas lorsque la représentation utilisée est 1-*sparse*. Le gain ne provient que de la classification et pas de la parcimonie.

Cette fois-ci, GOBAL est correctement comparé à une approche à base de classification. Le nombre total de modes de CB.GOBAL est réduit pour inclure les modes de classification. Les performances de reconstruction sont remarquables malgré l'utilisation de capteurs mal placés.

Positions $\mathcal{C}_{\mathbf{opt}}$

Le tableau 4.3 contient les meilleurs résultats obtenus avec l'approche CB.GOBAL sur la configuation des capteurs "Optimaux".

Une fois de plus, les performances obtenues avec CB.GOBAL sont très impressionnantes. La *structure* n'importe pas lorsque $n_S = 1$ puisque chaque représentation est 1-*sparse*. Contrairement à l'approche GOC, de très bonnes performances sont obtenues avec l'utilisation de moins de modes. Un compromis doit être trouvé entre la parcimonie utilisée dans chaque classe et le nombre de classes. Il est ici remarquable que l'utilisation de 4 ou 5 classes permet de réduire l'erreur d'estimation sans augmenter le nombre de modes utilisés.

Compléments sur les *structures*

Avant de conclure sur ce chapitre, il convient de noter que de nombreuses *structures* **différentes** ont été trouvées pour une même configuration. Les *structures* données dans

les parties précédentes ne l'ont été que pour illustrer l'approche CB.GOBAL. Elles dépendent entièrement de la classification retenue ainsi que des résultats des estimations GOBAL réalisées dans CB.GOBAL.

Par exemple, pour $\mathcal{C}_{\text{opt}_4}$, de nombreuses solutions acceptables ont été obtenues. GOBAL $n_{\text{mul}} = 10$ permettait de faire 5.56% d'erreur sur ce cas. GOC $n_K = 4$ permet d'effectuer 7.28% d'erreur et GOC $n_K = 5$ permet d'en effectuer 6.98%. Ces deux erreurs sont plus grandes que celles effectuées par GOBAL. Ceci est attendu vu le nombre de "modes" mis en jeu (plus de modes GOBAL que GOC). Cependant, $n_K = 4$ permettra d'avoir plus de "modes libres" que $n_K = 5$ et ainsi laisse une plus grande marge de manœuvre à CB.GOBAL.

Avec $n_K = 4$, les *structures* suivantes ont été obtenues (à partir du même dictionnaire de classification) :
— $(8, 8, 8, 12)$ associée à l'erreur $\epsilon = 3.93\%$
— $(6, 10, 11, 9)$ associée à l'erreur $\epsilon = 3.64\%$
— $(7, 7, 13, 10)$ associée à l'erreur $\epsilon = 3.63\%$
Chaque classe est ici propice à la parcimonie, avec une préférence pour les classes $2, 3$ et 4.

Avec $n_K = 5$, les *structures* suivantes ont été obtenues (à partir du même dictionnaire de classification) :
— $(7, 10, 7, 7, 4)$ associée à l'erreur $\epsilon = 3.98\%$
— $(7, 7, 10, 7, 4)$ associée à l'erreur $\epsilon = 3.64\%$
— $(7, 7, 10, 7, 4)$ associée à l'erreur $\epsilon = 3.75\%$
Les classes 2 et 3 sont ici plus propices à la parcimonie que les autres.

Ces *structures* font apparaître que certaines classes sont plus aptes à être décrites par une représentation creuse que d'autres. Ceci est très difficile à prévoir à partir des classes (H_i, S_i) seules.

4.4 Conclusion

Ce chapitre a permis de présenter de manière succincte une approche permettant d'associer l'algorithme GOBAL et la classification, essentielle en Machine Learning. L'algorithme GOC, basé sur K-means, permet d'obtenir des représentants pour effectuer une classification simple, via un test de proximité. Ce dernier diffère cependant de K-means dans le fait que les classes sont créées en vue de faire la plus faible erreur d'estimation possible sur une séquence **différente** de celle utilisée pour créer les mesures. Ce potentiel, non illustré dans ce chapitre, le sera dans le chapitre 6.

Ces classes ont ensuite été exploitées par l'algorithme GOBAL, couplé à une approche permettant de choisir la taille des dictionnaires associés à ces classes. Ces tailles de dictionnaires forment ce qui a été nommé une *structure*. Cette dernière est ainsi adaptée aux classes obtenues et de ce fait, au positionnement retenu des capteurs.

Cette approche générale a été regroupée sous un seul algorithme nommé CB.GOBAL. De nombreuses améliorations peuvent être faites vu la modularité de cet algorithme.

Enfin, ce chapitre clot définitivement la production des algorithmes orientés pour la création de dictionnaires exploitables en temps réel. Le chapitre suivant va permettre d'éclaircir l'origine des capteurs dit "Optimaux" utilisés tout au long de ce manuscrit.

Bibliographie

[1] D.a. Adeniyi, Z. Wei, and Y. Yongquan. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. *Applied Computing and Informatics*, 12(1) :90–108, 2016.

[2] Andrés E. Coca and Liang Zhao. Musical rhythmic pattern extraction using relevance of communities in networks. *Information Sciences*, 329 :819–848, 2016.

[3] Xuan Cheng, Yuanli Feng, Ming Zeng, and Xinguo Liu. Video segmentation with L0 gradient minimization. *Computers & Graphics*, 54 :38–46, 2016.

[4] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. 2005.

[5] Ivica Vilibić, Hrvoje Mihanović, Grozdan Kušpilić, Ante Ivčević, and Vesna Milun. Mapping of oceanographic properties along a middle Adriatic transect using Self-Organising Maps. *Estuarine, Coastal and Shelf Science*, 163 :84–92, 2015.

[6] Jarbas Joaci De Mesquita Sa Junior and Andre Ricardo Backes. ELM based signature for texture classification, 2015.

[7] Guang-Bin Huang, Qin-yu Zhu, Chee-kheong Siew, Guang-bin Huang Ã, Qin-yu Zhu, Chee-kheong Siew, Guang-Bin Huang, Qin-yu Zhu, and Chee-kheong Siew. Extreme learning machine : Theory and applications. *Neurocomputing*, 70(1-3) :489–501, 2006.

[8] Bassam Mokbel, Benjamin Paassen, Frank Michael Schleif, and Barbara Hammer. Metric learning for sequences in relational LVQ. *Neurocomputing*, 169 :306–322, 2015.

[9] M. C. Naldi and R. J G B Campello. Comparison of distributed evolutionary k-means clustering algorithms, 2013.

[10] Joe L. Villa Medina, Ricard Boque, and Joan Ferre. Bagged k-nearest neighbours classification with uncertainty in the variables. *Analytica Chimica Acta*, 646(1-2) :62–68, 2009.

[11] Jacek M. Leski. Fuzzy c-ordered-means clustering. *Fuzzy Sets and Systems*, 286 :114–133, 2013.

[12] Sašo Karakatič and Vili Podgorelec. Improved classification with allocation method and multiple classifiers. *Information Fusion*, 31 :26–42, 2016.

[13] Feng Jiang, Guozhu Liu, Junwei Du, and Yuefei Sui. Initialization of K-modes clustering using outlier detection techniques. *Information Sciences*, 332 :167–183, 2016.

[14] James B. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

CHAPITRE 5
Placement des capteurs

Chapitre 5

Placement des capteurs

Contenu

L'objectif de ce chapitre est de déterminer un positionnement *optimal* des capteurs. Jusqu'à maintenant, le problème du choix des capteurs a toujours été mis de côté. Différentes positions ont été retenues afin d'illustrer l'importance d'avoir des capteurs bien placés, en particulier sur la qualité de l'estimation de la quantité d'intérêt. C'est via ces derniers que toute forme d'information est obtenue sur le système lors du fonctionnement en temps réel.

La première partie de ce chapitre va définir le cadre du problème ainsi que l'approche la plus connue pour le résoudre. Les limites de telles approches seront explicitées et permettront ainsi de justifier la proposition d'une nouvelle méthode.

Ensuite, les performances de cette dernière seront étudiées. Les dernières sections serviront à présenter des extensions de la méthode proposée afin de l'adapter aux problèmes du type *Pb.GoalOriented*.

5.1 Cadre

Le problème principal à résoudre est le suivant, nommé *Pb.Capteur*, :

<u>Données</u> : Y, n_S
<u>Objectif</u> : Trouver une application $\Psi(.)$ définie sur $\mathbb{R}^{n_S} \rightarrow \mathbb{R}^{n_{\boldsymbol{x}}}$ ainsi qu'une matrice $C \in \mathcal{M}_C$ qui minimise $\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F$
où $\widehat{Y} = \left(\Psi(\boldsymbol{s}_{t_1}) \ldots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}\right)\right)$ et $S = CY$.

où n_S est le nombre de capteurs à placer, Y est la séquence d'entraînement et $\mathcal{M}_C$ est la structure particulière que doit vérifier C (voir le chapitre 2 paragraphe 2.1).

Toutes les informations concernant les notations peuvent être trouvées dans le chapitre 2. Les mêmes hypothèses de travail sont utilisées. Entre autres, le nombre de capteurs utilisé est *faible*. La différence principale avec le problème *Pb.Estimation* est qu'il faut maintenant trouver le positionnement des n_S capteurs en plus de la méthode d'estimation. Pour un nombre donné n_S de capteurs, le *meilleur* positionnement est recherché. Le couplage entre la position des capteurs et la méthode d'estimation rend le problème très délicat à résoudre. Comme indiqué dans les chapitres 3 et 4, les problèmes *Pb.Estimation* et *Pb.GoalOriented* sont déjà complexes. Rajouter une inconnue supplémentaire influant sur la création des dictionnaires va grandement changer l'approche de résolution.

Les problèmes de placement de capteurs sont présents dans de nombreux domaines comme le contrôle, l'estimation, la détection d'erreurs [1] et la classification [2]. De nouvelles méthodes sont constamment proposées à cause de la complexité de ce problème. Cette difficulté provient du grand nombre de configurations possibles puisque ce problème est de nature combinatoire. La grande diversité des algorithmes actuels de placement de capteurs provient de la manière de parcourir l'espace des solutions ainsi que le critère utilisé pour déterminer la validité d'un positionnement donné.

Dans le domaine du génie civil, l'importance du suivi de l'intégrité des structures est une source de nouveaux algorithmes de placement de capteurs. Peuvent être cités, parmi les plus connus, Effective Independence, Optimal Driving Point, Sensor Set Expansion, Kinetic Energy Method et Variance Method [3, 4]. Une grande majorité de ces algorithmes se base sur la matrice d'information de Fisher, plus connue sous le nom de *Fisher Information Matrix* (FIM), ainsi que sur une base de dimension réduite pour décrire le champ de grande dimension mesuré [5]. Le critère peut-être la maximisation de la plus petite valeur singulière de la FIM par exemple [6]. D'autres critères utilisés sont le Modal Assurance Criteria [7] et le conditionnement de la matrice modale [8]. Un autre algorithme notable est le très récent FrameSense (FS) [9]. Ce dernier exploite de nombreux résultats de la théorie des *Frame* (une généralisation de la notion de base) et permet de garantir de très bonnes propriétés d'estimation.

Des méthodes plus spécifiques sont aussi d'un grand intérêt suivant la nature des informations *a priori* disponibles sur le système. Des informations de nature physique sur une configuration d'écoulement fluide peuvent conduire à un placement plus adapté à déterminer l'état de ce dernier [10]. Un modèle spécifique peut contribuer à un placement optimisé pour la localisation d'une source de nature isotrope [11]. Enfin, l'utilisation de *splines* permet un placement donnant une meilleure reconstruction de réponse de nature sismique [12].

Les méthodes citées précédemment sont toutes issues d'un formalisme *traitement du signal*. D'autre techniques issues de la théorie du contrôle, l'automatique, sont aussi cités. Parmi ces dernières se trouvent celles se basant sur le grammien d'observabilité [13] et les valeurs singulières de la matrice de Hankel [14].

Cela dit, un point très intéressant est que ces méthodes peuvent être adaptées à de nombreux problèmes à l'aide de modifications mineures.

Les critères ci-dessus sont ensuite couplés à une méthode de balayage de l'espace des solutions. La majorité de ces méthodes utilise une approche *greedy* (voir le chapitre 3). Les approches heuristiques telles que les algorithmes génétiques [15] et le recuit simulé (Simulated Annealing) [16] sont aussi utilisées.

5.1.1 Approches Usuelles

EI - Effective Independence

Parmi la myriade d'approches possibles, les travaux de Kammer [17] réalisés sur Effective Independence (EI) doivent être présentés. Cette méthode est extrêmement utilisée de par sa facilité de mise en œuvre et ses performances. Le nom de cette méthode indique déjà l'approche qu'elle utilise. EI fait en sorte que chaque mesure effectuée est la plus possible indépendante possible des autres afin qu'elle puisse contribuer efficacement à l'objectif d'estimation.

Dans le cadre actuel, cette approche se fonde sur l'utilisation de la base POD (voir le chapitre 2). L'idée principale est l'utilisation d'un espace de dimension plus faible pour décrire le champ $\boldsymbol{y}$. Soit Ψ_{n_D} la base POD d'ordre $n_D \in \mathbb{N}^*$ obtenue à partir de Y. Il est rappelé que dans cette base POD, chaque élément $\boldsymbol{y}$ de Y est décrit par le vecteur $\boldsymbol{a}$ comportant $n_D \ll n_{\boldsymbol{x}}$ éléments. Le vecteur de mesure $\boldsymbol{s} = C\boldsymbol{y}$ est utilisé pour obtenir une estimée $\widehat{\boldsymbol{a}}$ de $\boldsymbol{a}$ qui a son tour permet d'obtenir une estimée $\widehat{\boldsymbol{y}} = \Psi_{n_D}\widehat{\boldsymbol{a}}$ de $\boldsymbol{y}$. $\widehat{\boldsymbol{a}}$ est la valeur de $\boldsymbol{a}$ minimisant la norme $||\boldsymbol{s} - C\Psi_{n_D}\boldsymbol{a}||_2$. EI, dans le cadre où les mesures ne sont pas bruitées, se base sur ce contexte. Il faut maintenant trouver où placer ces n_S capteurs. Tout d'abord, pour garantir une solution unique au problème d'estimation, et ainsi assurer sa nature bien posée, le nombre de modes POD est choisi tel que $n_D \leq n_S$. En pratique, le choix $n_D = n_S$ est fait afin d'exploiter le plus de modes possibles et ainsi autoriser une plus faible erreur d'estimation.

La représentation réduite de référence est définie par $\boldsymbol{a} = \Psi_{n_D}^T\boldsymbol{y}$. Elle est obtenue par projection de $\boldsymbol{y}$ sur la base POD d'ordre n_D. Cela revient au cas où toute l'information est disponible ($n_{\boldsymbol{x}}$ capteurs). Elle est aussi obtenue lors de la décomposition SVD aboutissant à la base POD. Ensuite, la représentation estimée est donnée par $\widehat{\boldsymbol{a}} = (C\Psi_{n_D})^+\boldsymbol{s}$.

L'objectif de la majorité des méthodes de placement de capteurs est de placer ces derniers (d'agir sur la matrice C) afin de réduire l'erreur $||\boldsymbol{a} - \widehat{\boldsymbol{a}}||_2$. Le vecteur estimé $\widehat{\boldsymbol{a}}$ peut encore être exprimé en fonction de $\boldsymbol{a}$ par :

$$\widehat{\boldsymbol{a}} = (C\Psi_{n_D})^+\boldsymbol{s} = (C\Psi_{n_D})^+C\boldsymbol{y} = (C\Psi_{n_D})^+C\Psi_{n_D}\boldsymbol{a}. \tag{5.1}$$

Une relation similaire est obtenue entre $\boldsymbol{s}$ et le vecteur de mesure effectivement associé à $\widehat{\boldsymbol{a}}$, nommé $\widehat{\boldsymbol{s}}$. Soit

$$\widehat{\boldsymbol{s}} = C\Psi_{n_D}(C\Psi_{n_D})^+\boldsymbol{s}. \tag{5.2}$$

La matrice de projection P permettant d'affecter la mesure effectuée $\boldsymbol{s}$ à la mesure estimée est :

$$P = C\Psi_{n_D}(C\Psi_{n_D})^+ = C\Psi_{n_D}\left((C\Psi_{n_D})^T(C\Psi_{n_D})\right)^{-1}(C\Psi_{n_D})^T \tag{5.3}$$

La dernière expression est obtenue à partir du développement de la pseudo-inverse rendu possible par le fait que $C\Psi_{n_D}$ est de rang égal au nombre de colonnes ($n_D \leq n_S$).

Des mesures bruitées sont maintenant étudiées. Le bruit sur chaque capteur est considéré blanc (le signal aléatoire mis en jeu est une ligne de la matrice Y), identique et indépendamment distribué (dans leur ensemble) selon une loi gaussienne centrée. La matrice de covariance du bruit est $\Psi_0^2 I_{n_S}$, $\Psi_0^2 \in \mathbb{R}_+$.

$\widehat{\boldsymbol{a}}$ est obtenu à partir du meilleur estimateur linéaire non biaisé. La matrice de covariance de l'erreur d'estimation (au niveau de la représentation réduite) est dans ce cas donnée par [17] :

$$\mathrm{E}\left[(\boldsymbol{a} - \widehat{\boldsymbol{a}})(\boldsymbol{a} - \widehat{\boldsymbol{a}})^T\right] = \left[\frac{1}{\Psi_0^2}(C\Psi_{n_D})^T(C\Psi_{n_D})\right]^{-1} \tag{5.4}$$

Cette matrice de covariance peut encore être vue comme F^{-1} où

$$F = \frac{1}{\Psi_0^2} \left(C\Psi_{n_D}\right)^T \left(C\Psi_{n_D}\right) \tag{5.5}$$

est la **matrice d'information de Fisher**.

Cette matrice est à la base de l'algorithme EI et de nombreuses méthodes de placement de capteurs. L'objectif de EI est de minimiser une norme particulière de la matrice de covariance. Cela revient à maximiser une norme particulière de la matrice d'information de Fisher. Vu l'expression de cette dernière et des hypothèses faites sur le bruit, Ψ_0 n'interviendra pas dans le critère de minimisation. Après plusieurs manipulations, le critère retenu par EI fait intervenir la diagonale de la matrice P (même en présence de bruit).

Afin de faciliter la description de l'algorithme, deux ensembles sont introduits :
— $\mathrm{pos_P}$ contenant les index des positions possibles que peuvent prendre les capteurs,
— $\mathrm{pos_S}$ contenant les index des positions actuelles des capteurs.
Les ensembles ne sont pas ordonnés puisque tous les capteurs sont considérés comme étant identiques. Ces derniers vérifient :

$$\begin{aligned} \mathrm{pos_P} &\subset \{x_1, x_2, \ldots, x_{n_x}\} \\ \mathrm{pos_S} &\subset \{x_1, x_2, \ldots, x_{n_x}\} \\ \mathrm{pos_S} &\subset \mathrm{pos_P} \\ \mathrm{card}\left(\mathrm{pos_S}\right) &= n_S \\ \mathrm{card}\left(\mathrm{pos_P}\right) &= n_P \end{aligned} \tag{5.6}$$

$\mathrm{pos_P}$ permet aussi de prendre en compte les emplacement interdits. Suivant les contraintes que doivent respecter les capteurs, certains emplacements ne peuvent pas être utilisés. Dans la modélisation effectuée, ceci se traduit par des points du maillage qui ne peuvent pas être considérés comme points de mesures. Par exemple, certaines zones peuvent être à des températures trop élevées pour autoriser le bon fonctionnement des capteurs. D'autres zones peuvent aussi être soumises à des pressions trop fortes pour certains capteurs.

Dans le cas présent, les capteurs ne peuvent être posés qu'à la surface du cylindre et pas dans le sillage de l'écoulement. Ils doivent être fixés sur une surface solide. D'autres contraintes seront considérées dans la suite, comme l'impossibilité de placer des capteurs dans le voisinage immédiat des actionneurs ainsi que l'impossibilité de placer des capteurs trop proches les uns des autres.

L'approche utilisée par EI peut maintenant être décrite. Un capteur est posé sur chaque position possible. Cette étape d'initialisation fait que $\mathrm{pos_S} = \mathrm{pos_P}$. Ensuite, un par un, les capteurs de $\mathrm{pos_S}$ sont retirés afin d'arriver au nombre requis de n_S capteurs. Chaque capteur est associé à un élément de la diagonale de la matrice P. Cette matrice permet de relier les mesures réelles aux mesures qui seront reconstruites, via la représentation réduite. Le capteur dont l'élément diagonal de P est le plus petit est retiré. La matrice P est ensuite recalculée de par la modification de la matrice C. Vu la structure de C, cela revient à retirer une ligne de Ψ_{n_D} pour chaque itération effectuée.

Cette approche est en effet *greedy* puisqu'une solution globale est espérée à partir de modifications locales (retrait d'un capteur à chaque itération).

Cette approche est aussi qualifiée de *Worst One Out*. Plus de n_S capteurs sont initialement placés puis, un par un, les capteurs sont retirés. Une autre approche qui doit être mentionnée est son opposée, le *Best One In*. Comme son nom l'indique, les capteurs

sont placés un par un à partir d'une configuration sans capteurs. Le meilleur capteur selon un critère donné est retenu puis placé de manière définitive. L'approche *Best One In* est moins performante que l'approche *Worst One Out* dans une grande partie des cas.

Une implémentation numérique efficace de ce code peut être trouvée dans [18]. Le temps de calcul nécessaire est alors réduit par rapport au temps de calcul de l'implémentation de *base*.

Le pseudo-code de l'algorithme EI se situe dans l'algorithme 10.

Algorithm 10 Effective Independence Pseudo-Code

Require: Y, n_S, pos_P
1: **Initialisation** :
2: $n_D \leftarrow n_S$
3: $\Psi_{n_D} \leftarrow$ base POD de Y d'ordre n_D
4: $\text{pos}_\text{S} \leftarrow \text{pos}_\text{P}$ (un capteur est placé sur chaque position admissible)
5:
6: **while** $\text{card}\,(\text{pos}_\text{S}) \neq n_S$ **do**
7: **Boucle principale** :
8: $C \leftarrow$ les positions indexées par pos_S
9: $P \leftarrow C\Psi_{n_D}\left((C\Psi_{n_D})^T(C\Psi_{n_D})\right)^{-1}(C\Psi_{n_D})^T$
10: $p \leftarrow \text{Diag}\,(P)$
11: retirer le capteur de pos_S (la ligne de C) associé à l'index de la plus petite valeur de p
12:
13: **Sortie** : $C \leftarrow$ les positions indexées par pos_S

FS - FrameSense

Une autre méthode récente et affichant de très bonnes propriétés d'estimation est FrameSense [9]. Cet algorithme de type *Worst One Out* trouve ses origines dans la théorie des *Frame*.

La matrice $\Phi \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_D}$ est une *frame* de $\mathbb{R}$ s'il existe $(a, b) \in \mathbb{R}_+^{*\,2}$ tel que :

$$a||\boldsymbol{x}||_2^2 \leq ||\Phi\boldsymbol{x}||_2^2 \leq b||\boldsymbol{x}||_2^2 \quad ,\forall \boldsymbol{x} \in \mathbb{R} \tag{5.7}$$

Le vecteur à estimer vérifie :

$$\boldsymbol{y} = \Phi\boldsymbol{a} \tag{5.8}$$

où $\boldsymbol{a} \in \mathbb{R}^{n_D}$ est la représentation réduite de $\boldsymbol{y}$.

La mesure associée à $\boldsymbol{y}$ vérifie :

$$\boldsymbol{s} = C\Phi\boldsymbol{a} \tag{5.9}$$

où C vérifie les contraintes définies précédemment.

L'objectif de FrameSense est de trouver la matrice $\Phi_C = C\Phi$ qui permet d'effectuer le moins d'erreur d'estimation sur $\boldsymbol{a}$ à partir de $\boldsymbol{s}$. Tout ceci en ne modifiant que C, c'est-à-dire en choisissant une bonne restriction de Φ. Il s'avère que cette erreur d'estimation peut s'exprimer en fonction du **Frame Potential** définie comme :

$$\text{FP}\,(\Phi_C) = \text{Trace}\left((\Phi_C)^+\,\Phi_C\right) = \sum_{k=1}^{n_D} |\lambda_k|^2 \tag{5.10}$$

où les λ_k sont les valeurs propres de $(\Phi_C)^+ \Phi_C$.

Enfin, l'algorithme FrameSense trouve la matrice Φ_C en retirant une ligne de Φ lors de chaque cycle. La ligne qui, une fois retirée de Φ, permet de maximiser le Frame Potential de cette restriction de Φ, est effectivement retirée. Chaque ligne est ainsi testée.

FrameSense manipule la trace de la matrice $((C\Psi_{n_D})^T(C\Psi_{n_D}))^T(C\Psi_{n_D})^T(C\Psi_{n_D})$ qui présente quelques similarités avec la matrice principale manipulée par EI. Le pseudo-code de FS et bien plus d'explications sur ce dernier peuvent être trouvés dans [9].

5.1.2 Limitations

Mesures réelles

Malgré la popularité et les avantages de EI, cette méthode s'avère limitée dans le cas où très peu de capteurs sont utilisés.

Pour illustrer ce problème, un champ $\boldsymbol{y}$ est considéré ainsi que son approximation sur la base POD d'ordre n_D noté $\boldsymbol{y}_{n_D}$. Pour rappel, avec Ψ_{n_D} cette base POD (associée à la séquence Y), $\boldsymbol{y}_{n_D}$ est obtenue par :

$$\boldsymbol{y}_{n_D} = \Psi_{n_D}\Psi_{n_D}^T\boldsymbol{y}. \tag{5.11}$$

Le champ $\boldsymbol{y}$ peut ensuite s'écrire selon :

$$\boldsymbol{y} = \boldsymbol{y}_{n_D} + \delta\boldsymbol{y} \tag{5.12}$$

où $\delta\boldsymbol{y}$ correspond à la partie du champ qui ne peut pas être décrite par la base POD. Cela dit, par rapport à d'autres bases de décomposition, la base POD garantit un terme $||\delta\boldsymbol{y}||_2$ minimal.

La mesure effectuée pour un placement des capteurs représenté par C est

$$\boldsymbol{s} = C\boldsymbol{y} = \boldsymbol{s}_{n_D} + \delta\boldsymbol{s} \tag{5.13}$$

où

— $\boldsymbol{s}_{n_D} = C\boldsymbol{y}_{n_D}$
— $\delta\boldsymbol{s} = C\delta\boldsymbol{y}$

EI considère que la mesure effectuée sera $\boldsymbol{s}_{n_D}$ au lieu de $\boldsymbol{s}$, puisque cette méthode se base sur $\boldsymbol{a}$. Le placement des capteurs résultants ne peut donc pas prendre en compte un $\delta\boldsymbol{s}$ non nul. Vu que $||\delta\boldsymbol{s}||_2 \leq ||C||_2||\delta\boldsymbol{y}||_2$, plus le nombre de modes POD utilisés est grand, plus $||\delta\boldsymbol{y}||_2$ est faible et de ce fait, plus $||\delta\boldsymbol{s}||_2$ est faible. Cependant, dans le cas *peu de capteurs*, $\delta\boldsymbol{y}$ n'est pas garanti d'être *suffisamment faible* et le $\delta\boldsymbol{s}$ résultant peut grandement nuire à l'estimation. Il devient alors intéressant de porter plus d'intérêt aux mesures réellement faites $\boldsymbol{s}$ au lieu de $\boldsymbol{s}_{n_D}$. Le placement des capteurs découlant de ce choix sera *a priori* différent de l'approche EI.

Un ressenti similaire peut être obtenu à travers l'estimateur issu du placement EI. Le champ estimé s'écrit $\widehat{\boldsymbol{y}} = \Psi_{n_D}(C\Psi_{n_D})^+\boldsymbol{s}$. L'estimateur utilisé par EI est représenté par la matrice $\Psi_{n_D}(C\Psi_{n_D})^+$. Modifier la structure de l'estimateur pour tenter de relier $\boldsymbol{s}$ à $\boldsymbol{y}$ est une approche valide dans le cas peu de capteurs où $\delta\boldsymbol{s}$ n'est pas à négliger. Utiliser une autre matrice que la base POD d'ordre n_D peut être une solution.

Pour récapituler, deux sources d'erreur sont à considérer lors de l'estimation de $\boldsymbol{y}$ avec l'approche EI. La première provient du fait que la base POD d'ordre n_D ne peut pas

parfaitement décrire $\boldsymbol{y}$ à cause du nombre fini de modes retenus. La seconde découle des mesures effectuées sur $\boldsymbol{s}$ qui ne peuvent pas permettre d'obtenir la représentation réduite associée à $\boldsymbol{y}_{n_D}$. Ainsi, à partir de $\boldsymbol{s}$, une erreur est faite sur l'obtention de $\boldsymbol{a}$ et elle est ensuite amplifiée sur l'estimée $\widehat{\boldsymbol{y}}$. La base POD permet la meilleure description de $\boldsymbol{y}$ sur un nombre fini de modes mais, lorsque la chaîne complète est considérée, il peut être intéressant d'en utiliser une autre (pour l'estimation et le placement des capteurs).

Contraintes de placement

Une autre limitation d'EI et des méthodes générales de placement de capteurs est leur prise en compte des contraintes de placement. Une fois la base POD déterminée, ces méthodes la restreignent pour indiquer quels sont les emplacements possibles des capteurs. Bien que ceci soit une approche rapide pour prendre en compte des contraintes de placement, elle rend la base de décomposition choisie *non optimale*. Soient Y la séquence d'entraînement retenue et Ψ_{n_D} la base POD associée. La restriction de Ψ_{n_D} aux lignes associées à des positions mesurables, notée $\Psi_{n_{D_r}}$ n'est plus la base POD de la restriction de Y à ces mêmes lignes. Étant donné que la méthode EI est basée sur l'utilisation de $\Psi_{n_{D_r}}$, une autre approche prenant en compte des contraintes de placement avant le calcul de la base POD présente serait plus performante.

Enfin, l'introduction de contraintes liées à la géométrie des capteurs est difficilement prise en compte avec des approches type EI. Ceci provient du fait que le placement d'un capteur influe sur la position possible des autres capteurs. Comme EI est une approche *Worst One Out*, une fois qu'il ne reste plus que les n_S désirés capteurs, ces derniers peuvent être trop proches les uns des autres et ainsi interdire une mise en place réelle. Une solution consiste à appliquer EI plusieurs fois en modifiant successivement l'espace $\mathrm{pos_P}$ de départ pour prendre en compte la géométrie des capteurs. En effet, après un premier lancer, EI fournit une liste de positions retenues des capteurs. Si deux capteurs sont trop proches, un seul de ces dernier est retenu et figé. Son voisinage immédiat est ensuite retiré de l'ensemble $\mathrm{pos_P}$ initial et l'algorithme EI est lancé avec la recherche de $n_S - 1$ capteurs. Cette approche est répétée jusqu'à ce que les placements obtenus soient satisfaisants. Il est clair que cette modification de l'algorithme EI l'a rendu non déterministe suivant quel capteur est retenu. Il est ainsi souhaitable de lancer cette modification de EI plusieurs fois pour augmenter les chances de trouver une bonne solution.

Ces limitations incitent à proposer une nouvelle approche de placement de capteurs. Cette méthode doit être adaptée à l'utilisation d'un très faible nombre de capteurs et à la prise en compte de contraintes (de placement et de géométrie). Étant donné l'utilisation future de ces positions de capteurs avec les algorithmes de type SOBAL, la motivation de se détacher d'une approche POD est grande.

5.2 SensorSpace

Avant de développer la méthode retenue, il convient de noter que des modifications des algorithmes SOBAL et GOBAL (voir le chapitre 3) pour prendre en compte une matrice C variable ont été tentées. Le couplage fort entre C et les dictionnaires issus de ces algorithmes impose l'introduction de limites sur les variations de C (d'un cycle SC-CU à l'autre) afin de permettre aux algorithmes précédents de *converger*. Cependant, même avec de telles modifications, les résultats obtenus sont décevants. Il faudrait ainsi repenser la structure de ces algorithmes pour intégrer C variable.

Le choix qui résulte de ces premiers échecs est, tout comme Effective Independence, d'utiliser un estimateur bien plus *simple* afin de se focaliser sur le choix des capteurs. L'estimateur devra cependant avoir certains points communs avec les approches conçues jusqu'à présent. Ceci permettra ensuite d'appliquer avec succès un des estimateurs complexes proposés. Il résulte de cette contrainte qu'un estimateur POD *classique* ne peut pas être utilisé. L'accent va être mis sur la séquence d'apprentissage Y et sur les mesures.

Cet algorithme a été nommé SENSORSPACE pour insister sur le fait qu'il va directement manipuler ces mesures, exprimée par CY pour un placement des capteurs décrit par C.

5.2.1 Algorithme

Le cas non bruité est tout d'abord considéré afin de présenter plus clairement le fonctionnement de l'algorithme. Ce dernier doit répondre au problème *Pb.Capteur* et surmonter les limitations d'EI.

Le critère $||Y - \widehat{Y}||_F$ est utilisé par l'algorithme directement. Trouver une estimée $\widehat{Y}$ qui minimise la norme ci-dessus revient à minimiser $\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F$, l'erreur de reconstruction relative sur la séquence d'entraînement, un des critères d'intérêt présenté dans le chapitre 2. Aucune information physique supplémentaire sur le système n'est prise en compte. Ce faisant, cet algorithme peut déjà être qualifié de *Machine Learning* puisqu'il se base seulement sur la séquence d'entraînement totale. Y n'est pas remplacé par une base POD d'ordre réduit. Ceci va bien évidemment engendrer une plus grande complexité de calcul.

Un estimateur linéaire est aussi retenu afin de pouvoir facilement prendre en compte des modifications de l'emplacement des capteurs C. Soit $R \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_S}$ l'estimateur linéaire utilisé. Il doit agir sur une mesure $\boldsymbol{s}$ et produire une estimée $\widehat{\boldsymbol{y}}$ proche de $\boldsymbol{y}$ en norme 2.

Le critère devient alors :

$$C = \underset{\tilde{C} \in \mathcal{M}_C}{\arg\min} ||Y - R\tilde{C}Y||_F, \tag{5.14}$$

où $\mathcal{M}_C$ est l'ensemble des matrices C décrivant une disposition des capteurs (voir 2.1).

Pour un emplacement des capteurs C, la norme ci-dessus peut être minimisée par un choix convenable d'estimateur linéaire. Une fois de plus, l'expression d'un tel estimateur est donnée par la pseudo-inverse de Moore-Penrose selon :

$$R_{\text{opt}} = Y\,(CY)^+ \tag{5.15}$$

Il convient de remarquer que cet estimateur n'est pas unique. Dans la suite, R_{opt} sera noté R pour ne par alourdir les notations.

Le critère prend presque sa forme définitive :

$$C = \underset{\tilde{C} \in \mathcal{M}_C}{\arg\min} ||Y - Y\left(\tilde{C}Y\right)^+ \tilde{C}Y||_F \tag{5.16}$$

La seule inconnue est bien le positionnement des capteurs représenté par C. Les mesures réelles (non bruitées), représentées par CY, sont utilisées directement. Il n'y a plus la contrainte concernant le nombre de modes POD retenus puisqu'il n'y a plus de base POD à calculer.

Comparaison Initiale

Avant de proposer une manière de s'approcher d'un tel C minimisant le critère vu en 5.16, il est déjà intéressant de comparer de manière quantitative cette approche avec EI.

Tout comme EI, cette approche permet d'obtenir un placement des capteurs C et un estimateur. Cet estimateur joue un rôle prédominant dans la détermination de C vu les critères retenus. Soit C associée à une position possible des capteurs. L'estimateur utilisé par EI s'écrit encore $R_{\text{POD}} = \Psi_{n_D} \left(C\Psi_{n_D}\right)^+$. L'erreur faite sur la séquence totale d'entraînement $\epsilon = ||Y - \widehat{Y}||_F$ devient $\epsilon = ||Y - R_{\text{POD}}CY||_F$.

Pour toute matrice $L \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_S}$, la relation suivante est obtenue d'après les propriétés de la pseudo-inverse de Moore-Penrose :

$$||Y - LCY||_F \geq ||Y - Y\left(CY\right)^+ CY||_F \tag{5.17}$$

L'estimateur R utilisé par SensorSpace verifie alors :

$$||Y - R_{\text{POD}}CY||_F \geq ||Y - Y\left(CY\right)^+ CY||_F = ||Y - RCY||_F \tag{5.18}$$

Ainsi, quelle que soit la position retenue des capteurs, l'estimateur utilisé par EI délivrera toujours une estimée de moins bonne qualité (du point de vue ϵ_r) que l'estimateur SensorSpace. Ceci est vrai même pour une position des capteurs donnée par EI. Il suffirait d'utiliser EI puis de modifier l'estimateur utilisé pour obtenir un gain en performance d'estimation.

Comme SensorSpace utilise un estimateur exploitant la totalité de la séquence d'entraînement sous sa forme brute, les positions obtenues des capteurs sont espérées être plus performantes. Cette utilisation directe des matrices Y et $S = CY$ est similaire aux approches de types SOBAL, GOBAL et CB.GOBAL. L'estimateur R utilisé par SensorSpace *ressemble* aux estimateurs $R_i \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_S}$ (où i représente le support retenu du dictionnaire) manipulés par ces approches *sparse*. Les positions SensorSpace sont ainsi espérées être **compatibles** avec ces approches performantes.

Cependant, chaque approche *sparse* utilise plusieurs dictionnaires de type R_i. Ces derniers sont tous *couplés* via le (les) dictionnaire(s) produit(s). Les performances d'estimation de ces méthodes peuvent alors être encore meilleures que SensorSpace.

Minimisation du critère

Maintenant que le critère a été défini et que cette méthode semble très propice à délivrer des positions de capteurs performantes, il faut décrire la manière de trouver une solution C. Le problème de placement des capteurs est toujours de type combinatoire et, de ce fait, de complexité dite *NP-hard*. Une approche *greedy* est une fois de plus retenue (tout comme EI, FrameSense et de nombreux autres méthodes de placement de capteurs).

Des approches *greedy* de type *Worst One Out* comme EI et *Best One In* ne sont pas adaptées pour prendre en compte des contraintes de placement. Dans ce cadre, elles sont moins performantes que des mises à jour successives des capteurs. Les ensembles pos_P et pos_S sont une fois de plus utilisés (voir 5.6). L'ensemble des positions possibles pos_P est modifié en permanence afin d'incorporer les contraintes de placement.

Le déroulement de l'algorithme est maintenant explicité. Tout d'abord, pos_P est initialisé pour contenir l'index des positions possibles, sans prise en compte des contraintes

géométriques. Ensuite, n_S index distincts de pos_P sont choisis pour former pos_S. A cette étape, les index dans pos_S sont modifiés afin de vérifier les contraintes géométriques.

Ces contraintes consistent à prendre en compte l'extension spatiale des capteurs. Suivant les séquences d'entraînement, certaines méthodes ont tendance à placer des capteurs très proches les uns des autres. Un tel placement peut s'avérer impossible à réaliser en pratique. Pour éviter ce genre de situation, un voisinage est défini autour de chaque capteur. Dans un tel voisinage, il y a interdiction de placer un autre capteur. Dans le cas de l'initialisation de pos_S, si des capteurs sont trop proches les uns des autres, ces derniers sont retirés puis replacés aléatoirement selon la liste pos_P.

La boucle principale, nommée cycle, consiste en la mise à jour de chaque capteur de pos_S, dans un ordre aléatoire. Si la totalité des n_S capteurs n'est pas déplacée lors d'un cycle, un minimum local a été trouvé et l'algorithme est interrompu. Tout comme les algorithmes K-SVD, GOBAL et CB.GOBAL, un autre critère d'arrêt peut bien évidemment être retenu, comme la variation de l'erreur d'estimation totale.

La mise à jour d'un capteur consiste à retirer celui-ci de pos_S. Ensuite, pos_P est modifié pour inclure la nouvelle liste des positions possibles. En particulier, les voisinages de chaque capteur sont interdits, donc retirés de la liste pos_P. Le capteur retiré est ensuite placé sur chaque position de pos_P et l'erreur d'estimation sur la séquence d'entraînement totale ϵ est calculée. Cette dernière évaluation nécessite alors le calcul de l'estimateur R pour **chaque** position testée des capteurs. Enfin, le capteur est placé dans la position qui minimise l'erreur ϵ. Un autre capteur est choisi aléatoirement puis un nouveau cycle recommence avec le retrait de ce capteur. La phase de mise à jour est ainsi répétée jusqu'à ce que les positions trouvées ne puissent plus être améliorées.

La figure 5.1 illustre cette procédure de mise à jour de $n_S = 3$ capteurs SENSORSPACE. Cette procédure est :

— (a) : Placement de $n_S = 3$ capteurs de manière aléatoire sur les positions possibles
— (b) : Choix aléatoire d'un capteur (rouge) et retrait de ce dernier
— (c) : Définition de la nouvelle zone possible (vert) pour le repositionnement de ce capteur
— (d) : Placement du capteur retiré (bleu) dans la position permettant de minimiser l'erreur d'estimation ϵ
— (e) : Choix aléatoire d'un capteur (rouge) n'ayant pas été modifié ce cycle puis retrait de ce dernier

Avant de présenter le pseudo-code de l'algorithme SENSORSPACE, il est nécessaire de de modifier la phase d'évaluation de l'erreur ϵ. En effet, cette dernière est répétée de très nombreuses fois et constitue l'opération *coûteuse* de l'algorithme.

Évaluation efficace du critère

Pour une position donnée C des capteurs, caractérisée par l'état pos_S, le critère à évaluer est :

$$\epsilon = ||Y - Y\,(CY)^{+}\,CY||_F \tag{5.19}$$

La première modification permet une réduction considérable du temps de calcul. Cette dernière consiste à utiliser la décomposition en valeur singulière (SVD) de Y (*i.e.*, la base POD obtenue avec la méthode des *snapshots*). Il convient d'insister ici que cette modification ne revient pas à utiliser l'approche POD exploitée par EI. En effet, le nombre de modes POD n'est plus contraint d'être égal à n_S. Soit $n_0 \in \mathbb{N}^*$ le nombre de modes POD de Y retenus pour cette simplification du critère . La décomposition SVD réduite d'ordre n_{snap} de Y s'écrit : $Y \approx U\Sigma V^T$ où :

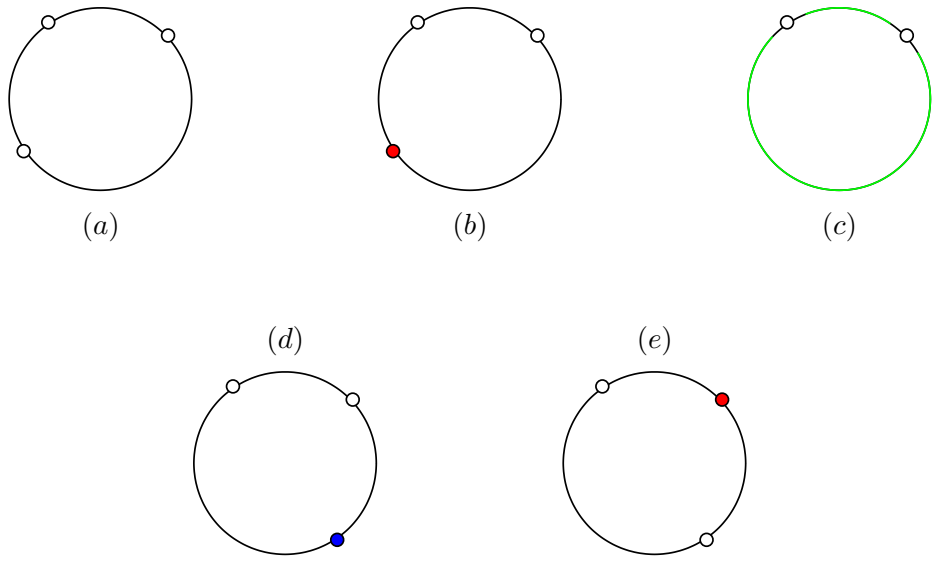

FIGURE 5.1 – SENSORSPACE - Illustration de la procédure de mise à jour des capteurs

— U est une matrice orthonormale de dimension $n_{\boldsymbol{x}} \times n_{\text{snap}}$ contenant les modes POD
— Σ est une matrice carrée diagonale de dimension $n_{\text{snap}} \times n_{\text{snap}}$ contenant les valeurs singulières
— V est une matrice orthonormale de dimension $n_{\boldsymbol{x}} \times n_{\text{snap}}$

ΣV^T contient les décompositions réduites de Y sur la base POD U. n_0 est choisi afin de vérifier $n_0 > n_S$, différenciant ainsi l'approche SENSORSPACE de l'approche EI. La limitation d'utiliser autant de modes POD que de capteurs n'est pas applicable à cette nouvelle approche. n_0 doit en fait permettre d'approximer convenablement Y en minimisant $||Y - Y_{n_0}||_F$, où Y_{n_0} est l'approximation de Y utilisant n_0 modes POD. Sans considérer les contraintes de placement, le gain de l'approche SENSORSPACE provient du fait que Y_{n_0} est utilisée au lieu de Y_{n_S}.

La décomposition d'ordre n_0 de Y s'écrit $U_{n_0}\Sigma_{n_0}V_{n_0}^T$ où :
— U_{n_0} est la restriction de U aux n_0 premières colonnes
— Σ_{n_0} est la restriction de Σ aux n_0 premières lignes et colonnes
— V_{n_0} est la restriction de V n_0 premières colonnes
Le critère est approximé selon :

$$||Y - Y\,(CY)^+\,CY||_F \approx ||U_{n_0}\Sigma_{n_0}V_{n_0}^T - U_{n_0}\Sigma_{n_0}V_{n_0}^T\,(CY)^+\,CU_{n_0}\Sigma_{n_0}V_{n_0}^T||_F \tag{5.20}$$

n_0 doit permettre une très *bonne* approximation de ce critère pour rendre l'approche SENSORSPACE intéressante. Ensuite, la propriété d'invariance de la norme de Frobénius d'une matrice face à une multiplication par une matrice orthonormale est utilisée. En factorisant à droite par V_{n_0} et à gauche par U_{n_0}, le critère devient :

$$\begin{aligned} ||Y - Y\,(CY)^+\,CY||_F &\approx ||U_{n_0}\Sigma_{n_0}V_{n_0}^T - U_{n_0}\Sigma_{n_0}V_{n_0}^T\,(CY)^+\,CU_{n_0}\Sigma_{n_0}V_{n_0}^T||_F \\ &= ||U_{n_0}\left(\Sigma_{n_0} - \Sigma_{n_0}V_{n_0}^T\,(CY)^+\,CU_{n_0}\Sigma_{n_0}\right)V_{n_0}^T||_F \\ &= ||\Sigma_{n_0} - \Sigma_{n_0}V_{n_0}^T\,(CY)^+\,CU_{n_0}\Sigma_{n_0}||_F \end{aligned} \tag{5.21}$$

Cette première simplification permet de réduire les dimensions des matrices mises en jeu de manière considérable. Les matrices Σ_{n_0} et $\Sigma_{n_0} V_{n_0}^T (CY)^+ CU_{n_0}\Sigma_{n_0}$ sont de dimensions plus faibles que Y et $Y(CY)^+ CY$ respectivement. Les dimensions n_{snap} et $n_{\boldsymbol{x}}$ n'interviennent plus.

Les modifications qui suivent ont pour but de réduire la redondance des évaluations successives du critère. En effet, il est intéressant de pré-calculer certains produits matriciels qui interviennent lors de l'évaluation de ϵ. Au détriment d'une évaluation *lourde* au début de l'algorithme, les nombreuses évaluations successives de ϵ (provenant de la modification permanente de pos_S) sont *optimisées* d'un point de vue temps de calcul. Ceci rendra l'algorithme SensorSpace bien plus rapide.

Tout d'abord, la matrice $(CY)^+$ peut être développé en remarquant que CY est de rang plein selon la dimension des lignes. Le cadre considéré impose que $n_S \ll n_{\text{snap}}$, le rang de CY est ainsi obligatoirement inférieur ou égal à n_S. Dans le cas rare où le rang en question est strictement inférieur à n_S, le coupable principal est le fait que deux capteurs se trouvent superposés. Il suffit alors de légèrement déplacer un des capteurs en question.

La pseudo-inverse de CY peut maintenant s'écrire :

$$(CY)^+ = (CY)^T \left((CY)(CY)^T\right)^{-1} \tag{5.22}$$

En développant les produits matriciels, la forme suivante est obtenue :

$$(CY)^+ = Y^T C^T \left(CYY^T C^T\right)^{-1} \tag{5.23}$$

La forme développée de la pseudo-inverse de CY fait ainsi intervenir le terme YY^T. Vu le faible nombre de lignes de C, il est intéressant de pré-calculer le terme YY^T. En effet, le terme CYY^TC^T est une matrice carrée de taille n_S issue de la **restriction** de YY^T aux lignes et colonnes indexées par C. La forme particulière de C permet de remplacer deux multiplications matricielles par des restrictions (élimination de termes). Pour chaque position testée des capteurs, au lieu de calculer la pseudo-inverse de CY, l'inverse de la restriction de YY^T est calculée. Le produit YY^T n'est calculé qu'une seule fois. Ceci permet d'alléger considérablement l'algorithme SensorSpace.

Le critère prend maintenant la forme suivante :

$$\begin{aligned} ||Y - Y(CY)^+ CY||_F &\approx ||\Sigma_{n_0} - \Sigma_{n_0} V_{n_0}^T \left(U_{n_0}\Sigma_{n_0} V_{n_0}^T\right)^T C^T \left(CYY^T C^T\right)^{-1} CU_{n_0}\Sigma_{n_0}||_F \\ &\approx ||\Sigma_{n_0} - \Sigma_{n_0}\Sigma_{n_0}^T U_{n_0}^T C^T \left(CYY^T C^T\right)^{-1} CU_{n_0}\Sigma_{n_0}||_F \end{aligned} \tag{5.24}$$

puisque $V_{n_0}^T V_{n_0} = I_{n_0}$, où I_{n_0} est la matrice identité d'ordre n_0.

Les matrices suivantes sont maintenant introduites :
- $A_1 = \Sigma_{n_0}$
- $A_2 = \Sigma_{n_0}\Sigma_{n_0}^T U_{n_0}^T$
- $A_3 = YY^T$
- $A_4 = U_{n_0}\Sigma_{n_0}$

Ainsi, l'algorithme SensorSpace débute par le calcul de la SVD réduite de Y et la réduction de cette dernière aux n_0 premiers modes POD. Les matrices U_{n_0}, Σ_{n_0} et V_{n_0} sont ainsi connues. Ensuite, les matrices A_1, A_2, A_3 et A_4 sont calculées et stockées en mémoire.

Pour une position donnée des capteurs, représentée par pos_S, les matrices $A_{2,r}$, $A_{3,r}$ et $A_{4,r}$ sont déterminées selon :

— $A_{2,r} = A_2 C^T$ la restriction de A_2 aux colonnes indexées par pos_S
— $A_{3,r} = C A_3 C^T$ la restriction de A_3 aux lignes et aux colonnes indexées par pos_S
— $A_{4,r} = C A_4$ la restriction de A_4 aux lignes indexées par pos_S

Le critère approché est enfin donné par :

$$\boxed{||A_1 - A_{2,r} \left(A_{3,r}\right)^{-1} A_{4,r}||_F} \tag{5.25}$$

Le pseudo-code de l'algorithme se situe dans l'algorithme 11

Algorithm 11 SensorSpace Pseudo-Code

Require: Y, n_S, pos_P, n_0
1: **Initialisation** :
2: $U\Sigma V^T \leftarrow$ décomposition SVD of Y
3: $A_1 \leftarrow \Sigma_{n_0}\Sigma_{n_0}$, $A_2 \leftarrow V_{n_0}\Sigma_{n_0}^T U_{n_0}^T$, $A_3 \leftarrow YY^T$, $A_4 \leftarrow U_{n_0}\Sigma_{n_0}$
4: $\text{pos}_\text{P} \leftarrow$ positions possibles initiales
5: $\text{pos}_\text{S} \leftarrow n_S$ valeurs aléatoires de pos_P satisfaisant aux contraintes
6: $\varepsilon \leftarrow ||A_1 - A_{2,r} \left(A_{3,r}\right)^{-1} A_{4,r}||_F$
7: Répéter jusqu'à ce que le critère d'arrêt soit vérifié
8: **Boucle Principale** :
9: Retirer aléatoirement un capteur de pos_S
10: Mise à jour des éléments de pos_P pour tenir en compte des contraintes (contient n_p éléments) :
11: **for** $j \in [1 \ldots n_p]$ **do**
12: $\quad C \leftarrow \{\text{pos}_\text{S}, \text{pos}_\text{P}(j)\}$
13: $\quad \varepsilon_j \leftarrow ||A_1 - A_{2,r} \left(A_{3,r}\right)^{-1} A_{4,r}||_F$
14: Retenir la position pos_S associée à la plus petite valeur de ε_j
15: **Sortie** : $C \leftarrow \text{pos}_\text{S}$

5.2.2 Résultats

L'approche SensorSpace dans le cas non bruité va maintenant être appliquée sur la configuration test, l'écoulement autour d'un cylindre. Le cas bruité sera traité dans la section suivante de ce chapitre. L'algorithme doit être modifié pour le rendre robuste au bruit de mesure.

Cadre

Deux configurations sont retenues : le cas **contraint** et le cas **non-contraint**.

Le cas **non-contraint** ne prend en compte que les contraintes de géométrie des capteurs. Ainsi, les capteurs peuvent être placés sur la totalité du périmètre du cylindre à condition de ne pas être trop proches les uns des autres. Ici, chaque capteur placé occupe 3 points de maillage supplémentaire sur sa droite et sa gauche.

Le cas **contraint** est une restriction du cas non-contraint. Ainsi, en plus de cette contrainte de géométrie des capteurs, ces derniers ne peuvent pas être placés dans le voisinage des actionneurs. 5 points de maillage à droite et à gauche de chaque actionneur sont ainsi interdits pour le placement des capteurs. Ceci est pris en compte par une simple modification du pos_P initial. Ces zones interdites sont ici très informatives et des performances détériorées par rapport au cas précédent sont attendues.

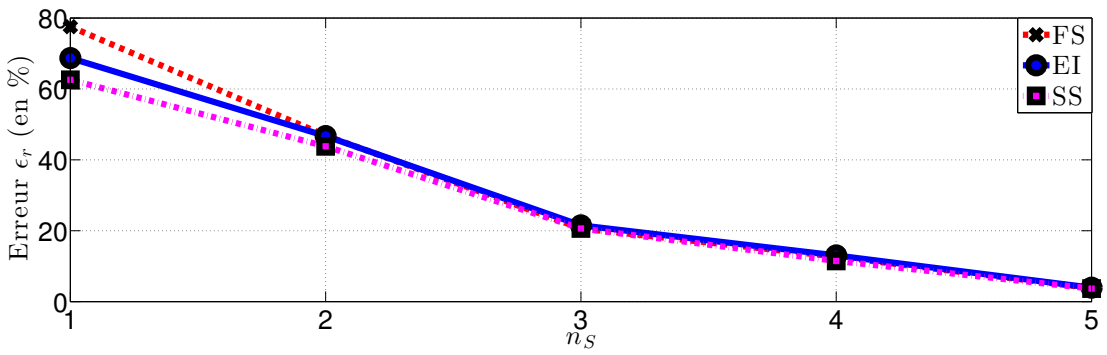

FIGURE 5.2 – SENSORSPACE - Performances - Cas non-contraint

SENSORSPACE est comparé à deux approches : Effective Independence (EI, [17],) et FrameSense (FS,[9]). EI et FS peuvent facilement prendre en compte l'interdiction de placer des capteurs dans le voisinage des actionneurs puisque cela revient à modifier l'ensemble pos_P initial. Par contre, il se peut que les capteurs placés soient trop proches les uns des autres. Si cela est le cas, un des capteurs compatibles est retenu pour la configuration finale. L'algorithme est ensuite relancé avec une modification sur pos_P pour retirer le voisinage du capteur retenu. L'approche *Worst One Out* est ensuite appliquée jusqu'à ce qu'il ne reste que $n_S - 1$ capteurs. Si cela ne suffit pas, la même démarche est répétée jusqu'à ce que les capteurs vérifient les contraintes.

Pour la totalité des résultats obtenus, les algorithmes ont été lancés 50 fois afin d'augmenter les chances d'obtenir un minimum local convenable.

Performances

La première expérience consiste à comparer EI, FS et SENSORSPACE dans le cas non-contraint. L'erreur d'estimation relative ϵ_r est affichée. Jusqu'à 5 capteurs sont ici placés. Pour chaque nombre retenu de capteurs, les algorithmes produisent un placement et une erreur d'estimation ϵ. L'estimateur type POD est utilisé pour EI et FS contrairement à SENSORSPACE.

Nombre de capteurs n_S	Effective Independence	FrameSense	SENSORSPACE
1	66.7%	77.5%	62.6%
2	46.7%	46.8%	43.8%
3	21.5%	20.6%	20.6%
4	13.0%	12.7%	11.5%
5	3.92%	3.80%	3.68%

TABLE 5.1 – SENSORSPACE - Performances - Cas non contraint

Comme visible sur la figure 5.2 et le tableau 5.1, les trois approches présentent de très bons résultats. L'erreur d'estimation décroît lorsque des capteurs sont ajoutés, ce qui est caractéristique du couplage entre l'estimateur utilisé et le positionnement des capteurs. Ceci n'était pas le cas avec l'approche K-SVD classique. C'est d'ailleurs cette limitation qui avait poussé la création de l'algorithme SOBAL.

L'approche SENSORSPACE présente de meilleurs résultats dans le cas $n_S = 1$ et $n_S = 2$. Ce gain provient du fait qu'une base POD d'ordre n_S n'est pas utilisée. Cependant, comme

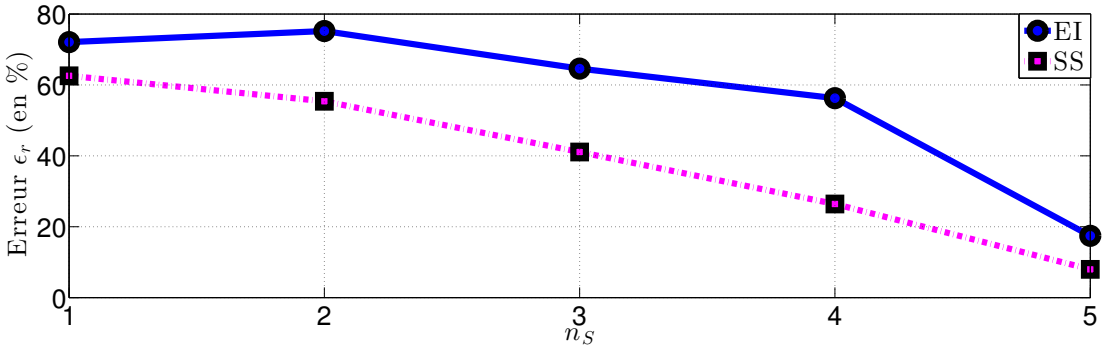

FIGURE 5.3 – SENSORSPACE - Performances - Cas contraint

prévu, ce gain s'estompe lorsque n_S augmente. L'écart entre les 3 approches est de plus en plus faible quand $n_S > 5$ et, avec l'échelle utilisée pour le graphe précédent, les 3 courbes se retrouvent superposées. Dans le cas non contraint, l'approche SENSORSPACE n'est intéressant que dans le cas *peu de capteurs*.

La deuxième expérience considère le cas contraint. L'approche FS s'est ici avérée incapable de produire des positions acceptables. Le retrait des *meilleures* positions (dans le voisinage des actionneurs) a empêché FS de fonctionner convenablement et il faudrait modifier légèrement ce dernier. Le choix a donc été fait de ne comparer que EI et SENSORSPACE. Les résultats sont disponibles sur la figure 5.3 et dans le tableau 5.2.

Nombre de capteurs n_S	Effective Independence	SENSORSPACE
1	72.1%	62.6%
2	75.2%	55.4%
3	64.6%	41.1%
4	56.3%	26.4%
5	17.4%	7.96%

TABLE 5.2 – SENSORSPACE - Performances - Cas contraint

L'approche SENSORSPACE est clairement plus performante que l'approche EI. Dans ce cadre contraint, SENSORSPACE est une approche valide même lorsque plus de capteurs peuvent être utilisés. SENSORSPACE est capable de s'adapter au fait que les positions POD *optimales* ne sont plus disponibles.

5.3 Extensions

5.3.1 Robustesse au bruit de mesure

Cette sous-section a pour objectif de présenter une approche permettant à l'algorithme SENSORSPACE de naturellement prendre en compte le bruit de mesure au niveau des capteurs. L'approche considérée s'appuie sur celle utilisée pour rendre les algorithmes SOBAL généralisé, GOBAL et CB.GOBAL robustes au bruit de mesure (voir la partie 3.4).

Il suffit une fois de plus d'ajouter de la **redondance** au niveau de la séquence d'entraînement et d'ajouter artificiellement du bruit ayant des propriétés statistiques proches du bruit attendu en pratique.

Le bruit de mesure affectant le résultat produit par un capteur est supposé être réel et scalaire. Ces bruits sont indépendants dans leur ensemble et le signal vectoriel aléatoire formé par ces bruits est blanc (sur la durée de prise de *snapshot*).

Tout comme dans la partie 3.4, deux types de bruits sont considérés :
— le bruit de mesure, Ξ_b (défini plus bas), utilisé pour obtenir les performances d'estimation
— le bruit d'entraînement, Ξ_n (défini plus bas), utilisé pour créer S

Soient Ξ_n et Ξ_b les matrices de variables aléatoires permettant de générer ces bruits. Ξ_b est une matrice de taille $n_S \times n_{\text{snap}}$ dont chaque élément est une variable aléatoire suivant une loi gaussienne centrée et d'écart type $\sigma_b \in \mathbb{R}_+^*$ $(\mathcal{N}(0, \sigma_b^2))$. Une réalisation de Ξ_b est notée ξ_b.

Une fois de plus, la redondance est choisie égale à $n_r = 3$. De ce fait, la matrice Ξ_n est de taille $n_S \times n_r n_{\text{snap}}$ dont chaque élément est une variable aléatoire de loi $\mathcal{N}(0, \sigma_n^2)$ avec $\sigma_n \in \mathbb{R}_+^*$. Une réalisation de Ξ_n est notée ξ_n.

La matrice à estimer prend la forme $Y_{\text{ext}} = [Y, Y, Y]$. Cette dernière doit être obtenue à partir des mesures $S_{\text{ext}} = CY_{\text{ext}} + \xi_n$, où ξ_n est une réalisation de Ξ_n. Ainsi, pour chaque *snapshot* $\boldsymbol{y}$, $n_r = 3$ mesures bruitées y sont associées. Augmenter la redondance permet de réduire l'effet de la réalisation de Ξ_n au détriment du temps de calcul.

Le critère à minimiser s'écrit :

$$||Y_{\text{ext}} - R_n S_{\text{ext}}||_F = ||Y_{\text{ext}} - R_n (CY_{\text{ext}} + \xi_n)||_F \tag{5.26}$$

où l'estimateur R_n est encore défini comme $R_n = Y_{\text{ext}} (CY_{\text{ext}} + \xi_n)^+$. Cet estimateur est qualifié de robuste. Il convient de remarquer que la matrice S_{ext} est de dimension $n_S \times n_r n_{\text{snap}}$. Ce faisant, l'extension n'augmente que raisonnablement le temps de calcul associé à la pseudo-inverse.

En effet, d'après [19], la complexité (nombre d'opérations élémentaires nécessaires) du calcul de la pseudo inverse de la matrice $(CY_{\text{ext}} + \xi_n)$ est de

$$\frac{3}{2} {n_S}^2 (n_r n_{\text{snap}}) + \frac{1}{3} {n_S}^3 \tag{5.27}$$

à l'ordre le plus élevé (ici 3). La complexité dépend **linéairement de** n_r ce qui rend exploitable cette approche d'extension.

Cette procédure d'extension de la matrice ne modifie pas son nombre de lignes et le fait qu'elle est de rang plein selon la dimension des lignes. Sans l'extension, cette complexité était de $\frac{3}{2} {n_S}^2 n_{\text{snap}} + \frac{1}{3} {n_S}^3$.

Avant de détailler les critères possibles pour cette approche, il est possible de simplifier le calcul de R_n. Au lieu de définir une matrice de variable aléatoire Ξ_n de taille $n_S \times n_r n_{\text{snap}}$, une matrice $\widetilde{\Xi}_n$ de taille $n_{\boldsymbol{x}} \times n_r n_{\text{snap}}$ est créée. Tout comme Ξ_n, les entrées de $\widetilde{\Xi}_n$ sont toutes des variables aléatoires suivant une loi de probabilité identique. Dans ce cas, avec $\widetilde{\xi}_n$ une réalisation de $\widetilde{\Xi}_n$, l'estimateur R_n peut s'écrire :

$$R_n = Y_{\text{ext}} \left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right) \right)^+ \tag{5.28}$$

Une fois de plus, $C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right)$ est presque toujours de rang plein selon la dimension des lignes. R_n prend donc la forme :

$$\begin{aligned} R_n &= Y_{\text{ext}} \left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right) \right)^T \left(\left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right) \right) \left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right) \right)^T \right)^{-1} \\ &= Y_{\text{ext}} \left(Y_{\text{ext}} + \widetilde{\xi}_n \right)^T C^T \left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n \right) \left(Y_{\text{ext}} + \widetilde{\xi}_n \right)^T C^T \right)^{-1} \end{aligned} \tag{5.29}$$

Algorithme

Deux types de critères pour SensorSpace peuvent maintenant être définis.

Le premier considère les matrices Y_{ext} et S_{ext} bruitées :

$$\epsilon = ||Y_{\text{ext}} - R_n \left(CY_{\text{ext}} + \xi_n\right)||_F, \tag{5.30}$$

où $R_n = Y_{\text{ext}} \left(CY_{\text{ext}} + \xi_n\right)^+$. L'approche utilisée pour minimiser ce critère est développée dans la sous-section suivante 5.3.2.

Le second considère que l'obtention de l'estimateur robuste est suffisante et que l'approche SensorSpace classique non bruitée peut être utilisée. Cela revient à minimiser le critère suivant :

$$\epsilon = ||Y - R_n CY||_F, \tag{5.31}$$

où $R_n = Y_{\text{ext}} \left(CY_{\text{ext}} + \xi_n\right)^+$. En d'autres termes, les positions des capteurs sont choisies afin d'être optimaux dans le cas non-bruité. Ceci est le choix fait pour la suite de ce chapitre.

La simplification du critère s'effectue à l'aide du décomposition SVD de Y, comme l'approche SensorSpace classique. Avec $U_{n_0}\Sigma_{n_0}V_{n^{\text{o}}}^T$ la décomposition SVD d'ordre $n_0 \in \mathbb{N}^*$ de Y, le critère approchée devient :

$$\epsilon \approx ||U_{n_0}\Sigma_{n_0} - R_n CU_{n_0}\Sigma_{n_0}||_F \tag{5.32}$$

Vu que $R_n = Y_{\text{ext}} \left(Y_{\text{ext}} + \widetilde{\xi}_n\right)^T C^T \left(C \left(Y_{\text{ext}} + \widetilde{\xi}_n\right) \left(Y_{\text{ext}} + \widetilde{\xi}_n\right)^T C^T\right)^{-1}$, en posant :

- $A_1 = U_{n_0}\Sigma_{n_0}$
- $A_2 = Y_{\text{ext}} \left(Y_{\text{ext}} + \widetilde{\xi}_n\right)^T$ et $A_{2,r} = A_2 C^T$
- $A_3 = \left(Y_{\text{ext}} + \widetilde{\xi}_n\right) \left(Y_{\text{ext}} + \widetilde{\xi}_n\right)^T$ et $A_{3,r} = CA_3C^T$
- $A_4 = U_{n_0}\Sigma_{n_0}$ et $A_{4,r} = CA_4$

Enfin, pour une réalisation $\widetilde{\xi}_n$, l'approche SensorSpace robuste au bruit peut être utilisée en minimisant le critère :

$$\boxed{\epsilon \approx ||A_1 - A_{2,r} \left(A_{3,r}\right)^{-1} A_{4,r}||_F} \tag{5.33}$$

Le pseudo-code de l'algorithme SensorSpace rendu robuste au bruit $\widetilde{\xi}_n$ se situe dans l'algorithme 12

Résultats

Cette approche va maintenant être illustrée avec des données numériques.

La cas contraint est considéré. Pour rappel, cela veut dire que les capteurs ne peuvent pas être placés trop proches les uns des autres (3 points de maillage de chaque côté) et que les capteurs ne peuvent pas être placés dans le voisinage des actionneurs (10 points de maillage de part et d'autre).

Ensuite, étant donné que l'objectif est d'illustrer les capacités de robustesse au bruit de SensorSpace, seulement le cas $n_S = 4$ est considéré.

Le bruit d'entraînement, qualifié avec l'indice n, suit une loi gaussienne centrée et de variance $\sigma_n \in \mathbb{R}_+^*$. Plusieurs valeurs de σ_n vont être utilisées afin de générer différents estimateurs et positions associées de capteurs. Pour chaque valeur de σ_n, 10 réalisations de $\widetilde{\xi}_n$ sont générées. SensorSpace bruité est ensuite appliqué (5 fois) sur chaque réalisation et

Algorithm 12 SensorSpace (Noise Robust) Pseudo-Code

Require: Y, $\widetilde{\xi}_n$, n_S, pos_P, n_0, n_r
1: **Initialisation** :
2: $U\Sigma V^T \leftarrow$ décomposition SVD of Y
3: $Y_\text{ext} \leftarrow [Y, \ldots, Y]$ (n_r fois)
4: $A_1 \leftarrow U_{n_0}\Sigma_{n_0}$, $A_2 \leftarrow Y_\text{ext}\left(Y_\text{ext} + \widetilde{\xi}_n\right)^T$ et $A_{2,r} = A_2 C^T$
5: $A_3 \leftarrow \left(Y_\text{ext} + \widetilde{\xi}_n\right)\left(Y_\text{ext} + \widetilde{\xi}_n\right)^T$, $A_{3,r} \leftarrow CA_3C^T$, $A_4 \leftarrow U_{n_0}\Sigma_{n_0}$, $A_{4,r} =\leftarrow CA_4$
6: $\text{pos}_\text{P} \leftarrow$ positions possibles initiales
7: $\text{pos}_\text{S} \leftarrow n_S$ valeurs aléatoires de pos_P satisfaisant aux contraintes
8: $\varepsilon \leftarrow ||A_1 - A_{2,r}\left(A_{3,r}\right)^{-1} A_{4,r}||_F$
9: Répéter jusqu'à ce que le critère d'arrêt soit vérifié
10: **Boucle Principale** :
11: Retirer aléatoirement un capteur de pos_S
12: Mise à jour des éléments de pos_P pour tenir en compte des contraintes (contient n_p éléments) :
13: **for** $j \in [1 \ldots n_p]$ **do**
14: $\quad C \leftarrow \{\text{pos}_\text{S}, \text{pos}_\text{P}(j)\}$
15: $\quad A_{2,r} = A_2C^T$, $A_{3,r} \leftarrow CA_3C^T$, $A_{4,r} =\leftarrow CA_4$
16: $\quad \varepsilon_j \leftarrow ||A_1 - A_{2,r}\left(A_{3,r}\right)^{-1} A_{4,r}||_F$
17: Retenir la position pos_S associée à la plus petite valeur de ε_j
18: **Sortie** : $C \leftarrow \text{pos}_\text{S}$

le meilleur estimateur (minimisant le critère ϵ) ainsi que les positions associées sont retenus.

Ensuite, les performances de ces paires estimateurs/positions sont testées sur des mesures bruitées par un bruit suivant une loi gaussienne centrée et de variance $\sigma_b \in \mathbb{R}_+^*$. Une fois de plus, plusieurs valeurs de σ_b vont être considérées afin de tester la paire estimateur/positions dans des conditions plus ou moins bruitées. Pour chaque valeur de σ_b, 100 réalisations de ξ_b sont générées. Chaque réalisation permet de déterminer une valeur de l'erreur d'estimation relative globale. Enfin, en effectuant la moyenne de ces dernières valeurs, une estimée de l'espérance de l'erreur d'estimation relative sur la séquence totale est obtenue.

Ces résultats se trouvent sur la figure 5.4 où cette espérance est tracée pour différentes valeurs de σ_b. Le tableau 5.3 contient les données numériques associées. Chaque courbe correspond à une valeur différente de σ_n, et ainsi, à une configuration spécifique des capteurs. Les performances d'Effective Independence dans le même contexte sont aussi représentées.

σ_n^2	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.01$	$\sigma_b^2 = 0.02$	$\sigma_b^2 = 0.05$	$\sigma_b^2 = 0.1$	$\sigma_b^2 = 0.2$	$\sigma_b^2 = 0.5$	$\sigma_b^2 = 1$
EI	56.3%	56.5%	57.2%	61.8%	76.3%	117%	262%	517%
SS (0.01)	26.4%	27.8%	31.5%	50.4%	89.8%	174%	429%	858%
SS (0.05)	28.9%	29.2%	30.3%	37.2%	55.0%	98.0%	235%	469%
SS (0.5)	59.8%	59.9%	60.0%	60.0%	60.2%	61.3%	68.5%	89.7%
SS (1)	66.7%	66.7%	66.7%	66.8%	66.8%	67.1%	68.6%	74.0%

Table 5.3 – SensorSpace - Performances Cas non-contraint, bruité

Pour un bruit d'entraînement de variance faible, $\sigma_n^2 = 0.01$, SensorSpace présente de

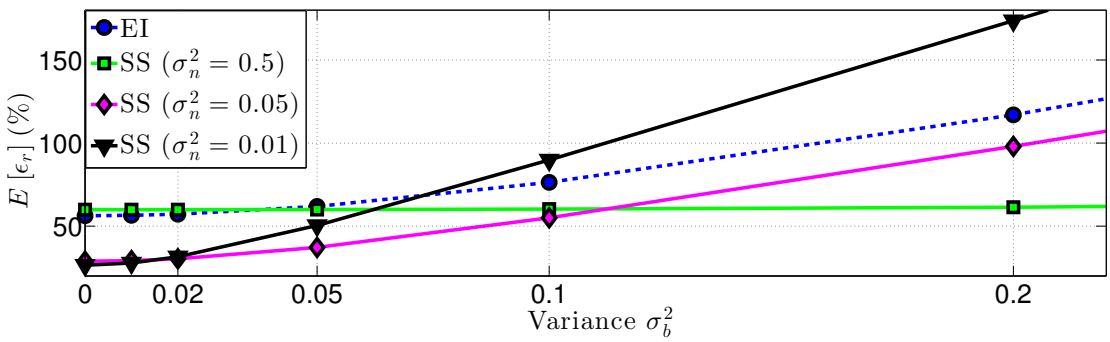

FIGURE 5.4 – SENSORSPACE - Performances $n_S = 5$ - Cas contraint bruité

très bonnes performances lorsqu'il y a peu de bruit de mesure. SENSORSPACE est ici plus performant que EI. Cependant, lorsque σ_b augmente, les performances de SENSORSPACE sont de plus en plus mauvaises et, pour $\sigma_b^2 = 0.1$, elles se retrouvent inférieures à EI.

Heureusement, en utilisant $\sigma_n^2 = 0.05$, SENSORSPACE présente un comportement bruité comparable à Effective Independence (pour les valeurs de σ_b testés). Pour chaque valeur de σ_b, SENSORSPACE présente une espérance de l'erreur d'estimation plus faible qu'EI. Il est aussi remarquable que la performance non-bruitée de SENSORSPACE réglée avec $\sigma_n^2 = 0.05$ est légèrement inférieure (espérance de l'erreur d'estimation plus grande) que lorsque SENSORSPACE était réglé avec $\sigma_n^2 = 0.01$. Ceci est encore plus visible dans le cas $\sigma_n^2 = 0.5$. Finalement, un **compromis** est réalisé entre la robustesse au bruit et les performances dans le cas non-bruité. Ceci était le comportement désiré de l'approche retenue.

SENSORSPACE est capable de produire des **résultats robustes au bruit de mesure** à condition d'avoir été **entraîné/réglé** sur un bruit de propriétés statistiques adéquat.

5.3.2 Cas Goal-Oriented

Afin de pouvoir traiter les problèmes de type *Pb.GoalOriented* à l'aide des algorithmes GOBAL et CB.GOBAL, il est nécessaire d'utiliser des positions adaptées de capteurs. Les mesures sont effectuées sur le champ Y mais, cette fois-ci, ce n'est plus Y qui doit être estimée à partir de ces mesures mais H. Ainsi, il est très probable que des placements optimisés de capteurs pour estimer Y vont différer de ceux optimisés pour estimer H.

Avant de considérer le cas général, il convient de noter le cas particulier où H est reliée de manière linéaire à Y. Dans ce cas, il existe une matrice $G \in \mathbb{R}^{n_h \times n_x}$ telle que :

$$H = GY \tag{5.34}$$

La norme à minimiser, selon la variable C, s'écrit :

$$||H - RCY||_F, \tag{5.35}$$

où R est donné par $H\,(CY)^+$.

Dans le cas particulier considéré, cette norme se ré-écrit :

$$\begin{aligned} ||H - RCY||_F &= ||GY - GY\,(CY)^+\,CY||_F \\ &= ||G\left(Y - Y\,(CY)^+\,CY\right)||_F \end{aligned} \tag{5.36}$$

Le nouveau critère à minimiser s'exprime en fonction du critère vu dans la section précédente. Si G est orthonormale, le critère dans le cas Goal-Oriented est le même que dans

le cas classique et, de ce fait, les positions des capteurs restent inchangées. Cependant, dans le cas général où G n'est pas orthonormale, le critère est modifié et de même pour les résultats qui en découlent.

Ce qu'il faut retenir de ce cas particulier est que, même dans le cas où une relation linéaire existe entre H et Y, il est intéressant de minimiser le critère 5.36 pour obtenir de nouvelles positions pour les capteurs. Utiliser SensorSpace avec le critère classique pour estimer Y puis multiplier cette matrice *estimée* par G sera une approche moins performante qu'obtenir de nouvelles positions à partir du critère Goal-Oriented.

Ceci peut être ressenti avec l'exemple de l'estimation de la traînée de pression. Cette traînée est obtenue en intégrant la pression disponible à la surface du cylindre pondérée par une fonction cosinus. De ce fait, appliquer SensorSpace dans le cas Goal-Oriented permettra une meilleure estimation du champ de pression à la surface du cylindre, et en particulier, des points qui sont proches de l'axe principal de l'écoulement (où la fonction cosinus est max, en valeur absolue). Bien évidemment, ceci ce fait au détriment d'une bonne estimation du champ de pression éloigné de la surface du cylindre.

Pour minimiser ce critère, les mêmes simplifications peuvent être effectuées directement sur le critère sauf la factorisation à gauche par une matrice orthonormale. Une décomposition SVD de G permet d'effectuer cette factorisation.

Le cas général est maintenant considéré. Aucune relation *a priori* n'est connue entre H et Y. Le critère 5.35 s'écrit avant toute simplification : $||H - H\,(CY)^{+}\,CY||_F$.

Cette fois-ci, deux décompositions SVD sont nécessaires.

La décomposition SVD d'ordre $n_0 \in \mathbb{N}^*$ de Y produit :

— $U_{Y_{n_0}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_0}$, orthogonale
— $\Sigma_{Y_{n_0}} \in \mathbb{R}^{n_0 \times n_0}$, diagonale
— $V_{Y_{n_0}} \in \mathbb{R}^{n_0 \times n_{\text{snap}}}$, orthogonale

tel que $||Y - U_{Y_{n_0}} \Sigma_{Y_{n_0}} V_{Y_{n_0}}^T||_F$ soit minimale.

La décomposition SVD d'ordre $n_L \in \mathbb{N}^*$ de H produit :

— $U_{H_{n_L}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_L}$, orthogonale
— $\Sigma_{H_{n_L}} \in \mathbb{R}^{n_L \times n_L}$, diagonale
— $V_{H_{n_L}} \in \mathbb{R}^{n_L \times n_{\text{snap}}}$, orthogonale

tel que $||H - U_{H_{n_L}} \Sigma_{H_{n_L}} V_{H_{n_L}}^T||_F$ soit minimale.

Le choix de n_0 et n_L se fait pour que les erreurs relatives $||Y - U_{Y_{n_0}} \Sigma_{Y_{n_0}} V_{Y_{n_0}}^T||_F / ||Y||_F$ et $||H - U_{H_{n_L}} \Sigma_{H_{n_L}} V_{H_{n_L}}^T||_F / ||H||_F$ soient suffisamment faibles selon les besoins de l'utilisateur.

En d'autres termes, en définissant la matrice erreur générale $A_g = H - H\,(CY)^{+}\,CY$ et la matrice erreur approximée : $A_a = U_{H_{n_L}} \Sigma_{H_{n_L}} V_{H_{n_L}}^T - U_{H_{n_L}} \Sigma_{H_{n_L}} V_{H_{n_L}}^T\,(CY)^{+}\,C U_{Y_{n_0}} \Sigma_{Y_{n_0}} V_{Y_{n_0}}^T$, le choix de n_0 et n_L doit être tel que l'erreur relative $||A_g - A_a||_F / ||A_g||_F$ soit de quelques ordres de grandeur plus faible que l'erreur d'estimation désirée $||H - \widehat{H}||_F / ||H||_F$.

Une fois ces décompositions SVD effectuées, les mêmes approches de simplification d'évaluation du critère que celles utilisées pour SensorSpace *classique* sont effectuées. Une seule factorisation par une matrice orthogonale peut être effectuée cette fois-ci, rendant le critère plus complexe à évaluer (d'un point de vue temps de calcul) que le critère

classique. En factorisant à gauche par $U_{H_{n_L}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_L}$, le critère approché devient :

$$||H - H\,(CY)^+\,CY||_F \approx ||\Sigma_{H_{n_L}} V_{H_{n_L}}^T - \Sigma_{H_{n_L}} V_{H_{n_L}}^T\,(CY)^+\,CU_{Y_{n_0}}\Sigma_{Y_{n_0}} V_{Y_{n_0}}^T||_F \tag{5.37}$$

En développant une fois de plus la pseudo-inverse $(CY)^+$ dans le cas où CY est de rang plein selon la dimension des lignes (n_S) et en introduisant les matrices :

— $A_1 = \Sigma_{H_{n_L}} V_{H_{n_L}}^T$
— $A_2 = \Sigma_{H_{n_L}} V_{H_{n_L}}^T V_{Y_{n_0}} \Sigma_{Y_{n_0}} U_{Y_{n_0}}^T$ et $A_{2,r} = A_2 C^T$
— $A_3 = YY^T$ et $A_{3,r} = CA_3C^T$
— $A_4 = U_{Y_{n_0}} \Sigma_{Y_{n_0}} V_{Y_{n_0}}^T$ et $A_{4,r} = CA_4$

le critère approché devient :

$$\epsilon \approx ||A_1 - A_{2,r}\,(A_{3,r})^{-1}\,A_{4,r}||_F \tag{5.38}$$

Le pseudo-code de l'algorithme SensorSpace généralisé pour le cas Goal-Oriented est donné dans l'algorithme 13

Algorithm 13 SensorSpace (Goal-Oriented) Pseudo-Code

Require: Y, H, n_S, pos_P, n_0, n_L
1: **Initialisation** :
2: $U_Y \Sigma_Y V_Y^T \leftarrow$ décomposition SVD of Y
3: $U_H \Sigma_H V_H^T \leftarrow$ décomposition SVD of H
4: $A_1 \leftarrow \Sigma_{H_{n_L}} V_{H_{n_L}}^T$, $A_2 \leftarrow \Sigma_{H_{n_L}} V_{H_{n_L}}^T V_{Y_{n_0}} \Sigma_{Y_{n_0}} U_{Y_{n_0}}^T$, $A_3 \leftarrow YY^T$, $A_4 \leftarrow U_{Y_{n_0}} \Sigma_{Y_{n_0}} V_{Y_{n_0}}^T$
5: $\text{pos}_\text{P} \leftarrow$ positions possibles initiales
6: $\text{pos}_\text{S} \leftarrow n_S$ valeurs aléatoires de pos_P satisfaisant aux contraintes
7: $\varepsilon \leftarrow ||A_1 - A_{2,r}\,(A_{3,r})^{-1}\,A_{4,r}||_F$
8: Répéter jusqu'à ce que le critère d'arrêt soit vérifié
9: **Boucle Principale** :
10: Retirer aléatoirement un capteur de pos_S
11: Mise à jour des éléments de pos_P pour tenir en compte des contraintes (contient n_p éléments) :
12: **for** $j \in [1 \ldots n_p]$ **do**
13: $\quad C \leftarrow \{\text{pos}_\text{S}, \text{pos}_\text{P}(j)\}$
14: $\quad A_{2,r} = A_2 C^T$, $A_{3,r} \leftarrow CA_3C^T$, $A_{4,r} =\leftarrow CA_4$
15: $\quad \varepsilon_j \leftarrow ||A_1 - A_{2,r}\,(A_{3,r})^{-1}\,A_{4,r}||_F$
16: Retenir la position pos_S associée à la plus petite valeur de ε_j
17: **Sortie** : $C \leftarrow \text{pos}_\text{S}$

Afin de ne pas surcharger ce chapitre, des exemples d'applications de SensorSpace Goal-Oriented seront présentés dans le chapitre suivant. En effet, de nouveaux problèmes de type *Pb.GoalOriented* doivent être présentés pour utiliser un tel algorithme. Enfin, dans la suite, il n'y aura pas de distinction entre les algorithmes SensorSpace et SensorSpace Goal-Oriented afin de ne pas surcharger les notations. L'algorithme adapté sera utilisé suivant le contexte.

5.3.3 Capteurs pour la classification

Avant de conclure, une approche pour trouver un placement de capteurs adapté à l'approche GOC va être proposée. Cette approche est encore dans un stade embryonnaire. Elle n'a pas été optimisée ni été testée en détail.

L'idée principale consiste à **modifier l'estimateur** utilisé par SensorSpace. Au lieu d'utiliser un seul estimateur linéaire, un ensemble d'estimateurs linéaires est considéré. Vu l'implémentation de GOC, l'approche SensorSpace peut être utilisée sans grande modification. Pour chaque nouvelle configuration testée des capteurs, GOC est appliqué pour classer les *snapshots* et calculer les estimateurs.

Le pseudo-code *brut* se trouve dans l'algorithme 14.

Dans cet état, l'algorithme est très lent à converger. Avec la séquence d'entraînement $\mathcal{S}_{\mathrm{l}}$, dans le cas non contraint, il a fallu attendre 4 heures pour obtenir un seul résultat avec $n_S = 3$. SensorSpace classique ne nécessite que $8.8s$ par comparaison ($8.2s$ pour calculer les matrices redondantes et $0.6s$ pour effectuer la recherche des capteurs).

Algorithm 14 SensorSpace (Classification) Pseudo-Code

Require: Y, H, n_S, n_K, $\mathrm{pos_P}$
1: **Initialisation** :
2: $\mathrm{pos_P} \leftarrow$ positions possibles initiales
3: $\mathrm{pos_S} \leftarrow n_S$ valeurs aléatoires de $\mathrm{pos_P}$ satisfaisant aux contraintes
4: Répéter jusqu'à ce que le critère d'arrêt soit vérifié
5: **Boucle Principale** :
6: Retirer aléatoirement un capteur de $\mathrm{pos_S}$
7: Mise à jour des éléments de $\mathrm{pos_P}$ pour tenir en compte des contraintes (contient n_p éléments) :
8: **for** $j \in [1 \dots n_p]$ **do**
9: $\quad C \leftarrow \{\mathrm{pos_S}, \mathrm{pos_P}(j)\}$
10: $\quad \varepsilon_j \leftarrow$ l'erreur d'estimation relative effectuée par GOC (n_K)
11: Retenir la position $\mathrm{pos_S}$ associée à la plus petite valeur de ε_j
12: **Sortie** : $C \leftarrow \mathrm{pos_S}$

Aucun résultat numérique ne sera donné ici mais les tests préliminaires effectués permettent d'obtenir des positions de capteurs qui produisent des erreurs GOC plus faibles que l'approche SensorSpace classique.

5.4 Conclusion

Un nouvel algorithme, SensorSpace, ayant pour fonction d'effectuer du placement de capteurs, a été présenté dans ce chapitre. Ce dernier était nécessaire afin de permettre aux algorithmes de type SOBAL de pleinement exprimer leur potentiel. A partir d'une séquence d'entraînement, d'une indication sur les emplacements possibles et du nombre de capteurs pouvant être placés, l'algorithme produit un ensemble indiquant les positions retenues des capteurs ainsi qu'un estimateur linéaire. Contrairement à l'approche POD (utilisée par Effective Independence), cet estimateur linéaire est soumis à moins de contraintes de structure. Ceci est a l'origine de la bonne interaction entre les positions SensorSpace et les algorithmes de types SOBAL.

L'algorithme SensorSpace a tout d'abord été présenté afin de répondre aux limitations de Effective Independence dans le cas considéré. Cependant, l'utilisation de SensorSpace ne se limite pas qu'à un rôle de support pour l'algorithme SOBAL vu dans le chapitre 3. SensorSpace permet aussi de répondre plus efficacement que l'approche Effective Independence au prix d'un plus grand temps de calcul. Un tel résultat est suffisant pour ajouter SensorSpace à la longue liste des méthodes de placement de capteurs.

Les particularités de SensorSpace sont de pouvoir délivrer de très bons résultats dans le cas peu de capteurs (c'est à dire $n_S \leq n_D$ où n_D est le nombre de modes POD requis pour garantir l'erreur d'estimation désirée avec l'approche EI) ainsi que de pouvoir incorporer efficacement des contraintes sur le placement des capteurs. Ces caractéristiques sont rendues possibles d'une part, en manipulant Y dans le critère à minimiser au lieu de n'utiliser que la base POD associée, d'autre part en s'éloignant d'une approche *Worst One Out* permettant de modifier $\mathrm{pos_P}$ à chaque modification d'un capteur. Afin de pouvoir rivaliser pleinement avec Effective Independence, une approche pour rendre les positions et l'estimateur utilisé robuste au bruit de mesure a été présentée. Cette approche se base sur l'ajout de redondance dans la séquence d'apprentissage et la nature de l'estimateur linéaire obtenu est rendue directement robuste. Des bruits de natures différentes de celles des bruits gaussiens peuvent être considérés. La seule contrainte étant pour l'instant l'utilisation de capteurs identiques, soumis à des bruits de mesures de mêmes paramètres statistiques.

Enfin, SensorSpace a pu être facilement modifié pour être applicable au problème *Pb.GoalOriented*. Ce faisant, des positions de capteurs exploitables par les algorithmes GOBAL et CB.GOBAL peuvent être créées. Cette modification ouvre la porte vers la résolution de problèmes bien plus variés que l'estimation du champ de pression à partir de mesures de pression en surface du cylindre. L'objectif du chapitre suivant est ainsi d'illustrer le potentiel des approches conçues lors de ce travail de thèse.

Bibliographie

[1] K Worden and A.P Burrows. Optimal sensor placement for fault detection. *Engineering Structures*, 23(8) :885–901, August 2001.

[2] Bingni W. Brunton, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Optimal Sensor Placement and Enhanced Sparsity for Classification. page 13, October 2013.

[3] P Gündeş Bakir. Evaluation of Optimal Sensor Placement Techniques for Parameter Identification in Buidlings. *Mathematical and Computational Applications*, 16(2) :456–466, 2011.

[4] M. Meo and G. Zumpano. On the optimal sensor placement techniques for a bridge structure. *Engineering Structures*, 27(10) :1488–1497, August 2005.

[5] Xiao-Hua Zhang, You-Lin Xu, Songye Zhu, and Sheng Zhan. Dual-type sensor placement for multi-scale response reconstruction. *Mechatronics*, 24(4) :376–384, June 2014.

[6] Antonio A. Alonso, Christos E. Frouzakis, and Ioannis G. Kevrekidis. Optimal sensor placement for state reconstruction of distributed process systems. *AIChE Journal*, 50(7) :1438–1452, July 2004.

[7] Wei Liu, Wei-cheng Gao, Yi Sun, and Min-jian Xu. Optimal sensor placement for spatial lattice structure based on genetic algorithms. *Journal of Sound and Vibration*, 317(1-2) :175–189, October 2008.

[8] Rafael Castro-Triguero, Senthil Murugan, Rafael Gallego, and Michael I. Friswell. Robustness of optimal sensor placement under parametric uncertainty. *Mechanical Systems and Signal Processing*, 41(1-2) :268–287, December 2013.

[9] Juri Ranieri, Amina Chebira, Martin Vetterli, and Student Member. Near-Optimal Sensor Placement for Linear Inverse Problems. *IEEE Transactions on Signal Processing*, 62(5) :1135–1146, March 2014.

[10] K. Cohen. Effective Sensor Placements for the Estimation of Proper Orthogonal Decomposition Mode Coefficients in von Karman Vortex Street. *Journal of Vibration and Control*, 10(12) :1857–1880, December 2004.

[11] Tamir Hegazy and George Vachtsevanos. Sensor Placement for Isotropic Source Localization. *Lecture notes in computer science*, 2634 :432–441, 2003.

[12] M. P. Limongelli. Optimal location of sensors for reconstruction of seismic responses through spline function interpolation. *Earthquake Engineering & Structural Dynamics*, 32(7) :1055–1074, June 2003.

[13] Wei Kang and Liang Xu. Optimal placement of mobile sensors for data assimilations. *Tellus A*, 64 :1–12, October 2012.

[14] N. Debnath, A. Dutta, and S. K. Deb. Placement of sensors in operational modal analysis for truss bridges. *Mechanical Systems and Signal Processing*, 31 :196–216, August 2012.

[15] Maik Brehm, Volkmar Zabel, and Christian Bucher. Optimal reference sensor positions using output-only vibration test data. *Mechanical Systems and Signal Processing*, 41(1-2) :196–225, December 2013.

[16] F.Y.S. Lin and P.L. Chiu. A near-optimal sensor placement algorithm to achieve complete coverage-discrimination in sensor networks. *IEEE Communications Letters*, 9(1) :43–45, 2005.

[17] Daniel C. Kammer. Sensor placement for on-orbit modal identification and correlation of large space structures. *Journal of Guidance, Control, and Dynamics*, 14(2) :251–259, March 1991.

[18] Dong-Sheng Li, Hong-Nan Li, and Claus-Peter Fritzen. A note on fast computation of effective independence through QR downdating for sensor placement. *Mechanical Systems and Signal Processing*, 23(4) :1160–1168, May 2009.

[19] H. Michael Möller. Exact computation of the generalized inverse and least-squares solution. Technical report, Universität Dortmund, Germany, 1999.

CHAPITRE 6
Mise en œuvre pratique

Chapitre 6

Mise en œuvre pratique

Contenu

L'objectif de ce chapitre est d'illustrer la flexibilité et les performances des algorithmes proposés. Jusqu'à maintenant, le problème *Pb.Estimation* a constitué le cadre de comparaison des études menées. Cependant, chaque algorithme a été modifié afin de pouvoir agir sur un problème plus large : *Pb.GoalOriented*. Ce chapitre considère ainsi de nouveaux problèmes, plus complexes, afin de pouvoir illustrer expérimentalement le bon fonctionnement des algorithmes.

Ces derniers sont :
— SOBAL (généralisé), permettant de répondre à *Pb.Estimation* et à une restriction de *Pb.GoalOriented*,
— GOBAL, permettant de répondre à *Pb.GoalOriented* en reprenant les avantages de SOBAL,
— GOC, permettant de répondre à *Pb.GoalOriented* via l'utilisation de classes et d'estimateurs linéaires,
— CB.GOBAL, une fusion de GOC et GOBAL permettant de répondre à *Pb.GoalOriented* de manière efficace (réduction de la parcimonie et utilisation de classes),
— SensorSpace permettant de placer des capteurs pour répondre à *Pb.Estimation* et *Pb.GoalOriented*.

Trois thèmes sont ici abordés :
— le cas Dynamique, où les mesures passées peuvent être utilisées,
— le cas extension des mesures, où les commandes appliquées peuvent être utilisées,

— le *découplage pur*, où la nature du champ à estimer est différente du champ sur lequel les mesures sont effectuées.

Sans précisions supplémentaires, les notations non-introduites dans ce chapitre sont détaillées dans les chapitres précédents.

Enfin, étant donné que la totalité des algorithmes utilisés présentent une part d'aléatoire provenant des conditions initiales et de l'ordre de mise à jour des modes, chaque algorithme est utilisé un *grand* nombre de fois (plus de 15) afin d'avoir plus de chances d'obtenir une solution satisfaisante. En pratique, seuls quelques lancers $(4-5)$ sont nécessaires pour obtenir une solution performante.

Enfin, une étude détaillée et comparative des différents algorithmes sur chaque problème proposé ne sera pas effectuée. Les avantages et inconvénients de ces derniers ont déjà été présentés. Ce chapitre permet cependant au lecteur de se faire une idée plus concrète des réglages à effectuer pour exploiter pleinement ces algorithmes selon les problèmes étudiés.

6.1 Exploitation des mesures passées

Cette section a pour objectif de proposer des estimateurs qui peuvent exploiter les mesures passées. De telles informations sont en effet facilement disponibles et elles devraient pouvoir améliorer la qualité des estimations faites. La première partie de ce chapitre va formaliser le problème à l'aide de l'introduction de nouvelles notations. Les approches conçues doivent toujours être applicables en pratique. De ce fait, la limitation de ne pas avoir accès à un modèle d'évolution est encore d'actualité.

Il convient de remarquer que la méthode incontournable du filtrage de Kalman n'est pas utilisée dans le cas présent puisqu'un tel modèle d'évolution n'est pas disponible (par choix). L'objectif est d'exploiter au mieux les algorithmes présentés dans les chapitres précédents.

6.1.1 Extension des mesures

Le cadre de ce problème est maintenant posé. En reprenant une fois de plus les notations vues dans le chapitre 2, le problème considéré, nommé *Pb.Dynamique*, est défini par :

<u>Données</u> : Y, C

<u>Objectif</u> : Trouver une application $\Psi(.,.)$ définie sur $(\mathbb{R}^{n_S} \times \mathbb{R}^{n_S}) \to \mathbb{R}^{n_{\boldsymbol{x}}}$ qui minimise $\epsilon_r = ||Y - \widehat{Y}||_F / ||Y||_F$

où $\widehat{Y} = \left(\Psi(\boldsymbol{s}_{t_2}, \boldsymbol{s}_{t_1}) \ldots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}, \boldsymbol{s}_{t_{n_{\text{snap}}-1}}\right)\right)$ et $S = CY$.

La nouveauté principale par rapport au problème *Pb.Estimation* est le fait que l'estimateur retenu peut maintenant exploiter l'information passée. $\Psi(.)$ est ainsi plus complexe. Le problème ci-dessus est qualifié de $\boldsymbol{s}_i/\boldsymbol{s}_{i-1}$ pour indiquer que les mesures du *snapshot* présent i et du *snapshot immédiatement passé* $i-1$ sont utilisées pour estimer l'état du champ à l'instant présent i. Pour éclaircir cette notation, le problème qualifié par $\boldsymbol{s}_i/\boldsymbol{s}_{i-1}/\boldsymbol{s}_{i-2}$ consiste à estimer l'état présent i à partir de la mesure présente i et des mesures passées $i-1$ et $i-2$.

Soit $k \in \mathbb{N}^*$ l'écart considéré entre les deux instants de mesures. Le choix de k va dépendre entièrement des dynamiques contenues dans la séquence Y et de la période

d'échantillonnage des *snapshots*. Dans un premier temps, l'approche $\boldsymbol{s}_i/\boldsymbol{s}_{i-k}$ est considérée (bien que sous optimale par rapport à une approche exploitant plus de mesures). Les méthodes sont développées pour être flexibles sur l'intégration de plusieurs instants passés.

Le choix effectué pour traiter ce problème est une **extension** de la notion de mesure. Le cas où seule une mesure passée est gardée en mémoire est étudié dans un premier temps. Cette mesure passée est disponible à l'instant *présent* et peut donc être considérée comme un relevé supplémentaire.

Soit $S_{\text{ext}} \in \mathbb{R}^{2n_S \times n_{\text{snap}}}$ la matrice contenant les mesures dites étendues. Cette dernière est définie par :

$$S_{\text{ext}}{}^T = \left(S_i^T S_{i-k}^T\right), \tag{6.1}$$

où

— $S = CY$,
— S_i est la restriction de S aux *snapshots* indexés de $1+k$ à n_{snap},
— S_{i-k} est la restriction de S aux *snapshots* indexés de 1 à $(n_{\text{snap}} - k)$.

Le premier *snapshot* de S_{ext} est alors formé de la concaténation (selon la dimension des colonnes) des vecteurs $\boldsymbol{s}_{1+k}$ et $\boldsymbol{s}_1$. Dans le cas où $k = 1$ et $n_{\text{snap}} = 4$, les matrices précédentes deviennent :

— $S_i = [\boldsymbol{s}_2\ \boldsymbol{s}_3\ \boldsymbol{s}_4]$,
— $S_{i-1} = [\boldsymbol{s}_1\ \boldsymbol{s}_2\ \boldsymbol{s}_3]$,
— $S_{\text{ext}} = \begin{bmatrix} S_i \\ S_{i-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_2 & \boldsymbol{s}_3 & \boldsymbol{s}_4 \\ \boldsymbol{s}_1 & \boldsymbol{s}_2 & \boldsymbol{s}_3 \end{bmatrix}$

La séquence à estimer est $H = Y_i$ définie comme la restriction de Y aux *snapshots* indexés de $1+k$ à n_{snap}.

De par cette formulation, l'estimateur recherché agit sur une seule quantité vectorielle de dimension $2n_S$. Ces $2n_S$ composantes correspondent au nombre de mesures effectives noté $n_{S_{\text{ext}}} = 2n_S \in \mathbb{N}^*$.

Plus k est grand, plus la séquence initiale Y est raccourcie. Il faut ainsi s'assurer d'avoir suffisamment de *snapshots*. Dans le cas contraire, une pondération peut être utilisée afin de comparer de manière plus juste les résultats obtenus avec des valeurs de k différentes.

6.1.2 Placement des capteurs et retard

Placement des capteurs

Le choix du retard k semble être l'étape prioritaire après avoir choisi d'effectuer l'extension de mesure. Cependant, le placement des capteurs va être traité en premier et va ensuite être utilisé pour choisir k.

Il semble en effet intéressant de placer des capteurs qui vont exploiter le fait que 2 *snapshots* du champ $\boldsymbol{y}$ peuvent être mesurés. L'objectif des paragraphes suivants va être de modifier l'algorithme SensorSpace afin de satisfaire à la formulation présentée ci-dessus. L'application directe de cet algorithme n'est pas possible à cause de la forme retenue de la matrice S. En effet SensorSpace est basé sur l'utilisation d'une matrice de mesure de la forme $S = CY$ où $C \in \mathcal{M}_C$ et Y représente le champ sur lequel les mesures peuvent être faite.

Cependant, avec $S = S_{\text{ext}} = \begin{pmatrix} S_i \\ S_{i-k} \end{pmatrix} = \begin{pmatrix} CY_i \\ CY_{i-k} \end{pmatrix}$, une modification de SENSORSPACE est envisageable.

Tout comme SENSORSPACE, cet nouvel algorithme repose sur deux points essentiels :
— une mise à jour successive de la position des capteurs, en agissant sur un capteur à la fois,
— une évaluation rapide d'un critère adapté à l'objectif recherché.

Seul le deuxième point doit être ici traité. Pour une position donnée $C \in \mathcal{M}_C$, le critère ϵ utilisé est

$$\epsilon = ||H - RS_{\text{ext}}||_F, \tag{6.2}$$

où $R \in \mathbb{R}^{n_{\boldsymbol{x}} \times (2n_S)}$ est la meilleure matrice permettant de minimiser la norme ci-dessus. Ce dernier correspond encore à un estimateur linéaire. Pour chaque positionnement des capteurs, la matrice R doit être calculée afin d'évaluer le critère.

Il ne reste plus qu'à développer ce dernier afin d'exploiter pleinement la forme de S_{ext}, comme cela a été fait dans le chapitre précédent. La matrice R optimale est donnée par

$$R = H\left(S_{\text{ext}}\right)^{+}, \tag{6.3}$$

où S_{ext} est presque toujours bien conditionnée. Il est donc nécessaire d'évaluer le conditionnement de S_{ext} pour la position retenue des capteurs afin de savoir s'il est possible de déterminer R. Si ceci est impossible, la position suivante des capteurs est testée.

Dans le cas où cette matrice est bien conditionnée, le rang de S_{ext} correspond au nombre de lignes $2n_S$. Ceci est quasiment assuré si deux capteurs ne sont pas superposés (ce qui est pris en compte dans l'algorithme de balayage des capteurs) et si $n_{\text{snap}} \gg 2n_S$, ce qui est le cas ici. Dans le cas où il y a très peu de *snapshots*, un calcul direct est recommandé et il n'y a pas besoin d'effectuer cet effort pour *accélérer* l'algorithme. La pseudo-inverse peut ainsi s'écrire selon :

$${S_{\text{ext}}}^{+} = {S_{\text{ext}}}^{T}\left(S_{\text{ext}}{S_{\text{ext}}}^{T}\right)^{-1}. \tag{6.4}$$

En utilisant l'expression de S_{ext} faisant intervenir C, cette pseudo-inverse devient :

$${S_{\text{ext}}}^{+} = \begin{pmatrix} CY_i \\ CY_{i-k} \end{pmatrix}^{T}\left(\begin{pmatrix} CY_i \\ CY_{i-k} \end{pmatrix}\begin{pmatrix} CY_i \\ CY_{i-k} \end{pmatrix}^{T}\right)^{-1}. \tag{6.5}$$

Il ne reste plus qu'à développer les termes :

$${S_{\text{ext}}}^{+} = \begin{pmatrix} Y_i^T C^T & Y_{i-k}^T C^T \end{pmatrix}\left(\begin{array}{c|c} CY_iY_i^TC^T & CY_iY_{i-k}^TC^T \\ \hline CY_{i-k}Y_i^TC^T & CY_{i-k}Y_{i-k}^TC^T \end{array}\right)^{-1}. \tag{6.6}$$

Le calcul de la pseudo-inverse de S_{ext} de taille $2n_S \times n_{\text{snap}}$ fait ainsi intervenir l'inversion d'une matrice de taille $2n_S \times 2n_S$. La matrices $F_1 = \left(Y_iY_i^T\right)$, $F_2 = \left(Y_iY_{i-k}^T\right)$ et $F_3 = \left(Y_{i-k}Y_{i-k}^T\right)$ peuvent être calculées une fois pour toute. Ensuite, la matrice CF_jC^T (pour $1 \leq j \leq 3$) est obtenue en effectuant une restriction de F_j. Seules les lignes et les colonnes indexées par C sont retenues au vu de la structure particulière de C (un seul 1 par ligne). En notant $F_{j,r}$ la restriction associée à F_j, la pseudo-inverse recherchée devient :

$${S_{\text{ext}}}^{+} = \begin{pmatrix} Y_i^T C^T & Y_{i-k}^T C^T \end{pmatrix}\left(\begin{array}{c|c} F_{1,r} & F_{2,r} \\ \hline F_{2,r}^T & F_{3,r} \end{array}\right)^{-1}. \tag{6.7}$$

Ensuite, le reste de l'algorithme SensorSpace peut être utilisé. En particulier, il ne faut pas oublier d'effectuer une décomposition POD de H afin de réduire la dimension du problème (en retenant cependant un grand nombre de modes par rapport au nombre de capteurs utilisés). Les positions obtenues avec cet algorithme sont qualifiées de SensorSpace Dynamique. Il convient aussi de noter que la démarche effectuée peut être appliquée dans le cas où plus de 2 mesures sont gardées en mémoire.

Sur la figure 6.1, les positions obtenues avec cette méthode sont représentées dans le cas où $k = 1$. Les positions obtenues avec SensorSpace sont presque toutes aussi performantes que les positions SensorSpace Dynamique lorsqu'un estimateur linéaire exploitant ces mesures étendues est utilisé ($R = H\,(S_{\text{ext}})^{+}$). Le tableau 6.1 illustre ces performances. Seule la disposition trouvée pour 4 capteurs permet de faire *sensiblement* mieux que la position SensorSpace classique $\mathcal{C}_{\text{opt}}$ sur ce problème *Pb.Dynamique*.

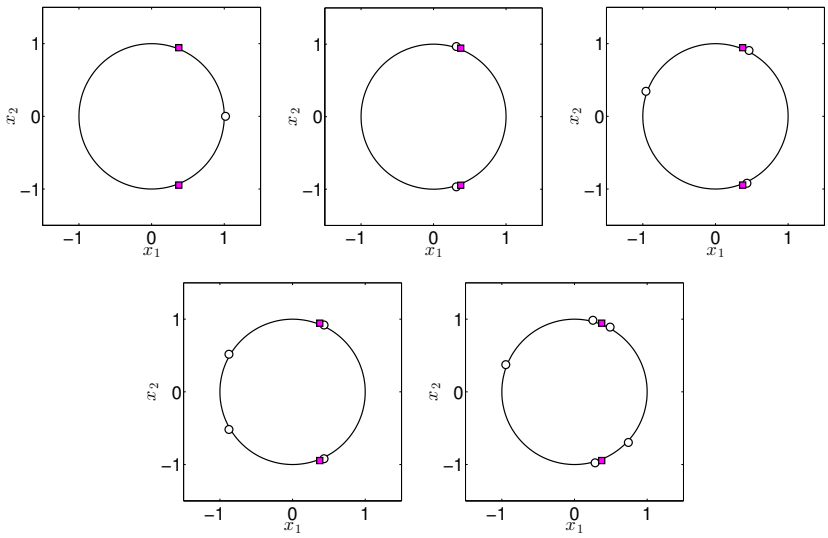

Figure 6.1 – Positions SensorSpace Dynamique $k = 1$ - $1 \leq n_S \leq 5$

Nombre de capteurs n_S	**Positions** $\mathcal{C}_{\text{opt}}$	**Positions Dynamique** k=1
1	61.1%	61.1%
2	42.3%	42.3%
3	17.5%	17.4%
4	10.1%	7.79%
5	1.15%	1.01%

Table 6.1 – Estimateur linéaire - Performances $\mathcal{C}_{\text{opt}}$ et Dynamique $k = 1$

Choix de k

Il faut maintenant choisir k, en espérant qu'il existe une meilleure valeur que 1. Étant donné que l'objectif est ensuite d'utiliser des estimateurs "adaptatifs", exploitant la position des capteurs, il est intéressant d'utiliser l'erreur d'estimation donnée par l'estimateur linéaire R précédent. De plus, comme il a été vu dans le chapitre 5, cet estimateur est associé au meilleur estimateur SOBAL généralisé ou GOBAL dans le cas plein.

Ainsi, pour un nombre donné de capteurs, SENSORSPACE Dynamique est appliqué pour différentes valeurs de k. Le meilleur positionnement et l'erreur d'estimation associé sont retenus. La plus petite valeur de k qui permet de minimiser l'erreur d'estimateur parmi les valeurs considérées est surlignée. Le tableau 6.2 permet d'illustrer les erreurs obtenues. Les cases surlignées correspondent à l'erreur minimale obtenue pour un nombre donné de capteur.

Retard k	$n_S = 1$	$n_S = 2$	$n_S = 3$	$n_S = 4$	$n_S = 5$
1	61.1%	42.3%	17.4%	7.79%	1.01%
2	59.8%	42.1%	17.0%	7.35%	1.05%
3	58.3%	41.7%	16.3%	6.51%	0.979%
4	57.2%	40.4%	15.7%	4.73%	0.825%
5	56.8%	38.8%	15.2%	2.84%	1.06%
6	57.0%	36.6%	14.9%	2.01%	1.15%
7	55.9%	33.8%	14.8%	3.38%	1.11%
8	53.4%	30.4%	14.5%	3.24%	1.02%
9	50.9%	26.5%	13.6%	2.93%	0.997%
10	48.6%	22.7%	12.1%	2.67%	1.01%
11	47.2%	20.6%	11.1%	1.83%	0.939%
12	47.6%	22.5%	12.7%	2.65%	1.05%
13	49.3%	26.6%	14.9%	3.25%	1.03%
14	51.6%	30.7%	16.0%	3.45%	1.11%

TABLE 6.2 – Réglage Dynamique - Performances SENSORSPACE Dynamique - Estimateur linéaire

La valeur de k qui permet de minimiser le plus d'erreurs de reconstruction est retenue. Cette dernière est ici de $\boxed{k = 11}$. $k = 4$ est optimale pour l'utilisation de 4 capteurs mais $k = 11$ permet quand même d'obtenir un résultat satisfaisant. Cependant, la valeur $k = 11$ correspond à un retard entre les *snapshots* mesurés de $\delta_{\text{dyn}} = 4.4s$. Ceci est une valeur remarquable puisque elle est très proche de la moitié de la période du décrochage des tourbillons dans le cas où une commande nulle est appliquée. Le fluide est ainsi dans l'état *symétrique*, c'est à dire dans un état *fortement différent* de l'état présent. Il est ainsi concevable que cette nouvelle mesure contienne plus d'informations exploitables. Les méthodes usuelles pour obtenir un tel k font intervenir le calcul de matrices de corrélation.

Les positions obtenues avec $k = 11$ sont représentées sur la figure 6.2. Ces dernières sont bien différentes des positions SENSORSPACE statique (figure 2.14) et des positions SENSORSPACE Dynamique $k = 1$ (figure 6.1).

6.1.3 Cas s_i/s_{i-k}

L'extension de mesure fait que le problème *Pb.Dynamique* est formalisable selon *Pb.GoalOriented*. Trois méthodes vont être utilisées pour répondre à ce problème :

— SOBAL généralisé,
— GOBAL,
— GOC

Le choix a été fait de ne pas tester en détail CB.GOBAL mais ce dernier est bien évidemment utilisable et donnera de bons résultats. Seul SOBAL généralisé sera traité en détail (étude complète des trois multiplicateurs).

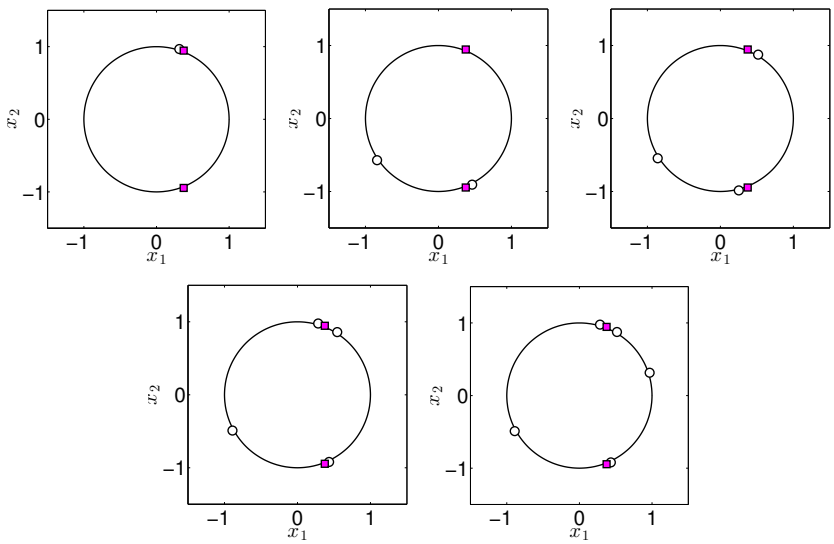

FIGURE 6.2 – Positions SENSORSPACE Dynamique $k = 11$ - $1 \leq n_S \leq 5$

Utilisation de SOBAL généralisé

La première approche considérée va exploiter l'algorithme SOBAL dans le cas généralisé. Cette approche est moins optimale que GOBAL mais elle est plus rapide. De plus, SOBAL généralisé n'a jamais été illustré jusqu'à présent. Comme présenté dans le chapitre 4, l'algorithme SOBAL peut être vu comme un algorithme manipulant deux dictionnaires D_1 et D_2 couplés. Afin de pouvoir appliquer cet algorithme au problème *Pb.Dynamique* sans modifications supplémentaires, il faut :

— trouver une formulation appropriée de S, la matrice des données mesurées,
— trouver une formulation appropriée de H, la matrice des données à estimer,
— et surtout, trouver un lien matriciel linéaire J entre les deux.

Deux manières d'adapter ce problème à la formulation requise sont considérées. La première s'efforce de trouver une telle matrice J adaptée aux séquences H et S naturelles. La seconde modifie le champ à estimer H afin de rendre trivial le choix de J.

Première approche - recherche de J

Pour cette approche, $S = S_{\text{ext}}$ et $H = Y_i$. Il faut trouver la matrice $J \in \mathbb{R}^{2n_S \times n_{\boldsymbol{x}}}$ qui permette de minimiser au mieux la quantité

$$||S_{\text{ext}} - JY_i||_F. \tag{6.8}$$

Comme la relation $S_i = CY_i$ est disponible, il ne reste plus qu'à trouver une relation entre S_{i-k} et Y_i. Aucune information *a priori* permettant de relier Y_{i-k} à Y_i n'est disponible. L'approche retenue ici est de trouver la matrice $\tilde{C} \in \mathbb{R}^{n_S \times n_{\boldsymbol{x}}}$ minimisant la norme $||S_{i-k} - \tilde{C}Y_i||_F$. La meilleure matrice est donnée par $\tilde{C} = S_{i-k}(Y_i)^+$. Étant donné la grande différence en dimension, traduite par $n_S \ll n_{\boldsymbol{x}}$, une telle matrice est prévue d'exposer de *bonnes performances*. L'erreur relative effectuée, définie par $||S_{i-k} - \tilde{C}Y_i||_F / ||S_{i-k}||_F$, n'est *que* de 0.02%. La pseudo-inverse à déterminer peut être rédhibitoire selon les dimensions du problème considéré. Dans ce cas, la deuxième approche sera plus appropriée.

La matrice réalisant le lien recherchée est

$$\boxed{J^T = \left(C^T \tilde{C}^T\right)}. \tag{6.9}$$

Une matrice J réalisant directement le lien entre H et S peut être calculée selon $J = H\,(S)^+$. Ce calcul est par contre plus lourd que le précédent. En pratique, ces deux manières de définir J fournissent les mêmes résultats. Cette méthode est utilisée pour obtenir les résultats associées à SOBAL généralisé.

Deuxième approche - modification de H

Une manière d'éviter d'avoir besoin de calculer une pseudo-inverse est de modifier l'expression de H_{ext}. En remarquant que $S_i = CY_i$ et que $S_{i-k} = CY_{i-k}$, il devient évident qu'en posant

$${H_{\text{ext}}}^T = \left(Y_i^T Y_{i-1}^T\right), \tag{6.10}$$

le matrice qui relie ce H_{ext} à S_{ext} est tout simplement

$$\boxed{J = \left(\begin{array}{c|c} C & \mathbf{0} \\ \hline \mathbf{0} & C \end{array}\right)}, \tag{6.11}$$

où $\mathbf{0}$ est la matrice nulle de dimension $n_S \times n_{\text{snap}}$. La simplicité de cette formulation se fait au prix d'une augmentation de la dimension du problème. Le champ d'intérêt n'est encore que Y_i mais c'est Y_i **et** Y_{i+1} (contenues dansH_{ext}) qui vont être utilisées par l'algorithme. Cependant, une fois le dictionnaire SOBAL calculé, les temps de calculs en ligne sont quasiment inchangés.

Par contre, le problème qui est maintenant rencontré est le fait que l'estimateur cherche à minimiser l'erreur faite sur Y_i et Y_{i-1} de **manière équivalente**. En pratique, ceci impacte les résultats de manière négligeable. Une pondération pourrait être effectuée pour accorder plus d'importance à la reconstruction de Y_i mais, vu les contraintes d'utilisation de SOBAL généralisé, ceci n'aura pas d'impact (puisqu'il faut aussi pondérer les mesures). Il faudrait alors utiliser GOBAL sans modifier S_{ext} et en pondérant la partie utile de H_{ext}. Bien évidemment, une telle approche ne sera jamais considérée puisque GOBAL peut tout simplement utiliser $H = Y_i$.

Autres réglages

Aucune modification de l'algorithme SOBAL n'est nécessaire vu le travail effectué sur la définition des matrices S_{ext}, J et H_{ext}. Cependant, il reste quelques paramètres de réglage à définir et dont l'effet reste à étudier.

Le plus important est maintenant le choix de la parcimonie K des représentations creuses. Pour répondre au problème *Pb.Estimation*, le plus grand nombre de modes pouvant être utilisés à un instant donné était $K = n_S$. Ici, afin d'utiliser le plus grand nombre de modes selon l'algorithme OMP, le choix a été fait d'utiliser $K = 2n_S$ pour le problème $\boldsymbol{s}_i/\boldsymbol{s}_{i-1}$. K correspond au nombre de capteurs étendus.

Trois multiplicateurs sont utilisés. Il convient de remarquer que l'algorithme SOBAL généralisé converge très lentement lorsque le dictionnaire recherché est plein ($n_{\text{mul}} = 1$). Aucun comportement de ce type n'a été observé pour des multiplicateurs supérieurs. La meilleure modification à effectuer consiste à remplacer le *Codebook Update* utilisant une SVD par l'étape correspondante de l'algorithme MOD. Ainsi, le dictionnaire est mis à jour (après un remplacement possible des modes non-utilisés) selon

$$\Phi = HA^+ \tag{6.12}$$

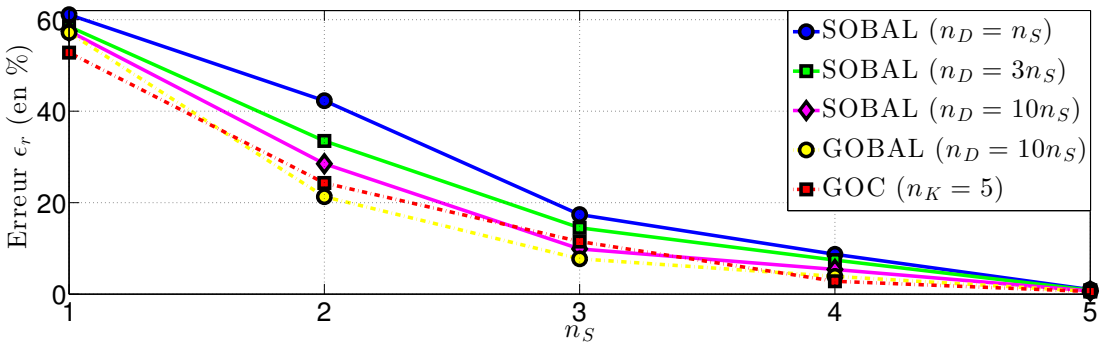

FIGURE 6.3 – Comparaison - Performances Dynamique $k = 1$

La spécificité de l'étape *Codebook Update* est que A est modifiée en même temps que Φ. Ceci est nécessaire dans les cas $n_{\text{mul}} > 1$ afin d'éviter d'être "enfermé" dans un minimum local. Dans le cas plein, cela revient à ajouter un handicap supplémentaire et ralentir ainsi la convergence.

Utilisation de GOBAL

Afin de diminuer l'erreur d'estimation relative, il est possible de rajouter des degrés de liberté (via le découplage des dictionnaires) au détriment d'un temps de calcul plus long. Ceci est fait via l'algorithme GOBAL.

Ce dernier ne nécessite que quelques réglages supplémentaires. Tout comme l'algorithme SOBAL, la parcimonie est choisie telle que $K = n_{S_{\text{ext}}}$, le nombre de mesures effectives avec $n_{S_{\text{ext}}} = 2n_S$. La vitesse de décroissance du nombre de modes utilisés est modifiée par rapport aux informations du chapitre 3. Un plus petit nombre de modes est mis à jour lors de chaque cycle.

Un seul multiplicateur est ici utilisé afin de ne pas rallonger la durée de l'étude. Le choix est fait de considérer $n_{\text{mul}} = 10$ afin de comparer avec la meilleure performance SOBAL généralisé obtenue.

Utilisation de GOC

Tout comme GOBAL, l'algorithme GOC peut être appliqué directement avec les séquences $S = S_{\text{ext}}$ et $H = Y_i$. L'estimateur linéaire est utilisé. Seul le nombre de classes doit être choisi. Le nombre retenu de classes est de $n_K = 5$.

Un tel nombre permettrait l'application future de l'algorithme CB.GOBAL afin d'effectuer une comparaison avec le cas GOBAL $n_{\text{mul}} = 10$.

Résultats

Les cas $k = 1$ et $k = 11$ sont examinés ci-dessous. Ceci est fait pour illustrer l'intérêt d'utiliser une mémorisation des mesures passées dans un cas sous-optimal et optimal. Les positions SENSORSPACE Dynamique $k = 1$ et $k = 11$ sont utilisées respectivement. Dans le cas $k = 1$, elles correspondent quasiment aux positions SENSORSPACE $\mathcal{C}_{\text{opt}}$ (statique). Seule la configuration 4 capteurs proposée avec SENSORSPACE Dynamique est réellement différente de celle proposée par SENSORSPACE.

La figure 6.3 permet de comparer ces différentes approches dans le cas $k = 1$.

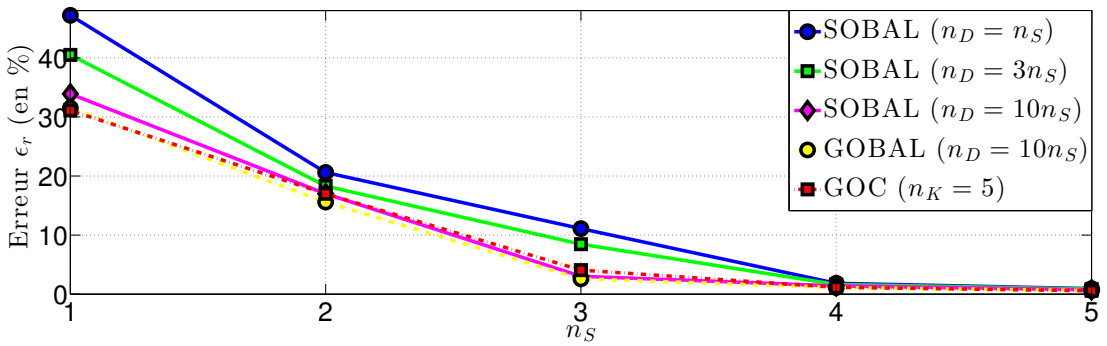

FIGURE 6.4 – Comparaison - Performances Dynamique $k = 11$

Les mesures supplémentaires ont permis d'obtenir une réduction de l'erreur d'estimation sachant que l'information ajoutée est bien moindre que si le nombre de capteurs avait été doublé dans le cas statique. Les mesures sont ici très corrélées.

Toutes ces méthodes ont su exploiter les mesures supplémentaires et ainsi faire mieux que leur contrepartie classique $\mathcal{C}_{\text{opt}}$ (voir le tableau 3.5). Le tableau 6.3 liste la totalité des résultats obtenus et permet de faire des comparaisons rapides avec les résultats présentés dans les chapitres précédents.

Nombre de capteurs n_S	SOBAL $n_D = n_S$	SOBAL $n_D = 3n_S$	SOBAL $n_D = 10n_S$	GOBAL $n_D = 10n_S$	GOC $n_K = 5$
1	61.1%	58.5%	57.5%	57.2%	52.8%
2	42.3%	33.5%	28.5%	21.3%	24.27%
3	17.4%	14.5%	9.89%	7.73%	11.5%
4	8.71%	7.45%	5.40%	3.88%	2.82%
5	0.998%	0.870%	0.679%	0.606%	0.576%

TABLE 6.3 – Comparaison - Performances Dynamique $k = 1$

GOBAL permet d'obtenir des résultats plus performants sachant que les dictionnaires sont de taille identique à ceux utilisés par SOBAL généralisé $n_{\text{mul}} = 1$.

Pour SOBAL et GOBAL, lorsqu'un seul capteur est utilisé, il est notable de remarquer que l'erreur de reconstruction relative diminue lorsque le multiplicateur augmente. Ceci provient simplement du fait que le dictionnaire intervenant dans le *Sparse Coding* possède autant de lignes que le nombre effectif de capteurs, soit 2 dans ce cas. Le calcul du mode le plus corrélé avec les mesures ne sera pas toujours le même.

Le gain obtenu avec GOC est intéressant dans le cas $n_S \geq 4$ par rapport aux deux dernières approches utilisées. Ceci annonce en effet de très bonnes performances CB.GOBAL à partir du dictionnaire de classification GOC. Au lieu d'augmenter de plus en plus n_{mul} avec l'approche GOBAL, il est intéressant de définir des classes qui utilisent des représentations moins creuses.

Bien que non exploité ici, l'approche GOC avec l'estimateur affine permet de réduire grandement l'erreur d'estimation par rapport à GOC utilisant l'estimateur linéaire.

La figure 6.4 concerne l'approche optimale associée à $k = 11$. Les mêmes remarques peuvent être effectuées concernant le comportement relatif des différentes méthodes. Le gain est ici bien plus important que dans le cas $k = 1$. Les mesures mémorisées sont moins corrélées avec les mesures présentes et, de ce fait, sont plus informatives. Les performances de SOBAL sont aussi remarquables et permettent de justifier l'utilisation de ces méthodes simples avec seulement 3 capteurs. Cependant, afin de tirer pleinement parti des mesures

effectuées, pour un nombre de modes identiques, l'approche GOBAL doit être utilisée.

Nombre de capteurs n_S	SOBAL $n_D = n_S$	SOBAL $n_D = 3n_S$	SOBAL $n_D = 10n_S$	GOBAL $n_D = 10n_S$	GOC $n_K = 5$
1	47.2 %	40.5%	33.9%	31.5%	31.1%
2	20.6%	18.3%	17.0%	15.6%	17.0%
3	11.1%	8.46%	3.02%	2.59%	4.07%
4	1.83%	1.65%	1.40%	1.12%	1.16%
5	0.930%	0.854%	0.735%	0.602%	0.570%

TABLE 6.4 – Comparaison - Performances Dynamique $k = 11$

6.1.4 Cas $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_1}/\boldsymbol{s}_{i-k_2}$

Le problème *Pb.Dynamique* a ainsi pu être traité à l'aide d'une extension des mesures suivi d'une application des algorithmes proposés. Pour conclure cette application, le cas $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_1}/\boldsymbol{s}_{i-k_2}$ est considéré. SENSORSPACE est à nouveau modifié afin de pouvoir traiter le cas où trois instants sont mis en mémoire. En utilisant le même système de notation, S est maintenant définie par :

$$S = S_{\text{ext}} = \begin{pmatrix} S_i \\ S_{i-k_1} \\ S_{i-k_2} \end{pmatrix} = \begin{pmatrix} CY_i \\ CY_{i-k_1} \\ CY_{i-k_2} \end{pmatrix} \tag{6.13}$$

La même démarche que celle présentée ci-dessus permet d'obtenir l'expression de ${S_{\text{ext}}}^+$ selon

$${S_{\text{ext}}}^+ = \begin{pmatrix} Y_i^T C^T & Y_{i-k_1}^T C^T & Y_{i-k_2}^T C^T \end{pmatrix} \left(\begin{array}{c|c|c} F_1 & F_2 & F_3 \\ \hline F_2^T & F_4 & F_5 \\ \hline F_3^T & F_5^T & F_6 \end{array}\right)^{-1}, \tag{6.14}$$

où $F_1 = CY_iY_i^TC^T$, $F_2 = CY_iY_{i-k_1}^TC^T$, $F_3 = CY_iY_{i-k_2}^TC^T$, $F_4 = CY_{i-k_1}Y_{i-k_1}^TC^T$, $F_5 = CY_{i-k_1}Y_{i-k_2}^TC^T$ et $F_6 = CY_{i-k_2}Y_{i-k_2}^TC^T$.

Seul le cas 3 capteurs est considéré ici. Il faut maintenant choisir les retards $k_1 \in \mathbb{N}^*$ et $k_2 \in \mathbb{N}^*$. L'approche retenue va être une mise à jour alternée de k_1 et de k_2. Tout d'abord, en reprenant l'étude précédente, k_2 est imposé avec $k_2 = 11$. Ensuite, les meilleures positions SENSORSPACE Dynamique et l'erreur associée sont déterminées pour différentes valeurs de k_1. Lorsqu'un minimum est trouvé, k_1 est fixé et une modification de k_2 est considérée. Ces étapes sont répétées jusqu'à ce qu'un minimum local soit obtenu.

Étant donné que 2 retards sont considérés, la formation des restrictions de Y est détaillée. En effet, pour former la matrice S_{ext}, il faut d'abord former les matrices Y_i, Y_{i-k_1} et Y_{i-k_2}. Ces dernières doivent toutes avoir le même nombre de *snapshots*. Y contient n_{snap} *snapshots* du champ à mesurer. Si $k_1 < k_2$, afin que Y_i contiennent le plus de *snapshots* possibles, ce dernier doit retenir $n_{\text{snap}} - k_2$ *snapshots* indexés de $1 + k_2$ jusqu'à n_{snap}. Ensuite, Y_{i-k_2} doit contenir les *snapshots* indexés de 1 jusqu'à $n_{\text{snap}} - k_2$. Enfin, Y_{i-k_1} doit contenir les *snapshots* indexés de $1 + (k_2 - k_1)$ jusqu'à $n_{\text{snap}} - k_1$. Si $k_1 > k_2$, il suffit d'échanger ces deux grandeurs dans les expressions ci-dessus.

La figure 6.5 représente l'évolution de l'erreur relative d'estimation dans le cas de l'estimateur linéaire SENSORSPACE Dynamique pour différentes valeurs de k_1. La barre verticale indique la valeur de k_2 qui est fixée. 2 minimums locaux sont trouvés : $k_1 = 6$

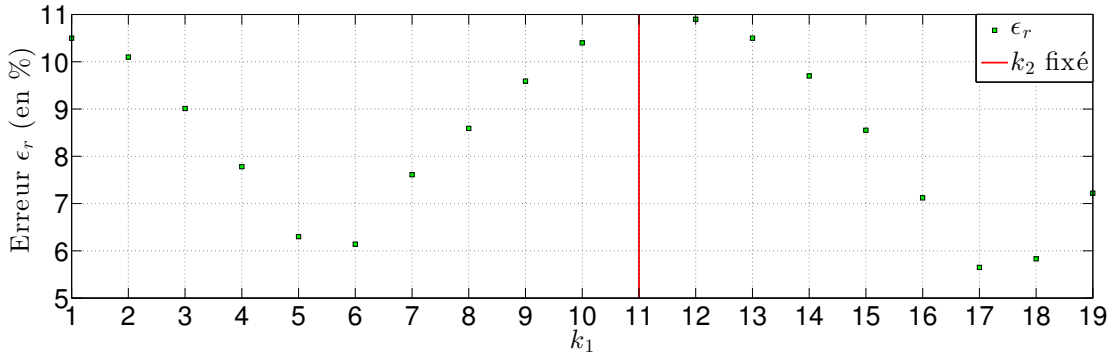

FIGURE 6.5 – Réglage Dynamique - ϵ_r pour k_1 variable et $k_2 = 11$

qui permet d'effectuer 6.14% d'erreur relative et $k_1 = 17$ qui permet d'en effectuer 5.65%. Ces deux valeurs vont maintenant être considérées.

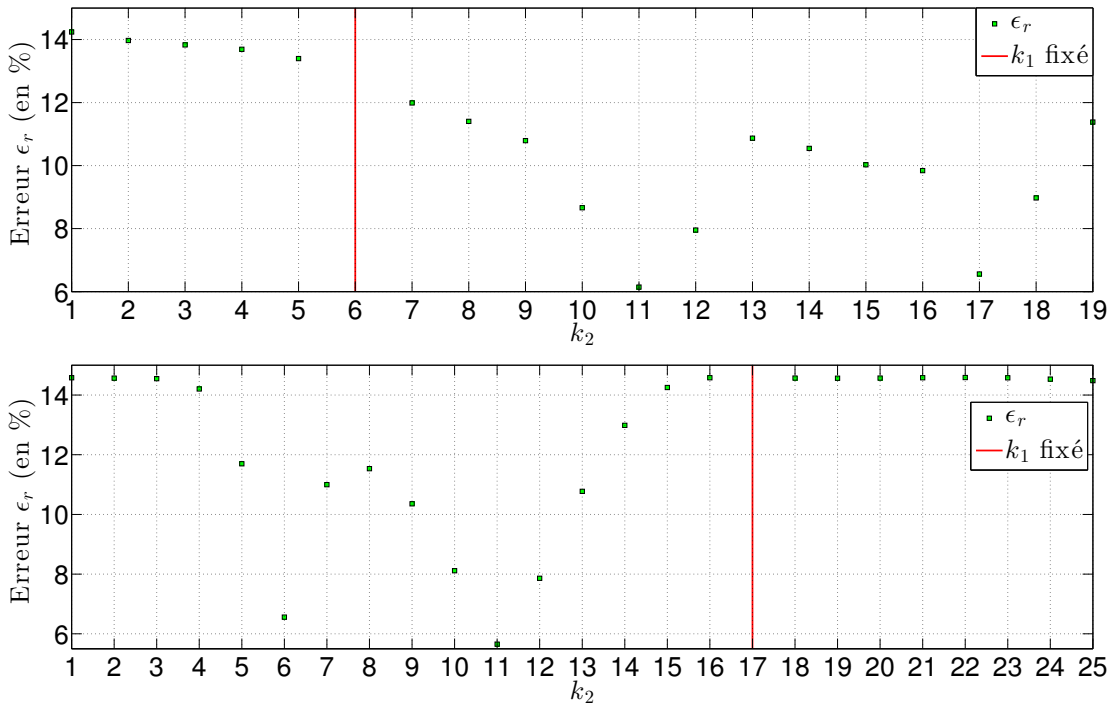

FIGURE 6.6 – Réglage Dynamique - ϵ_r pour k_2 variable et $k_1 = 6$ (haut) et $k_1 = 17$ (bas)

La figure 6.6 représente l'évolution de l'erreur relative selon k_2 pour deux valeurs de k_1. Aucun couple $\{k_1, k_2\}$ permet de faire moins d'erreur que le cas $k_1 = 17$ et $k_2 = 11$. Ces deux valeurs sont échangées afin de vérifier $k_2 > k_1$. Ceci rend la notation $\boldsymbol{s}_i/\boldsymbol{s}_{i-11}/\boldsymbol{s}_{i-17}$ du problème finalement retenu plus compréhensible.

Sur ces graphes, plus les mesures effectuées sont rapprochées (c'est à dire proches de l'abscisse 0 associée au *présent* ou proches de la barre verticale associée à une mesure figée), plus l'erreur relative associée est élevée. Ceci est cohérent avec le fait que de telles mesures sont fortement corrélées. Il faut aussi remarquer que lorsque $k_1 = 17$, choisir une valeur $k_2 > k_1$ ne permet pas de réduire notablement l'erreur sur la plage considérée. Ces mesures sont donc trop *anciennes* pour être exploitées par l'approche retenue.

Cette démarche a permis de sélectionner $k_1 = 11$ et $k_2 = 17$ pour le cas 3 capteurs. Une erreur relative de 5.65% est ainsi obtenue dans le cas plein (estimateur linéaire). Le cas $k_1 = 6$ et $k_2 = 11$ aurait aussi pu être retenu vu que l'erreur relative associée est de 6.14%. Il ne faut pas oublier que la solution obtenue correspond sûrement à un minimum local. Cependant, bien qu'un calcul exhaustif puisse être utilisé ici, en imposant une limite telle que $k_1 < k_2 < 30$, le choix d'utiliser une méthode *greedy* est fait puisqu'elle est directement utilisable dans des situations plus complexes comme le stockage des données de plusieurs instants passés.

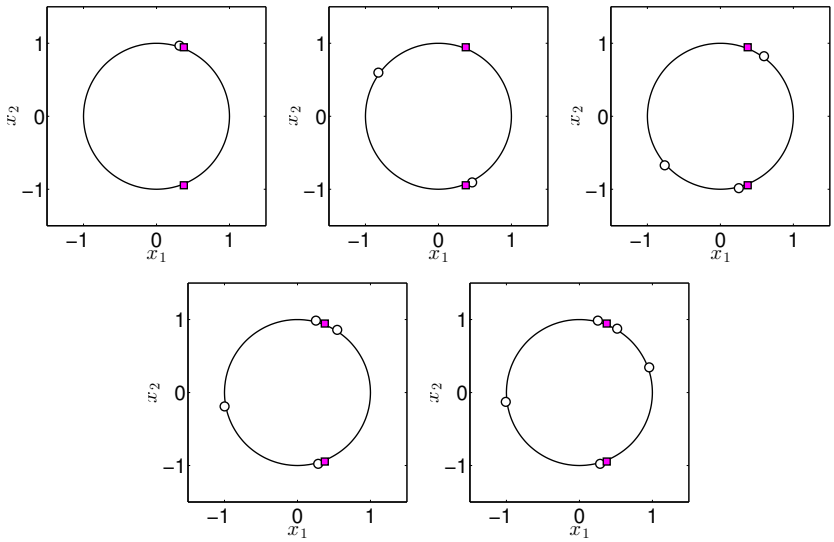

FIGURE 6.7 – Positions SENSORSPACE Dynamique $k_1 = 11$ et $k_2 = 17$ - $1 \leq n_S \leq 5$

Nombre de capteurs n_S	**Statique** $\boldsymbol{s}_i$	**Dynamique** $\boldsymbol{s}_i/\boldsymbol{s}_{i-11}$	**Dynamique** $\boldsymbol{s}_i/\boldsymbol{s}_{i-11}/\boldsymbol{s}_{i-17}$
1	62.6 %	47.2%	45.6%
2	43.8%	20.6%	18.3%
3	20.6%	11.1%	5.65%
4	11.5%	1.83%	1.25%
5	3.68%	0.930 %	0.557%

TABLE 6.5 – Performances Statique et Dynamique - Estimateur linéaire

Les positions obtenues sont représentées sur la figure 6.7. Éxcepté le cas 2 capteurs, ces positions sont très similaires à celles du problème $\boldsymbol{s}_i/\boldsymbol{s}_{i-11}$. Les erreurs relatives obtenues avec les estimateurs linéaires R des approches SENSORSPACE se trouvent dans le tableau 6.5. Le problème $\boldsymbol{s}_i$ correspond au problème classique ou statique. Les positions des capteurs utilisés correspondent à celles retenues pour répondre au problème associé. Le choix $k_1 = 11$ et $k_2 = 17$ donne le meilleur résultat pour $n_S = 3$, mais ce n'est pas forcément le cas pour les autres configurations de capteurs. Pour insister sur ce point, les configurations non-optimales pour les problèmes Dynamique sont surlignées dans le tableau 6.5.

Toutes ces erreurs correspondent aux minima obtenus en utilisant SOBAL ou GOBAL dans le cas plein. L'utilisation d'un multiplicateur $n_{\text{mul}} > 1$ garantit ensuite de meilleures performances pour chaque configuration retenue. Une étude détaillée de GOBAL n'est

pas effectuée ici.

La parcimonie de GOBAL est choisie égale au nombre de mesures étendues, c'est-à-dire $K = n_{S_{\text{ext}}} = 3n_S$. Dans le cas $n_S = 3$ et $n_{\text{mul}} = 3$, GOBAL permet d'atteindre $\boxed{3.53\%}$ d'erreur. Ceci est remarquable compte tenu de la faible taille des dictionnaires mis en jeu. En effet, le nombre de modes n'est ici que de 27 comparé aux 60 modes utilisés dans le cas SOBAL $n_{\text{mul}} = 10$ qui permettaient d'obtenir 3.02%.

Les algorithmes GOC et CB.GOBAL sont enfin utilisés sur cette configuration pour clore cette section sur l'exploitation des mesures passées. En utilisant 2 classes, GOC (estimateur linéaire) permet de faire une erreur relative de $\boxed{3.81\%}$. Le dictionnaire de classement obtenu est ensuite exploité par CB.GOBAL. Ce dernier a pour contrainte de ne produire que 25 modes (27 modes effectif en incluant les modes de classification) afin d'être comparable à l'approche GOKSVD. Dans ce cas, CB.GOBAL permet d'obtenir $\boxed{3.12\%}$ d'erreur. Comme prévu, cette approche permet d'atteindre une erreur plus faible que GOBAL avec le même nombre de modes utilisés.

6.2 Exploitation des commandes

Cette section traite maintenant du cas où les commandes appliquées sont exploitables lors de la phase de synthèse des estimateurs. Le problème général considéré est nommé *Pb.Etendu* et se formalise selon :

<u>Données</u> : Y, U, C
<u>Objectif</u> : Trouver une application $\Psi(.,.)$ définie sur $(\mathbb{R}^{n_S} \times \mathbb{R}^{nu}) \to \mathbb{R}^{n_{\boldsymbol{x}}}$ qui minimise $\epsilon_r = \|\|Y - \widehat{Y}\|\|_F / \|\|Y\|\|_F$
où $\widehat{Y} = \left(\Psi(\boldsymbol{s}_{t_1}, \boldsymbol{u}_1) \ldots \Psi\left(\boldsymbol{s}_{t_{n_{\text{snap}}}}, \boldsymbol{u}_{t_{n_{\text{snap}}}}\right)\right)$ et $S = CY$.

La matrice $U \in \mathbb{R}^{n_u \times n_{\text{snap}}}$ contient les *snapshots* des commandes appliquées. Pour un instant donné t_i, la commande appliquée est $\boldsymbol{u}_{t_i} \in \mathbb{R}^{n_u}$. Afin d'alléger la notation, ce dernier est noté $\boldsymbol{u}_i$.

Tout comme pour le problème *Pb.Dynamique*, le choix d'étendre le vecteur de mesure est effectué. Une mesure étendue correspond ici à

$$\boldsymbol{s}_{\text{ext}i} = \begin{pmatrix} \boldsymbol{s}_i \\ \boldsymbol{u}_i \end{pmatrix}. \tag{6.15}$$

L'estimateur recherché n'agit plus que sur une seule quantité : $\boldsymbol{s}_{\text{ext}i}$. Ceci permet de placer *Pb.Etendu* dans le contexte de *Pb.Estimation* et *Pb.GoalOriented*. Les méthodes de résolutions associées peuvent être appliquées.

Afin d'exploiter la notation introduite dans le chapitre précédent, ce problème est aussi qualifié de $\boldsymbol{s}_i/\boldsymbol{u}_i$. Cette notation indique que pour reconstruire le *snapshot* d'intérêt à la date i, les mesures ainsi que les commandes appliquées à l'instant i sont disponibles.

6.2.1 Placement des capteurs

Tout comme le cas précédent traitant du cas Dynamique, il semble intéressant de proposer des positions adaptées de capteurs. Les positions SENSORSPACE $\mathcal{C}_{\text{opt}}$ sont bien évidemment utilisables, mais elles sont sans doute sous-optimales puisque les informations liées aux commandes n'ont jamais été utilisées jusqu'à présent.

Une fois de plus, SENSORSPACE n'est pas directement applicable. Cependant, tout comme le problème précédent, cet algorithme est facilement modifiable pour traiter ce problème. La même démarche que celle développée dans la section précédente (voir 6.1.4) est utilisée pour réduire le plus possible le temps de calcul.

La matrice $S \in \mathbb{R}^{(n_S+n_u)\times n_{\text{snap}}}$ est ici définie par :

$$S = S_{\text{ext}} = \begin{pmatrix} CY \\ U \end{pmatrix}. \tag{6.16}$$

Le choix des capteurs n'agit que sur une partie de la matrice S_{ext}. Ce type de formulation est aussi adapté aux cas où des capteurs sont déjà placés et de nouveaux capteurs doivent être ajoutés (sans modifier les anciens). Dans cet exemple, U correspondrait aux mesures des capteurs $\mathcal{C}_{\text{f}}$ et CY correspondrait aux mesures des capteurs ajoutés.

Le critère à minimiser est :

$$\epsilon = ||H - RS_{\text{ext}}||_F, \tag{6.17}$$

où $R \in \mathbb{R}^{n_{\boldsymbol{x}}\times(n_S+n_u)}$ est donné par $R = H\,(S_{\text{ext}})^+$ pour une position donnée des capteurs, représentée par C.

Une fois de plus, il est bien probable que le rang de S_{ext} soit égal à son nombre de lignes $n_S + n_u$ qui est bien inférieure au nombre de *snapshots* n_{snap}. Ceci ne sera pas le cas si une commande appliquée est constamment nulle sur la totalité de la séquence de *snapshots*. Dans ce cas, il ne faut pas ajouter une ligne de zéros dans U.

Le développement suivant peut ensuite être fait si le conditionnement de S_{ext} le permet (pour cette configuration possible des capteurs C) :

$$\begin{aligned} (S_{\text{ext}})^+ &= \begin{pmatrix} CY \\ U \end{pmatrix}^T \left(\begin{pmatrix} CY \\ U \end{pmatrix} \begin{pmatrix} CY \\ U \end{pmatrix}^T \right)^{-1} \\ &= \begin{pmatrix} Y^TC^T & U^T \end{pmatrix} \left(\begin{array}{c|c} CYY^TC^T & CYU^T \\ \hline UY^TC^T & UU^T \end{array} \right)^{-1} \end{aligned}. \tag{6.18}$$

Il suffit de calculer une seule fois les matrices $F_1 = YY^T$, $F_2 = YU^T$ et $F_3 = UU^T$. La structure de C est ensuite exploitée. Soient :

— $F_{1,r}$ la restriction de F_1 aux lignes et aux colonnes indexées par C,

— $F_{2,r}$ la restriction de F_2 aux lignes indexées par C.

La pseudo-inverse de S_{ext} devient enfin :

$$(S_{\text{ext}})^+ = \begin{pmatrix} Y^TC^T & U^T \end{pmatrix} \left(\begin{array}{c|c} F_{1,r} & F_{2,r} \\ \hline F_{2,r}^T & F_3 \end{array} \right)^{-1}. \tag{6.19}$$

De plus Y^TC^T est la transposée de la restriction de Y aux lignes indexées par C. Pour chaque position étudiée, une matrice carrée de dimension $n_S + n_u$ doit être inversée. L'obtention de cette dernière est immédiate à travers les restrictions de F_1 et F_2. Le reste de l'algorithme SENSORSPACE est inchangé.

La figure 6.8 contient les positions obtenues avec cette approche. Ces positions sont très différentes des placements SENSORSPACE $\mathcal{C}_{\text{opt}}$ (voir figure 2.14). Les erreurs d'estimations relatives sur Y (lorsqu'un estimateur linéaire est utilisé) sont représentées dans le tableau 6.6.

Dans la colonne de droite, les positions SensorSpace Étendu sont utilisées et le meilleur estimateur linéaire pour estimer Y à partir de $\begin{pmatrix} CY \\ U \end{pmatrix}$ est retenu.

La colonne centrale correspond aux positions SensorSpace $\mathcal{C}_{\text{opt}}$ et le meilleur estimateur linéaire pour estimer Y à partir des mesures étendues est utilisé.

La colonne de gauche correspond aux positions SensorSpace $\mathcal{C}_{\text{opt}}$ et l'estimateur utilisé cherche à estimer Y à partir de CY seulement.

Il est clairement visible que l'intégration des commandes aux mesures effectuées permet une diminution de l'erreur d'estimation, même si les positions sous-optimales SensorSpace $\mathcal{C}_{\text{opt}}$ sont utilisées (comparaison de la colonne centrale avec la colonne de gauche). Cependant, sauf pour le cas 1 capteur, ces commandes supplémentaires ne permettent qu'une diminution *minime* de l'erreur.

La colonne de droite contient des erreurs bien plus faibles (pour les cas 1 à 4 capteurs). Ceci illustre clairement l'intérêt d'avoir effectué un positionnement des capteurs qui prend en compte une information de nature différente (tension de commande au lieu d'une pression). Par rapport au cas Dynamique sous-optimal $i/i-1$ (voir Tab.6.3), les informations données par les commandes sont plus utiles que celles données par une mémorisation d'une mesure passée pour les cas $1 \leq n_S \leq 3$. Enfin, par rapport au cas Dynamique *optimal* $i/i-11/i-17$ (voir Tab.6.5), seuls les cas étendus $1 \leq n_S \leq 2$ sont plus efficaces que la mémorisation de deux mesures passées. Ceci est une performance remarquable pour le cas Étendu.

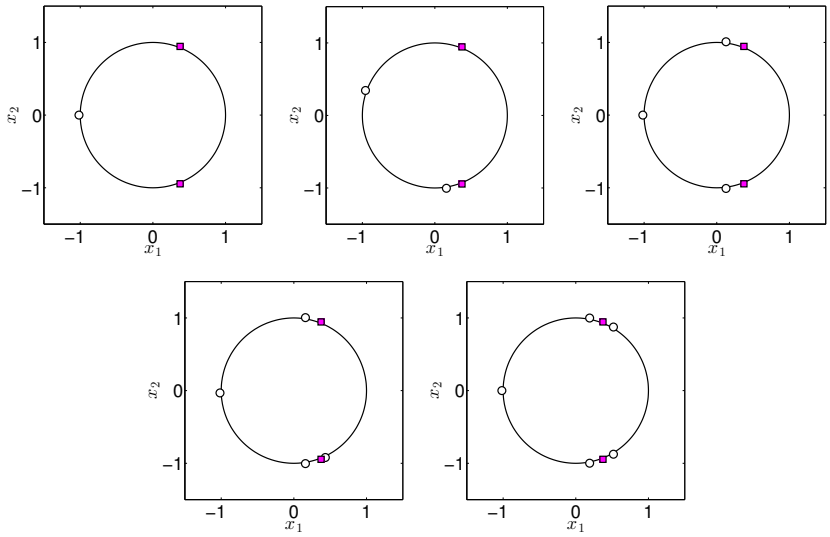

Figure 6.8 – Positions SensorSpace Étendu - $1 \leq n_S \leq 5$

Nombre de capteurs n_S	**Positions** $\mathcal{C}_{\text{opt}}$ (avec CS)	**Positions** $\mathcal{C}_{\text{opt}}$ (avec S_{ext})	**Positions Étendu** (avec S_{ext})
1	62.6%	35.9%	23.1%
2	43.8%	41.2%	15.4%
3	20.6%	20.5%	11.0%
4	11.5%	11.3%	7.84%
5	3.68%	3.43%	3.35%

Table 6.6 – Performances $\mathcal{C}_{\text{opt}}$ - Données CS et S_{ext} - Estimateur linéaire

Ces résultats (avec un estimateur linéaire) sont très prometteurs. L'utilisation des algorithmes à base de parcimonie et/ou de classes conduisent à des erreurs plus faibles.

6.2.2 Cas $\boldsymbol{s}_i/\boldsymbol{u}_i$

Les algorithmes retenus pour résoudre ce problème sont :
— GOBAL,
— CB.GOBAL.
Ce choix est arbitraire puisque SOBAL généralisé peut aussi être utilisé sans problème.

Concernant SOBAL généralisé, il faut trouver une matrice $\widetilde{J} \in \mathbb{R}^{n_u \times n_{\boldsymbol{x}}}$ qui permet de minimiser *suffisamment* la norme $||S_{\text{ext}} - \widetilde{J}Y||_F$. En utilisant la matrice $J = \begin{pmatrix} CY \\ U \end{pmatrix} Y^+$, l'erreur relative obtenue sur la description de S_{ext} est inférieure à $10^{-3}\%$ dans le cas 3 capteurs (placement Étendu). Une telle matrice peut être aussi calculée pour les autres positions retenues.

Afin de ne pas rallonger la durée de l'étude, seul GOBAL est considéré pour représenter le cas utilisation de la parcimonie. Le choix d'utiliser CB.GOBAL au lieu de GOC pour représenter l'intérêt de la classification/structure est fait pour varier les approches par rapport à l'application précédente.

Utilisation de GOBAL

La parcimonie utilisée pour GOBAL est $K = n_S + n_u$, le nombre de mesures étendues. Les multiplicateurs 1, 3 et 10 sont utilisés pour déterminer la taille des dictionnaires. Aucune modification de GOBAL n'est nécessaire.

Utilisation de CB.GOBAL

La parcimonie utilisée pour chaque sous-dictionnaire est aussi de $K = n_S + n_u$ afin d'exploiter toute l'information possible. Le nombre retenu de classes est de $n_K = 5$ afin de pouvoir comparer les résultats obtenus avec GOBAL $n_{\text{mul}} = 10$. Le nombre de modes utilisés par CB.GOBAL est tel que $n_{T.\text{st}} = n_{\text{mul}}\,(n_S + n_u)$. De ce fait, le nombre de modes utilisés par les dictionnaires type GOBAL est de $n_{\text{mul}}\,(n_S + n_u) - n_K = 10\,(n_S + 2) - 5$.

Résultats

Le tableau 6.7 permet d'illustrer les erreurs relatives ϵ_r obtenues.

Nombre de capteurs n_S	GOBAL $n_{\text{mul}} = 1$	GOBAL $n_{\text{mul}} = 3$	GOBAL $n_{\text{mul}} = 10$	GOC $n_K = 5$	CB.GOBAL $n_{T.\text{st}} = 10\,(n_S + 2)$	**Structure**
1	23.1 %	22.4%	21.5%	20.7%	20.0%	$(5, 5, 7, 3, 5)$
2	15.4%	12.8%	10.7%	10.4%	8.92%	$(7, 4, 6, 8, 10)$
3	11.0%	9.20%	8.18%	8.84%	6.58%	$(13, 9, 7, 9, 7)$
4	7.84%	6.05%	5.09%	5.43%	3.45%	$(12, 10, 13, 10, 10)$
5	3.35%	2.50%	1.77%	1.42%	0.738%	$(17, 13, 13, 13, 9)$

TABLE 6.7 – Comparaison - Performances Étendu

Pour chaque cas considéré, les performances sont meilleures que celles obtenues via l'utilisation des mesures de pression seules. Ceci illustre le fait que les algorithmes conçus

peuvent exploiter des sources d'informations de natures différentes (mesures de pression, tensions de commande). Ensuite, pour une configuration donnée, les performances s'améliorent lorsque n_{mul} augmente (GOBAL) et lorsque des classes sont introduites (CB.GOBAL).

6.2.3 Cas $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_s}/\boldsymbol{u}_i/\boldsymbol{u}_{i-k_u}$

Le problème *Pb.Etendu* a ainsi pu être traité de la même manière que *Pb.Dynamique*, à travers une extension de la notion de mesure. La suite directe de ce problème consiste à y intégrer les approches étudiées dans la partie Dynamique, c'est à dire une exploitation des mesures passées ainsi que des commandes passées.

Afin de ne pas rallonger l'étude, une seule mesure et une seule commande sont mémorisées. La matrice S résultant de l'extension de mesure est donnée par S_{ext} selon :

$$S = S_{\text{ext}} = \begin{pmatrix} CY_i \\ CY_{i-k_s} \\ U_i \\ U_{i-k_u} \end{pmatrix}, \tag{6.20}$$

où $(k_s, k_u) \in \mathbb{N}^{*2}$ sont les retards de mémorisation associés aux mesures et aux commandes respectivement.

Seul le cas 3 capteurs est considéré pour le choix de k_s et k_u. La modification de SensorSpace pour répondre à ce problème est identique à celle utilisée dans les cas précédents. La matrice $S_{\text{ext}}{S_{\text{ext}}}^T$ est développée et les sous matrices redondantes sont calculées une seule fois afin d'exploiter la structure de C.

Le choix de $\{k_s, k_u\}$ est fait selon la même méthode que le problème $i/i-k_1/i-k_2$ vu dans la section précédente. Suivant que $k_u \geq k_s$ ou que $k_u < k_s$, il faut faire bien attention à tronquer correctement les matrices Y et U pour former S_{ext}. Compte tenu des natures différentes de U et Y (la matrice C n'agit que sur Y), il ne faut pas échanger directement les indices k_u et k_s, comme dans le cas précédent.

Pour information, si $k_u \geq k_s$, Y_{i-k_s} contient les éléments de Y indexés de $1 + (k_u - k_s)$ jusqu'à $n_{\text{snap}} - k_s$. U_{i-k_u} contient les éléments de U indexés de 1 à $n_{\text{snap}} - k_u$. Ensuite, U_i et Y_i contiennent respectivement les éléments de U et Y indexés de $1 + k_u$ à n_{snap}.

Si $k_u < k_s$, U_i et Y_i s'obtiennent en échangeant k_u et k_s par rapport au cas précédent. U_{i-k_u} contient les éléments de U indexés de $1 + (k_s - k_u)$ à $n_{\text{snap}} - k_u$. Y_{i-k_s} contient les éléments de Y indexés de 1 à $n_{\text{snap}} - k_s$.

Les courbes suivantes illustrent le procédé de détermination de k_s et k_u. Les erreurs relatives obtenues avec l'estimateur linéaire $H{S_{\text{ext}}}^+$ pour différentes valeurs de k_u et k_s y sont représentées.

Le choix est fait d'agir sur k_s en premier. Bien qu'il soit tentant d'initialiser ce dernier à partir des informations obtenues dans la section précédente, on impose $k_s = 1$ au lieu de $k_s = 11$. La mémorisation des commandes peut possiblement modifier le meilleur k_s possible. La figure 6.9 illustre cette première étape. Les variations sont très régulières et 2 minima locaux sont identifiés. La valeur de k_u retenue est ainsi $k_u = 2$ et l'erreur associée y est déjà très faible (par rapport aux cas Dynamique $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_1}/\boldsymbol{s}_{i-k_2}$) avec 2.61%.

La figure 6.10 considère maintenant que $k_u = 2$ et que k_s est variable. La périodicité obtenue dans le cas Dynamique est retrouvée. Ceci se traduit par des minima locaux espacés de 6 *snapshots* environ. Cependant, l'évolution de l'erreur est plus *rapide* en ajoutant

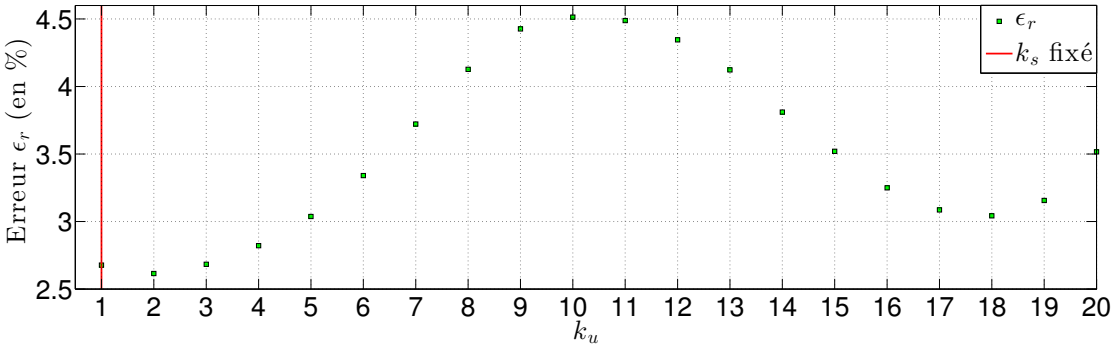

FIGURE 6.9 – Réglage Étendu-Dynamique - ϵ_r pour k_u variable et $k_s = 1$

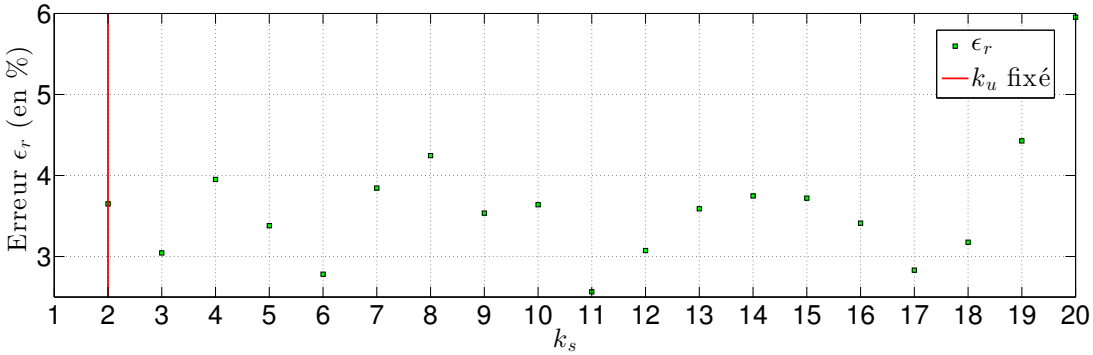

FIGURE 6.10 – Réglage Étendu-Dynamique - ϵ_r pour k_s variable et $k_u = 2$

des informations provenant des commandes. Le retard $k_s = 11$ est retenu puisqu'il permet d'atteindre une erreur de 2.57%. Ce retard est le même que celui constaté dans le problème $\boldsymbol{s}_i/\boldsymbol{s}_{i-k}$ précédent et confirme l'intuition initiale.

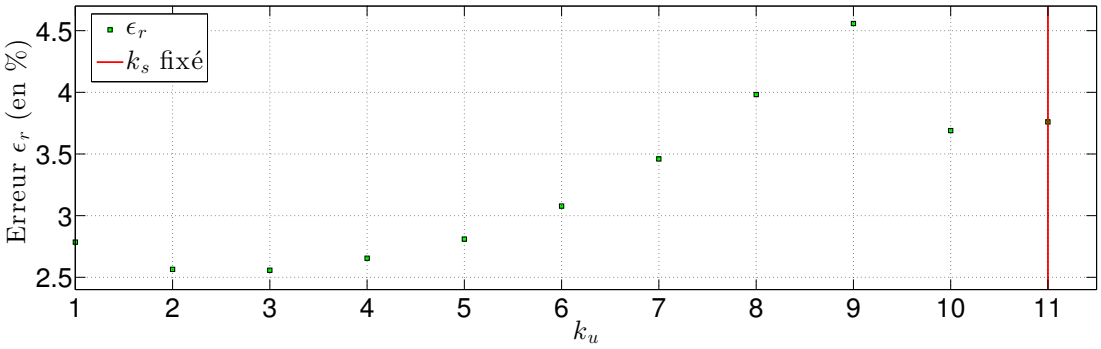

FIGURE 6.11 – Réglage Étendu-Dynamique - ϵ_r pour k_u variable et $k_s = 11$

En imposant $k_s = 11$, la figure 6.11 illustre le choix $k_u = 3$, retenu pour la suite de l'étude. Ce couple permet d'obtenir 2.56% d'erreur. Le couple $\{k_u = 3, k_s = 1\}$ permet de faire une erreur légèrement plus faible que les couples $\{k_u = 2, k_s = 11\}$, $\{k_u = 2, k_s = 11\}$ et $\{k_u = 3, k_s = 1\}$. Ces derniers peuvent aussi être considérés selon les contraintes d'implémentation.

Enfin, la figure 6.12 permet de s'assurer qu'un minimum local a bien été trouvé. Ainsi, par alternances successives, le meilleur couple obtenu est $\boxed{\{k_u = 3, k_s = 11\}}$.

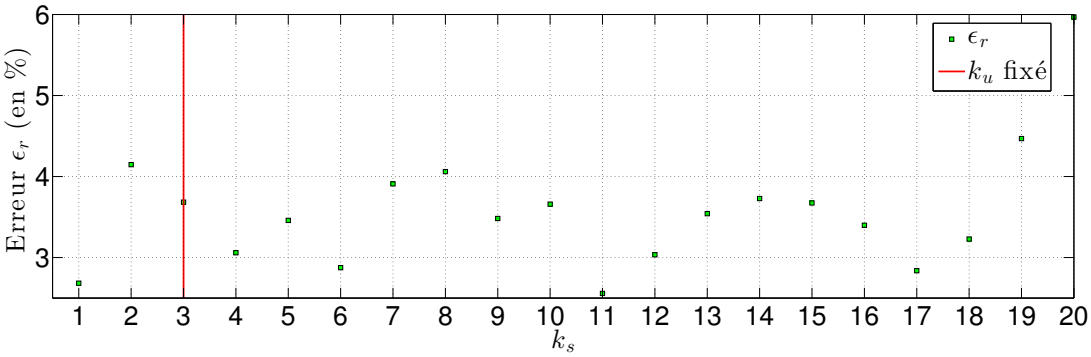

FIGURE 6.12 – Réglage Étendu-Dynamique - ϵ_r pour k_s variable et $k_u = 3$

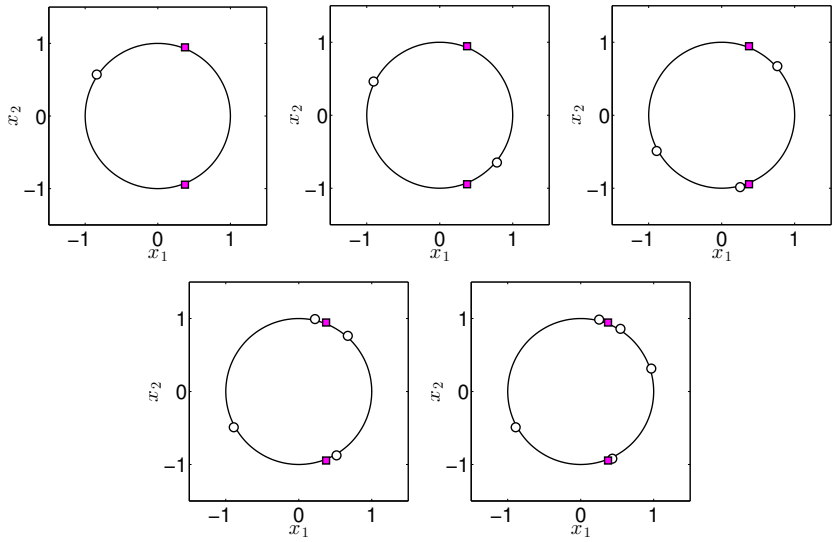

FIGURE 6.13 – Positions SENSORSPACE Étendu-Dynamique - $1 \leq n_S \leq 5$

La figure 6.13 illustre le positionnement des capteurs. Ces positions sont nommées Étendu-Dynamique. Elles sont remarquablement similaires aux positions Dynamique $\boldsymbol{s}_i/\boldsymbol{s}_{i-11}/\boldsymbol{s}_{i-17}$.

Le tableau 6.8 contient l'erreur relative d'estimation sur la séquence d'entraînement pour 2 méthodes différentes : l'utilisation de l'estimateur linéaire R retenu lors de la phase de placement et GOBAL. La parcimonie de ce dernier est égale au nombre de mesures effectives, c'est-à-dire $K = n_{S\text{ext}} = 2n_S + 2n_u$. Seul le multiplicateur $n_{\text{mul}} = 3$ est considéré.

Nombre de capteurs n_S	**Estimateur Linéaire** R	GOBAL $n_{\text{mul}} = 3$
1	13.9%	10.6%
2	8.14%	5.41%
3	2.56%	1.75%
4	1.74%	1.33%
5	0.868%	0.702 %

TABLE 6.8 – GOBAL - Performances Étendu-Dynamique

Les performances sont assez spectaculaires. Une fois de plus, il ne faut pas oublier que le choix de k_u et k_s a été fait pour le cas 3 capteurs seulement. Avec 3 capteurs seulement, moins de 2% d'erreur relative peut être obtenue sur la séquence d'entraînement à l'aide

de GOBAL. Les dictionnaires utilisés ne comportent que 30 modes.

Cependant, les performances obtenues dans les cas 4 et 5 capteurs sont moins bonnes que les cas Dynamique $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_1}/\boldsymbol{s}_{i-k_2}$ correspondants. Mémoriser 2 mesures passées de pression est plus efficace que mémoriser une mesure de pression et une commande passées. Les mesures de pression contiennent plus d'informations pour l'objectif considéré que les commandes. Dans le cas 5 capteurs, l'approche $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_1}/\boldsymbol{s}_{i-k_2}$ utilise $3 \times 5 = 15$ mesures alors que l'approche $\boldsymbol{s}_i/\boldsymbol{s}_{i-k_s}/\boldsymbol{u}_i/\boldsymbol{u}_{i-k_u}$ n'en utilise que $2 \times 5 + 2 \times 2 = 14$. Les 5 mesures de pressions sont donc plus *utiles*.

6.3 Estimation d'un champ de nature différente

Cette dernière section concerne des problèmes qui vont modifier la nature du champ estimé H. Jusqu'à présent, ce champ a toujours été le champ de pression Y. Les mesures ont toujours incorporé une restriction de ce dernier via CY. Les problèmes ci-dessous vont graduellement modifier la nature du champ à estimer.

6.3.1 Manipulation de la traînée

Tout d'abord, le coefficient de traînée C est considéré au lieu du champ de pression $\boldsymbol{y}$. Bien que ce dernier soit un *simple* scalaire, il est intéressant de ce concentrer uniquement sur son obtention à l'aide des méthodes proposées. Ceci évite le calcul du champ de pression complet lors de la détermination des dictionnaires. De plus, le coefficient de traînée **total** est considéré. Elle contient donc des informations supplémentaires reliées aux forces de viscosités non contenues dans le champ de pression. Ce *simple* scalaire n'est donc pas si trivial à obtenir.

Ces informations, non disponibles jusqu'à maintenant, vont être extraites de la nouvelle séquence d'entraînement $\Gamma \in \mathbb{R}^{1\times n_{\text{snap}}}$ définie selon :

$$\Gamma = \left[C_1 \dots C_{n_{\text{snap}}}\right], \tag{6.21}$$

où C_i est le coefficient de traînée subie par le cylindre à la date t_i pour $1 \leq i \leq n_{\text{snap}}$.

Estimation du coefficient de traînée

Le premier exemple concerne l'estimation du coefficient de traînée totale à partir des mesures de pression effectuées en surface et des commandes appliquées. Un schéma $\boldsymbol{s}_i/\boldsymbol{u}_i$ est retenu afin d'estimer la traînée. Les séquences d'apprentissage choisies sont alors :

$$H = \Gamma \quad \text{et} \quad S = S_{\text{ext}} = \begin{pmatrix} CY_i \\ U_i \end{pmatrix}. \tag{6.22}$$

Une mémorisation des mesures et des commandes peut en effet être effectuée si besoin. Les positions SensorSpace Drag sont obtenues à partir de ces séquences d'entraînement. La parcimonie de GOBAL est $K = n_{S_{\text{ext}}} = 2(n_S + n_u)$.

Le tableau 6.9 contient les erreurs relatives obtenues dans le cas de l'estimateur linéaire R et l'estimateur GOBAL.

Enfin, puisque la grandeur à reconstruire est un scalaire, il convient de représenter son évolution dans le temps.

La figure 6.14 représente l'évolution temporelle du coefficient de traînée totale C (obtenu par le simulateur) et l'estimé $\hat{C}$ à partir des dictionnaires GOBAL pour le cas $n_S = 5$, $n_{\text{mul}} = 10$. Il est remarquable que ces deux courbes soient quasiment confondues. Le coefficient de traînée totale (pression, viscosité) est estimé avec une très bonne précision.

Nombre de capteurs n_S	SensorSpace **Drag** (estim)	GOBAL $n_{\text{mul}} = 1$	GOBAL $n_{\text{mul}} = 3$	GOBAL $n_{\text{mul}} = 10$
1	3.61%	3.61%	2.84%	2.55%
2	1.89%	1.89%	1.50%	1.23%
3	0.858%	0.858%	0.611%	0.540%
4	0.652%	0.652%	0.444%	0.377%
5	0.561%	0.561%	0.422%	0.327%

Table 6.9 – Comparaison - Estimation du coefficient de traînée

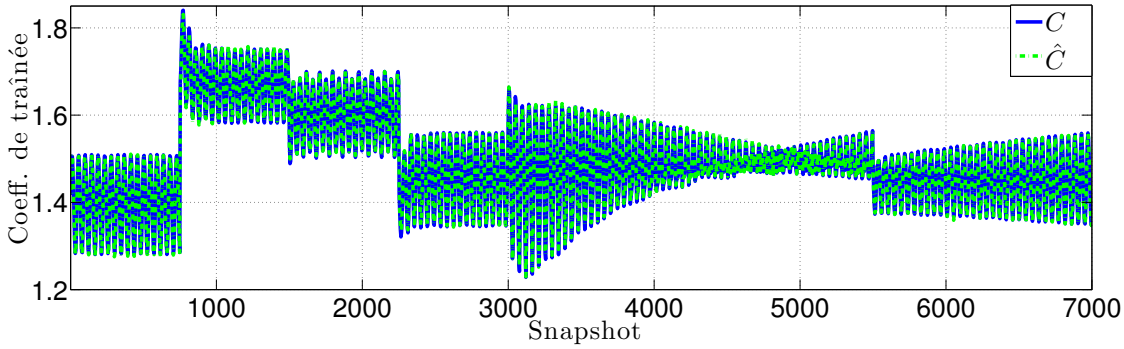

Figure 6.14 – Estimation du coefficient de traînée ($n_S = 5$)

Pour ressentir ceci, la figure 6.15 représente un zoom sur un changement de commande. **Même lors d'une phase de transition de régime, l'algorithme fonctionne correctement et donne une estimation très satisfaisante du coefficient de traînée**.

Prédiction du coefficient de traînée

La seconde proposition concerne la prédiction du coefficient de traînée. A partir des mesures présentes et passées, un état **futur** du coefficient de traînée est estimé.

Le schéma $\boldsymbol{s}_i/\boldsymbol{u}_i$ est retenu afin de prédire la traînée. De bien meilleurs résultats peuvent être obtenus si des mesures/commandes passées sont exploitées. Les séquences d'apprentissages retenues sont alors :

$$H = \Gamma_{i+1} \quad \text{et} \quad S = S_{\text{ext}} = \begin{pmatrix} CY_i \\ U_i \end{pmatrix}. \tag{6.23}$$

Les tailles des matrices sont ajustées pour être compatibles avec cette formulation. A partir

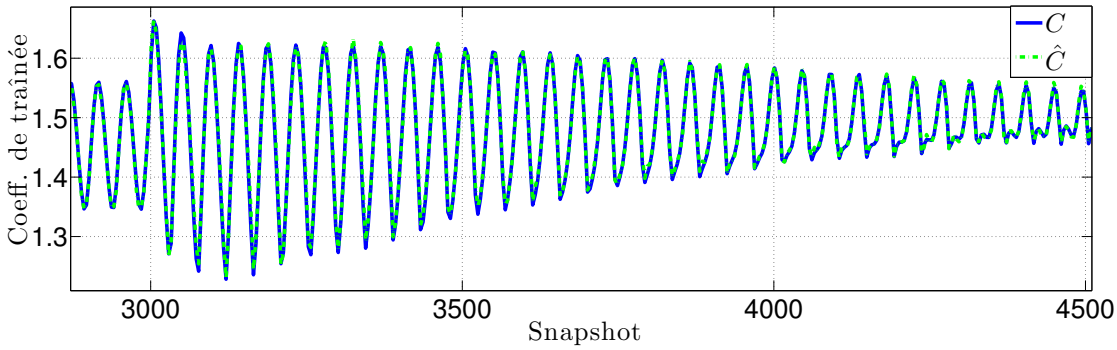

Figure 6.15 – Estimation du coefficient de traînée ($n_S = 5$) (zoom)

des informations à la date i, le coefficient de traînée à la date $i+1$ doit être estimé (prédit). Avec les données utilisées, cela revient à prédire le coefficient de traînée dans $0.4s$ à partir des données présentes.

Les résultants sont disponibles dans le tableau 6.10.

Nombre de capteurs n_S	SensorSpace **Drag** (pred)	GOBAL $n_{\text{mul}} = 1$	GOBAL $n_{\text{mul}} = 3$	GOBAL $n_{\text{mul}} = 10$
1	4.68%	4.68%	4.14%	3.83%
2	2.63%	2.63%	2.34%	2.13%
3	1.68%	1.68%	1.19%	0.980%
4	0.598%	0.598%	0.447%	0.395%
5	0.500%	0.500%	0.386%	0.316%

Table 6.10 – Comparaison - Prédiction du coefficient de traînée

Une telle approche est ainsi valide. Les erreurs relatives obtenues peuvent être inférieures à 1% dès l'utilisation de 3 capteurs. Il est aussi remarquable qu'à partir de 4 capteurs, le problème de prédiction est plus performant que le problème d'estimation précédent.

De la prédiction en temps réel peut être faite et, ensuite, être intégrée dans un schéma de commande plus classique.

6.3.2 Cas synthétique

Á titre d'exercice, cet avant-dernier exemple va traiter du problème *Pb.GoalOriented* dans un cas synthétique. L'objectif est ici d'estimer $Y \odot Y$ à partir des mesures de pression. $\odot$ représente le produit matriciel de Hadamard (encore appelé produit de Schur). A partir de mesures sur le champ de pression, l'estimateur conçu doit reconstruire ce dernier *mis au carré* (élément par élément). Bien évidemment, l'algorithme ne peut pas exploiter cette information *a priori*.

Les séquences d'apprentissages retenues sont :

$$H = Y \odot Y \quad \text{et} \quad S = CY. \tag{6.24}$$

Placement des capteurs

SensorSpace peut être utilisé directement pour répondre à ce problème. Le nombre r de modes retenus pour décrire correctement H est choisi égal à 50. Dans ce cas, l'erreur relative de description faite sur cette séquence n'est que de 1.2 10^{-3}%. La figure 6.16 illustre les positions obtenues des capteurs pour répondre à cet objectif. Sur les graphes 4 capteurs et 5 capteurs, un capteur se trouve confondu avec l'actionneur *supérieur*.

Résultats

Tous les algorithmes présentés jusqu'à maintenant peuvent être utilisés pour répondre à ce problème, même SOBAL généralisé. En effet la matrice J minimisant $||S - JH||_F$ lorsque 3 capteurs sont utilisés permet d'atteindre une erreur relative sur S de seulement 1.2 10^{-3}%. Tant que le nombre de mesures est très faible devant la dimension du champ à reconstruire, une telle approche est exploitable.

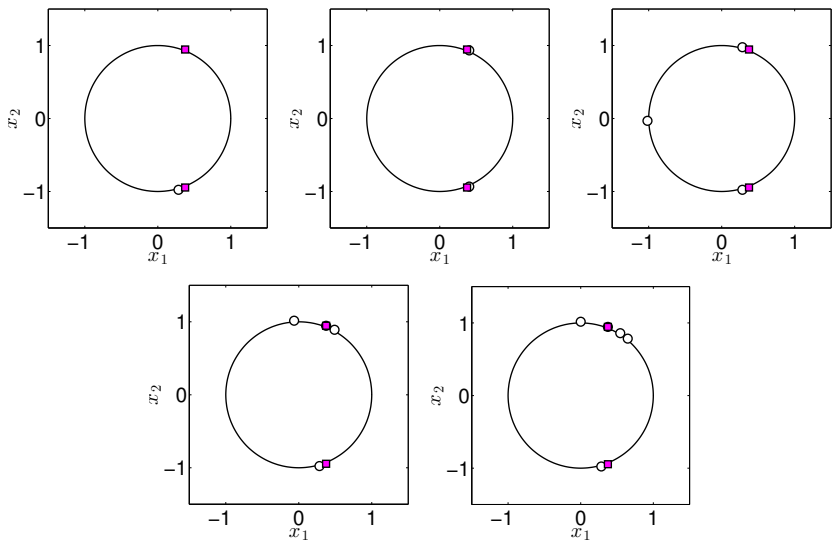

FIGURE 6.16 – Positions SENSORSPACE Square - $1 \leq n_S \leq 5$

Cette application est l'occasion d'illustrer les capacités des algorithmes à base de parcimonie pour répondre à un problème **non-linéaire**.

Approche POD

Afin de comparer les résultats avec ceux d'une approche classique, le cas POD est utilisé. La base POD de Y est calculée. Ensuite, aucun capteur n'est utilisé et la représentation réduite associée à chaque *snapshot* de Y est supposée connue. Ainsi, le cas où n_S modes sont utilisés revient à considérer que n_S capteurs sont parfaitement placés et que la représentation associée est parfaitement accessible (cas idéal). Soit n_S le nombre de modes retenus. Dans ce cas, la matrice des représentations réduites associées à Y est A_Y (matrice pleine). Ensuite, le meilleur estimateur linéaire R est retenu. Ce dernier minimise la norme $||H - RA_Y||_F$.

La figure 6.17 illustre l'évolution de l'erreur relative faite sur H suivant le nombre de modes n_S retenus (sachant que le meilleur estimateur R précédent est calculé à chaque fois). En utilisant des estimateurs linéaires classiques, ce problème nécessite l'utilisation de 21 modes (en pratique, au moins 21 capteurs) afin de pouvoir faire moins de 5% d'erreur relative sur H.

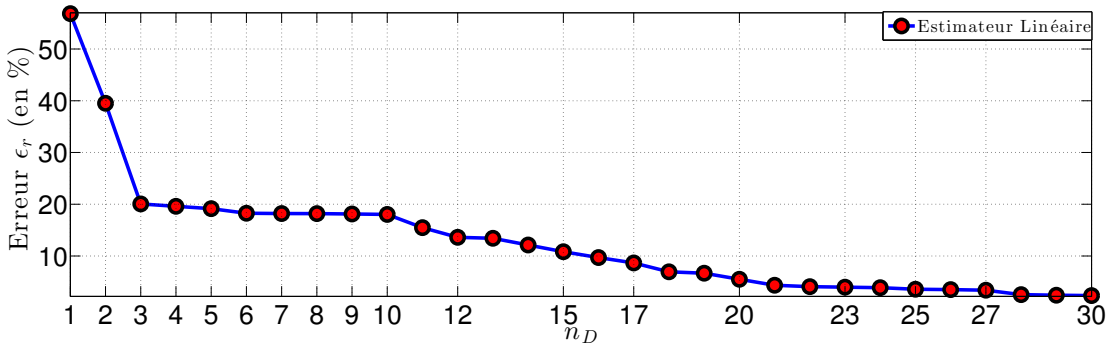

FIGURE 6.17 – POD - Performances Square selon n_D

Approches GOBAL et GOC

Les méthodes à base de parcimonie et de classes sont maintenant utilisées. Ces dernières

doivent (normalement) exhiber des performances nettement supérieures. Le fait qu'une restriction des modes est utilisée selon les mesures effectuées semble très prometteur. Cela revient à effectuer un *découpage* du problème et ainsi de proposer des modes adaptés pour chaque sous-problème.

Concernant GOBAL, la parcimonie de la représentation creuse est une fois de plus choisie égale au nombre de capteurs utilisés. Les multiplicateurs retenus sont 1, 3 et 10.

GOC est mis en œvre ici afin d'illustrer l'intérêt de pouvoir changer facilement d'estimateur. Un estimateur linéaire ainsi qu'un estimateur polynomial d'ordre 2 sont utilisés. Un tel estimateur polynomial devrait présenter de très bons résultats vu la transformation qui relie les mesures au champ à estimer.

En utilisant les notations du chapitre 4, l'estimateur utilisé pour la classe k est de la forme :

$$\Psi(\boldsymbol{s}) = C_k \boldsymbol{s} \odot \boldsymbol{s} + R_k \boldsymbol{s} + B_j \tag{6.25}$$

où C_k est une matrice de taille $\mathbb{R}^{n_{\boldsymbol{h}} \times n_S}$.

Le nombre retenu de classes est $n_K = 5$.

Le tableau 6.11 contient les erreurs d'estimations relatives ϵ_r obtenues.

Nombre de capteurs n_S	GOBAL $n_{\text{mul}} = 1$	GOBAL $n_{\text{mul}} = 3$	GOBAL $n_{\text{mul}} = 10$	GOC lin $n_K = 5$	GOC poly $n_K = 5$
1	54.6%	54.6%	54.6%	46.2%	39.5%
2	23.5%	17.3%	13.7%	9.24%	3.35%
3	19.4%	7.20%	4.35%	4.21%	1.14%
4	17.0%	8.68%	6.25%	4.86%	0.861%
5	12.7%	6.50%	5.01%	3.52%	0.738%

TABLE 6.11 – Comparaison - Performances Synthétique

Ces approches sont clairement supérieures à l'approche POD. En utilisant 3 capteurs, moins de 5% d'erreur relative peut être faite sur la séquence d'apprentissage si un multiplicateur de 10 est choisi. Le cas plein ($n_{\text{mul}} = 1$) est aussi remarquable puisqu'il permet de faire moins de 13% d'erreur relative avec 5 capteurs. Dans cette configuration, l'approche POD ferait au mieux 20% d'erreur relative.

L'augmentation du multiplicateur permet d'améliorer les performances GOBAL pour un emplacement donné des capteurs.

GOC donne ici de très bons résultats avec $n_K = 5$. Ceci illustre l'intérêt de la classification pour une telle séquence. Les estimateurs linéaire et polynomial (d'ordre 2) sont ici utilisés. Bien évidemment, l'estimateur polynomial est bien plus performant que toutes les méthodes considérées jusqu'ici. Ceci provient du fait qu'il est parfaitement compatible avec l'objet à reconstruire. Cet estimateur est capable de mettre au carré les mesures.

Existence de positions adaptées à la parcimonie

Le point le plus remarquable de cette expérience est le fait que l'augmentation du nombre de capteur avec $n_{\text{mul}} = 3$ et $n_{\text{mul}} = 10$ ne garantit pas une diminution de l'erreur d'estimation (voir les cases rouges dans le tableau 6.11). De ce fait, utiliser 4 capteurs **ne permet pas de faire mieux** que l'utilisation de 3 capteurs (avec les positions retenues ici).

De nombreuses simulations ont été faites afin d'être certain de valider ce résultat. La cause la plus probable est **l'existence** de positions de capteurs qui permettent à GOBAL

d'exhiber de **meilleures performances si les représentations réduites associées sont plus creuses**.

Bien que ce résultat fût pressenti pendant tout ce travail de thèse et que de nombreux efforts fûrent fournis pour tenter d'obtenir un algorithme donnant un tel placement, ces positions n'ont jamais été obtenues.

Afin de prouver l'existence de telles positions, un capteur supplémentaire est ajouté à la configuration 3 capteurs issue de SensorSpace. La figure 6.18 représente cette nouvelle configuration, nommée 4-*sparse*. Elle contient aussi (à droite) l'ancienne configuration 4 capteurs. L'objectif va être de comparer cette position 4-*sparse* avec la position 4-SensorSpace.

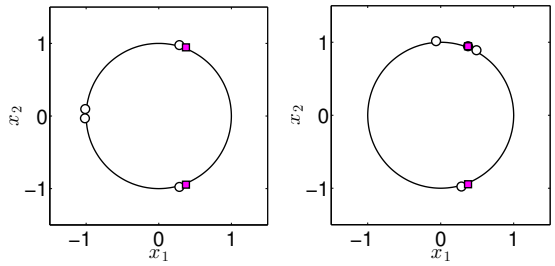

Figure 6.18 – Positions 4-*sparse* (gauche) et 4-SensorSpace (droite)

La position 4-*sparse* permet d'atteindre 18.6% d'erreur relative avec $n_{\text{mul}} = 1$ (cas plein). Cette erreur est plus grande que les 17.0% obtenu avec à la configuration 4-SensorSpace. SensorSpace permet d'obtenir la meilleure position des capteurs pour que GOBAL fasse le moins d'erreur dans le cas $n_{\text{mul}} = 1$ (le cas *plein*).

Cependant, avec $n_{\text{mul}} = 3$, GOBAL permet maintenant d'afficher 6.08% d'erreur relative avec 4-*sparse*. Cette erreur est bien inférieure aux 8.56% obtenus avec les positions 4-SensorSpace.

La position 4-*sparse* est moins intéressante que d'autres positions de capteurs pour l'approche GOBAL dans le cas plein. Par contre, elle permet d'afficher de meilleures performances que les positions 4-SensorSpace lorsque la parcimonie est effectivement utilisée ($n_{\text{mul}} > 1$). Ainsi, **l'existence de positions adaptées à la parcimonie est prouvée**.

Ceci montre aussi que SensorSpace ne produit pas de positions optimales pour GOBAL lorsque $n_{\text{mul}} > 1$. Cela dit, ce SensorSpace permet le *meilleur* placement avec $n_{\text{mul}} = 1$ et ensuite, pour une position fixe, l'augmentation de n_{mul} réduit de plus en plus l'erreur relative.

6.3.3 Estimation du champ des vitesses

Le cas de l'estimation du champ des vitesses à partir de mesures de pression effectuées en surface du cylindre est considéré. Soient $V_x \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\text{snap}}}$ et $V_y \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\text{snap}}}$ les matrices contenant les valeurs de la composante selon les axes $\boldsymbol{x}$ et $\boldsymbol{y}$ respectivement du champ des vitesses en chaque point du maillage.

Il est important de remarquer que des mesures de vitesse en surface du cylindre sont inexploitables puisque le champ des vitesses y est nul. En pratique, le positionnement réel des capteurs fait que des vitesses très faibles et peu informatives seront mesurées et ceci

rendra l'estimation du champ des vitesses dans sa totalité très délicate. En d'autres termes, le voisinage du cylindre n'est pas adapté à la mesure de la vitesse du fluide. Le couplage entre le champ des vitesses et celui de pression (comme les équations de la dynamique le montrent (voir l'équation 2.30)) doit être exploité. Une fois de plus, l'algorithme ne peut pas exploiter cette information physique (équations de Navier-Stokes) et doit calculer les estimateurs à partir des seules séquences H et S.

Bien que non-obligatoire, la séquence des commandes appliquées va aussi être utilisée. Ceci afin de manipuler 3 grandeurs de nature fondamentalement différentes : pression, vitesse, tension de commande.

Les séquences d'apprentissages retenues sont :

$$H = \begin{pmatrix} V_x \\ V_y \end{pmatrix} \quad \text{et} \quad S = S_{\text{ext}} = \begin{pmatrix} CY \\ U \end{pmatrix}, \tag{6.26}$$

où Y est la matrice des *snapshots* de pression associée à V_x et V_y et U contient les commandes appliquées.

La séquence V_x est bien plus riche que V_y dans le cas considéré. Pour illustrer ce point, il faut retenir plus de modes POD pour décrire V_x que V_y afin que l'erreur de description sur ces deux séquences soit proche. Ceci provient du fait que l'écoulement incident est orienté selon $\boldsymbol{x}$.

Les positions SensorSpace $\mathcal{C}_{\text{opt}}$ pourraient être utilisées mais elles semblent vouées à être non-optimales vu le changement de nature physique de la quantité d'intérêt. Les positions des capteurs C pour répondre à ce problème vont être déterminées. L'algorithme SensorSpace Étendu est utilisé sans modification supplémentaire.

Afin d'utiliser SensorSpace, il ne reste qu'à fournir le nombre de modes nécessaire pour décrire *correctement* H. En utilisant une décomposition SVD de cette séquence, il s'avère qu'utiliser 50 modes POD permet d'obenir une erreur relative inférieure à 0.02%.

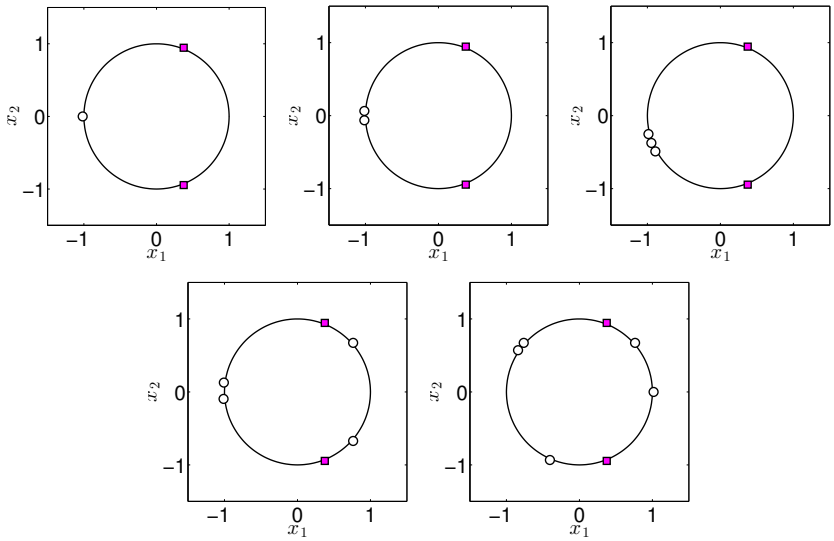

Figure 6.19 – Positions SensorSpace Fluide (C_{xy}) - $1 \leq n_S \leq 5$

La figure 6.19 contient les différentes positions des capteurs pour estimer les deux composantes du champ des vitesses. Le tableau 6.12 contient les erreurs relatives d'estimation

faites sur les différentes séquences d'entraînement considérées. Un estimateur linéaire optimal R est utilisé pour chaque configuration. Les positions utilisées sont indiquées entre parenthèses, en dessous de la séquence à estimer. Ces positions sont identifiées par les sigles :

— C_{xy}, indiquant les positions SENSORSPACE Étendu pour estimer V_x **et** V_y,
— C_x, indiquant les positions SENSORSPACE Étendu pour estimer V_x **seul**,
— C_y, indiquant les positions SENSORSPACE Étendu pour estimer V_y **seul**.

Nombre de capteurs n_S	V_x **et** V_y (C_{xy})	V_x (C_x)	V_x (C_{xy})	V_y (C_y)	V_y (C_{xy})
1	24.7%	36.4%	36.4%	24.3%	24.3%
2	11.6%	19.8%	19.8%	11.2%	11.2%
3	7.16%	14.6%	15.8%	6.72%	6.72%
4	3.91%	10.3%	11.9%	3.35%	3.38%
5	2.22%	6.22 %	6.23%	1.97%	1.97%

TABLE 6.12 – Réglage Fluide - ϵ_r pour différentes positions des capteurs

Si les positions C_{xy} sont utilisées pour estimer seulement V_x ou V_y (colonnes surlignées), les erreurs relatives obtenues ne sont que *légèrement* supérieures à celles obtenues avec les positions C_x ou C_y. Au prix d'un sacrifice sur les performances d'estimation de V_x pour 3 et 4 capteurs, les positions C_{xy} permettent d'obtenir une estimée des deux composantes du champ des vitesses.

Ce tableau illustre bien qu'il est plus difficile d'estimer correctement V_x que V_y pour un même nombre de capteurs.

Suivant les besoins de l'utilisateur, une pondération peut être appliquée afin d'accorder plus d'importance à l'estimation de V_x ou V_y. En fait, il s'avère que l'utilisation de $H = \begin{pmatrix} V_x \\ V_y \end{pmatrix}$ revient à accorder intrinsèquement plus d'importance à V_y qu'à V_x. Les algorithmes présentés jusqu'à maintenant manipulent la norme de Frobenius des matrices d'entraînement. Ainsi, vu que $||V_y||_F = 6.38\ 10^3$ et $||V_x||_F = 1.12\ 10^3$, V_y est environ 6 fois *plus important* que V_x.

La séquence à estimer est maintenant créée à l'aide d'une pondération. Soit $W \in \mathbb{R}^{2n_{\boldsymbol{x}} \times 2n_{\boldsymbol{x}}}$ une matrice diagonale définie semi-positive.

La matrice à estimer est :

$$H = H_p = W^{1/2} \begin{pmatrix} V_x \\ V_y \end{pmatrix}. \tag{6.27}$$

Le cas 3 capteurs a été utilisé pour étudier l'effet de W. La diagonale de la matrice W contient $2n_{\boldsymbol{x}}$ coefficients $\alpha \in \mathbb{R}_+^*$ suivi de $2n_{\boldsymbol{x}}$ coefficients nuls. Si $\alpha < 35$, les positions C_{xy} pondérées obtenues permettent d'atteindre 15.8% d'erreur relative sur V_x et 6.72% d'erreur relative sur V_y. Si $\alpha > 50$, les positions obtenues permettent de réduire l'erreur faite sur V_x jusqu'à 14.6% mais augmente celle faite sur V_y jusqu'à 9.52%. Ceci est le comportement attendu.

La figure 6.20 représente 4 positionnements différents des 3 capteurs de pression. Les deux positions centrales sont associées à C_{xy} pour deux différents coefficients α. Suivant la valeur choisie, ces dernières ressemblent très fortement au cas C_x (à droite) ou C_y (à

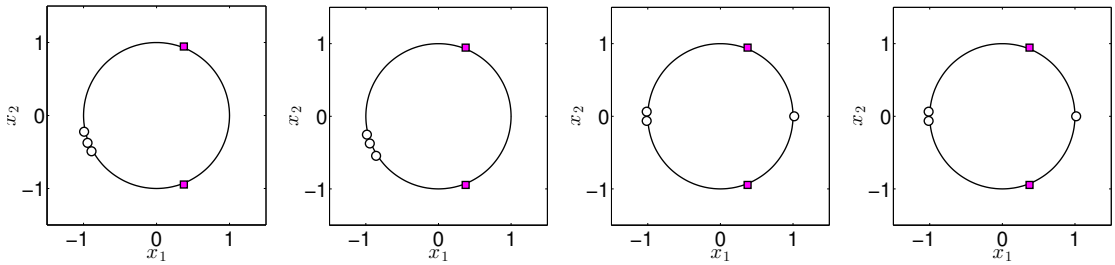

FIGURE 6.20 – Positions SENSORSPACE Fluide - C_y, C_{xy} ($\alpha < 35$), C_{xy} ($\alpha > 50$) et C_x (de gauche à droite)

gauche).

L'utilisation de méthodes à base de parcimonie/classes va maintenant permettre de réduire ces erreurs relatives. Les positions C_{xy} avec $\alpha = 1$ (sans pondération) sont utilisées dans la suite.

Utilisation de GOBAL

La parcimonie est choisie afin d'exploiter la totalité des informations disponibles. De ce fait, cette dernière est $K = n_S + n_u$, comme dans le cas Étendu statique. Seul un multiplicateur $n_{\text{mul}} = 4$ est considéré ici afin de ne pas alourdir l'étude.

Utilisation de CB.GOBAL

Afin de varier par rapport au cas précédent, l'algorithme CB.GOBAL est utilisé. Une étude détaillée des différentes combinaisons possibles du nombre de classes et de la taille des sous-dictionnaires n'est pas effectuée. Comme la méthode précédente, la parcimonie de chaque représentation creuse est choisie égale à $K = n_S + n_u$. Le nombre de classe est choisi égal à $n_K = 2$.

Résultats

Les résultats sont concentrés dans le tableau 6.13.

Nombre de capteurs n_S	GOBAL $n_{\text{mul}} = 4$	GOC $n_K = 2$	CB.GOBAL
1	23.5%	23.3%	22.8%
2	10.1%	8.31%	7.87%
3	6.48%	6.25%	5.64%
4	2.41%	2.12%	1.62%
5	1.72%	1.65%	1.36%

TABLE 6.13 – Comparaison - Performances Fluide

Avec seulement 5 mesures de pression en surface, un champ contenant 6030 éléments a pu être estimé avec une précision remarquable. En effet, l'erreur moyenne obtenue avec l'approche CB.GOBAL n'est que de 1.36% dans ce cas. Ces résultats peuvent être grandement améliorés en augmentant n_{mul} pour GOBAL et n_K pour CB.GOBAL.

CB.GOBAL permet de faire une erreur plus faible que GOBAL à nombre identique de modes. Le fait que GOC permet d'obtenir une erreur plus faible que GOBAL $n_{\text{mul}} = 4$

indique qu'ici la classification est plus bénéfique que la parcimonie. Ce comportement peut changer lorsque le nombre de classes devient trop grand.

6.4 Conclusion

Lors des chapitres précédents, un effort important a été effectué afin de réduire le plus possible l'erreur d'estimation dans le cadre de *Pb.Estimation*. L'utilisation de la parcimonie est l'étape clef pour se défaire de la limitation $n_D = n_S$. L'intégration de la position des capteurs dans l'algorithme K-SVD afin de proposer l'algorithme SOBAL correspond à la volonté d'exploiter au mieux possible l'information réellement disponible lors d'une utilisation pratique. L'utilisation de classes a permis de découper les séquences d'entraînement en sous-séquences afin d'y associer des estimateurs finement adaptés. Contrairement aux approches classiques, le cas "faible nombre de capteurs" est constamment considéré. L'intégration d'informations supplémentaires permet de réduire considérablement l'erreur effectuée quel que soit le nombre de capteurs utilisés.

Cette section a été l'occasion d'illustrer la flexibilité des algorithmes proposés face à l'intégration d'informations supplémentaires. Toujours dans un contexte de *Machine Learning*, d'apprentissage à partir de séquences d'entraînement, plusieurs problèmes ont pu être abordés.

Comme prévu, l'exploitation de mesures passées permet de réduire nettement l'erreur d'estimation. Une méthode *greedy* à base de l'algorithme SensorSpace a permis d'apprendre quelles mesures passées devaient être utilisées. Chaque méthode à base de parcimonie a pu s'adapter au problème et présenter des performances de plus en plus impressionnantes par rapport à la configuration pleine. L'utilisation de classes permet d'effectuer un constat similaire.

Ensuite, l'intégration d'une information de nature différente a été effectuée. Les méthodes présentées n'ont eu aucun problème à extraire les informations de commandes et de les utiliser conjointement avec les mesures de pression. De même, une mémorisation des mesures et commandes passés a affiché des performances remarquables compte tenu des tailles des dictionnaires utilisés.

Enfin, l'estimation d'une grandeur de nature fondamentalement différente des mesures a été étudiée. Jusqu'alors, les informations disponibles au niveau de l'utilisateur ont toujours incorporées une restriction du champ complet à estimer. La section correspondante de ce chapitre (6.3) a clairement illustré l'avantage d'utiliser des méthodes d'apprentissages pour répondre à des problèmes complexes.

Bien que ces approches subissent la limitation majeure d'être dépendantes des informations contenues dans les séquences d'apprentissages, elles permettent dans ce contexte de proposer des solutions à des problèmes variés d'estimation sans nécessiter un *a priori* physique. Cet *a priori* est nécessaire pour choisir la séquence d'entraînement par rapport à l'utilisation réelle attendue. Cependant, il ne l'est pas lors du positionnement des capteurs ou de la création des dictionnaires.

Des problèmes de prédiction peuvent être aussi considérés sans modifications majeures, tout comme des problèmes d'interpolation. Un remaniement des données d'apprentissage est la seule étape nécessaire pour placer les capteurs et produire les dictionnaires correspondants.

CHAPITRE 7
Conclusion

Chapitre 7

Conclusion

Contenu

7.1 Conclusion Générale

Il est maintenant opportun de conclure sur la totalité des travaux effectués. Pour observer ce travail de thèse sous une autre perspective, une description chronologique est maintenant donnée.

L'objectif premier était d'exploiter la **parcimonie** et ses propriétés, remarquables et souvent mentionnées dans ce manuscrit, afin d'effectuer une estimation performante des composantes d'un champ de pression. Au vu des contraintes de l'étude (aucun modèle dynamique, aucune hypothèse de nature statistique sur les données, peu de capteurs disponibles, estimation possible en temps réel) l'utilisation de la parcimonie était une voie qui semblait prometteuse. En effet, cela revenait à se détacher de la contrainte de devoir utiliser un système d'équations bien posé et ainsi, utiliser des représentations creuses *plus grandes* que les approches classiques. Tout ceci est possible à partir des *snapshots* du champ d'intérêt à disposition. Plusieurs obstacles ont été rencontrés et la décision de créer un dictionnaire adapté aux données a été prise. L'algorithme principal produit par ce travail de thèse est SOBAL, un algorithme très proche de K-SVD. Ce dernier a permis de répondre au problème fondateur du sujet de thèse et a pu afficher des performances de reconstruction bien supérieures à l'approche POD. Toute la complexité de cette approche est redirigée vers la phase de conception des dictionnaires. L'estimation est encore possible en temps réel, vu l'efficacité des algorithmes d'estimation creux type OMP.

L'importance de la **position des capteurs** était un facteur bien connu dès le début de ce travail. L'algorithme SOBAL peut s'adapter aux données fournies par les capteurs même si ces derniers sont mal placés. Cependant, il s'est avéré nécessaire de proposer une approche de placement capteurs capable de prendre en compte des contraintes géométriques de placement et d'exploiter pleinement les possibilités ouvertes par SOBAL. Les principales approches de placement de capteurs sont en effet limitées par la même contrainte principale que l'approche POD : l'utilisation d'autant de modes POD que de capteurs. De ce fait, l'algorithme SensorSpace a été proposé. Ce dernier permet d'effectuer un placement de capteurs pleinement compatible avec SOBAL. La contrainte *peu de capteurs* est ainsi parfaitement respectée et surmontée, dans le cadre de ce travail.

La suite de ce travail a été consacré à **l'amélioration** des algorithmes conçus.

Tout d'abord, l'algorithme SensorSpace a été largement optimisé d'un point de vue **temps de calcul**. Ceci a été fait en exploitant la redondance inhérente à la démarche SensorSpace proposée. L'algorithme a aussi été modifié pour produire des positions **robustes au bruit de mesure**.

Ensuite, l'algorithme SOBAL a été modifié afin de pouvoir agir sur une nouvelle classe de problèmes bien plus générale : **les mesures ne sont pas effectuées sur le champ à reconstruire**. Ceci peut en effet prendre en compte le bruit de mesure et les caractéristiques non linéaires des capteurs. L'algorithme SOBAL généralisé a ainsi été produit.

C'est à ce moment que l'algorithme SOBAL s'est révélé être un cas particulier d'un problème plus large. SOBAL a été vu comme un algorithme utilisant deux dictionnaires couplés. En **découplant ces deux dictionnaires**, l'algorithme GOBAL a été proposé. Ce dernier est bien plus complexe que SOBAL et nécessite diverses précautions de conception et d'utilisation. De par ces performances, GOBAL est devenu l'algorithme phare de ce travail de thèse. Couplé à SensorSpace qui a aussi été modifié pour agir sur le problème généralisé précédent, il permet d'atteindre des performances remarquables d'estimation. Cet algorithme peut facilement incorporer la démarche de SensorSpace pour être rendu robuste au bruit.

Enfin, il a été noté que l'augmentation continuelle de la parcimonie (via n_{mul}) n'était pas efficace. Afin de répondre à ce problème, l'utilisation de la **classification** a été retenue. Ainsi, à partir des données initiales, un dictionnaire de classification est formé à l'aide de l'algorithme GOC. Devant être compatible avec le problème généralisé, un tel algorithme a dû être proposé. Ce dernier permet de former des classes adaptées à un objectif donné. L'utilisation conjointe de la classification et de la parcimonie a abouti à la création de CB.GOBAL. Ce dernier est encore améliorable mais affiche déjà de très bonnes performances d'estimation.

Après cette série d'améliorations, l'objectif était d'illustrer les performances de ces algorithmes sur des nouveaux problèmes. Le cas dynamique a été traité afin de se rapprocher de la pratique commune du *filtrage*. Des données passées sont mises en mémoire et ensuite utilisées par les dictionnaires conçus. Comme prévu, ceci a grandement amélioré les performances d'estimation.

Ensuite, le cas étendu a été traité afin d'illustrer la possibilité d'exploiter des données de natures différentes : mesures de pression, commandes appliquées. Le gain obtenu est moindre que le cas dynamique mais est néanmoins très apprécié.

L'estimation de la traînée a ensuite été traité directement, sans passer par la reconstruction du champ de pression. De ce fait, le coefficient de traînée totale (incluant donc la traînée due aux forces tangentielles à la surface du cylindre) a pu être estimé de manière très convenable.

Les derniers exemples traités ont fait intervenir des champs de natures fondamentalement différentes du champ de pression. Le premier cas traité était purement synthétique et consistait à reconstruire le champ de pression *mis au carré*. Le second cas étudié était la reconstruction du champ des vitesses à partir de mesures de pressions effectuées à la surface du cylindre. Cette approche a été traitée avec GOC, GOBAL et CB.GOBAL avec un très grand succès, illustrant une dernière fois le potentiel de telles méthodes.

L'utilisation de la parcimonie s'est révélée être un outil puissant permettant d'obtenir de très bonnes performances d'estimation par rapport à une approche linéaire classique. La classification a ici permis de découper des problèmes complexes en sous-problèmes plus simples. Pour chaque classe, un estimateur linéaire ou plus complexe (tel que GOBAL) pouvait être utilisé.

Pour conclure, tout ce travail peut être vu comme une contribution à **l'association non supervisée**. A partir de deux séquences de données Y et H **quelconques**, un **lien** entre ces dernier est déterminé sous la forme de dictionnaires. SensorSpace permet de placer des capteurs pour mesurer Y en vue de reconstruire H. Via l'utilisation de la parcimonie et/ou de la classification, un estimateur de H utilisant uniquement $S = CY$ est produit. Ces algorithmes tentent de minimiser au mieux $||\widehat{H} - H||_F$. Ils ne nécessitent aucun *a priori* physique ou statistique pour fonctionner correctement.

7.2 Poursuite du travail

Un travail de thèse nécessite de faire des choix. Certaines pistes n'ont pas pu être approfondies dans le temps imparti au détriment des directions proposant les plus grandes *chances de réussite*. Cette dernière section a pour objectif de présenter de telles pistes.

Tout d'abord, SensorSpace peut encore être facilement amélioré. A partir d'une approche rapide *Worst One Out*, l'espace intial des positions possibles pourrait être nettement réduit. De cet espace, une version de SensorSpace **manipulant** 2 **capteurs à la fois** permettrait d'obtenir de bien meilleures solutions. Une réduction de l'espace possible est nécessaire à cause de la complexité associée à la manipulation conjointe de $n_S \geq 2$ capteurs.

Des positions de capteurs encore plus robustes au bruit doivent pouvoir être obtenues à l'aide de SensorSpace. Comme mentionné dans la partie 5.3.1, il est possible d'utiliser un estimateur robuste dans le critère qui **agit sur les mesures bruitées** (et non sur les mesures non-bruitées). La minimisation d'un tel critère nécessite l'utilisation de SensorSpace Goal-Oriented. Les essais dans le cas bruité avaient été réalisés avant l'évolution de l'algorithme SensorSpace. Avec plus de temps, cette approche aurait été refaite avec ces nouvelles positions.

Le placement de capteurs permettant d'exploiter la parcimonie des approches conçues est encore un problème ouvert. L'existence de telles positions de capteurs a été prouvé au chapitre 6 (lors de l'exemple synthétique). Diverses techniques ont été tentées dont de nombreuses s'appuyant sur les travaux de [1]. Malheureusement, aucun algorithme valable pour modifier SensorSpace n'a été produit. Une piste prometteuse était l'utilisation *inverse* de OMP qui utilise le dictionnaire $C\Phi$ et $\boldsymbol{y}$ pour obtenir une représentation creuse $\boldsymbol{a}$. La méthode recherchée consistait à utiliser $\boldsymbol{y}$, le support de $\boldsymbol{a}$ et Φ pour obtenir C. L'idée était de placer les capteurs afin d'imposer les supports des représentations creuses, compatibles avec des solutions SOBAL. Outre la difficulté due au couplage entre les positions des capteurs et SOBAL, la tâche s'est révélée trop délicate à gérer sur l'ensemble de la séquence. Placer les capteurs pour imposer le support d'un seul *snapshot* est possible. Cependant, placer les capteurs pour satisfaire au mieux à un ensemble de *snapshots* est beaucoup plus délicat.

Comme indiqué dans le chapitre 4, l'obtention d'un critère simplifié permettant d'ob-

tenir une information sur la structure améliorerait grandement le temps de calcul de CB.GOBAL. Une idée consistait à appliquer GOBAL à chaque classe et ensuite à étudier les couples de dictionnaires obtenus. Aucun résultat convainquant n'a été trouvé en calculant directement la cohérence des dictionnaires ainsi que des critères dérivés de celle-ci. Une piste qui semblait prometteuse consistait à compter le nombre de supports distincts des représentations creuses par classe. À partir de ce nombre, la capacité qu'a cette classe de profitier d'une augmentation de n_{mul} a tenté d'être quantifiée.

Concernant GOBAL, une amélioration possible consisterait à incorporer les idées provenant de la Double Sparsity [2]. Une telle approche pourrait grandement améliorer GOBAL puisqu'elle a permis d'améliorer la K-SVD en imposant l'utilisation de dictionnaires creux. La Double Sparsity introduit une forme de classification via cette structure creuse [3]. Elle permet aussi de réduire les effets de l'*overfitting*, rendant ainsi les dictionnaires plus robustes face à des données non contenues dans les séquences d'entraînement. GOBAL pourrait ainsi bénéficier de bonnes performances provenant de la classification et de la parcimonie.

L'utilisation de méthodes à **noyaux** [4] pourrait être très bénéfique à l'algorithme GOC et de ce fait CB.GOBAL. Une telle approche consiste à affecter les vecteurs de mesures à des vecteurs de dimensions plus élevées. Cependant, il n'y a jamais besoin d'évaluer ces vecteurs de grandes dimensions. Seuls des produits scalaires, évalués à l'aide de fonctions nommées *Kernel*, sont réalisés dans cet espace.

Dans cet espace de plus grande dimension, une classification type K-Means serait bien plus efficace. Des classes plus performantes pourraient alors être obtenues.

Un exemple parlant concerne la classification de points disposés de manière co-centrique autour d'un même centre (étude plane). Seuls 2 cercles de points sont considérés. Chaque cercle doit être associé à une seule classe. En utilisant une approche de classification linéaire, il est nécessaire d'introduire un très grand nombre de classes pour séparer les deux cercles. Cependant, en affectant à chaque point une troisième coordonnée qui représente leur distance par rapport au centre du cercle, il devient possible de séparer les deux jeux de points à l'aide d'un seul plan dans l'espace de dimension plus élevée.

L'utilisation de méthodes à noyaux dans le *Sparse Coding* et le *Codebook Update* de GOBAL pourraient aussi constituer une piste d'amélioration.

Enfin, il ne faut pas oublier que tout ce travail peut être appliqué au **contrôle d'écoulement**. Une continuation du travail effectué serait l'élaboration d'une commande en boucle fermée, **exploitant les estimateurs développés**, permettant de réduire la traînée subie par le cylindre à l'aide des actionneurs plasmas. Ces estimateurs sont en effet adaptés à une application en **temps réel** (une fois les dictionnaires calculés) et ne nécessitent qu'un très **faible nombre de capteurs** (3 ou 4) placés de manière réaliste en **surface du cylindre**, **loin des actionneurs**, pour estimer convenablement le coefficient de traînée totale.

Bibliographie

[1] Bingni W. Brunton, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Optimal Sensor Placement and Enhanced Sparsity for Classification. page 13, October 2013.

[2] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Double sparsity : Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3 PART 2) :1553–1564, March 2010.

[3] Fei Wu, Xiao-Yuan Jing, Xinge You, Dong Yue, Ruimin Hu, and Jing-Yu Yang. Multi-view low-rank dictionary learning for image classification. *Pattern Recognition*, 50 :143–154, 2016.

[4] Marcelo R.P. Ferreira, Francisco De a.T. de Carvalho, and Eduardo C. Simões. Kernel-based hard clustering methods with kernelization of the metric and automatic weighting of the variables. *Pattern Recognition*, 51 :310–321, 2016.

ÉCOLE DOCTORALE

Sciences et technologies de l'information et de la communication (STIC)

Titre : Apprentissage d'estimateurs sans modèle avec peu de mesures - Application à la mécanique des fluides

Mots clefs : Estimation sans modèle, Apprentissage, Parcimonie, Classification, Placement de capteurs

Résumé : Cette thèse traite de techniques promouvant la parcimonie pour déterminer des estimateurs performants n'utilisant les mesures que d'un très faible nombre de capteurs. La position de ces capteurs est cruciale pour de bonnes performances et doit être déterminée avec soin. Les méthodes proposées dans ce travail reposent sur l'utilisation d'une base d'apprentissage du champ d'intérêt considéré et ne nécessitent pas de modèle dynamique du système physique. Les éléments de cette base d'apprentissage sont obtenus à l'aide de mesures effectuées sur le système réel ou par simulation numérique. Se basant uniquement sur ces éléments d'apprentissage, et non sur des modèles dynamiques, les approches proposées sont générales et applicables à des systèmes issus de domaines variés.
Les approches proposées sont illustrées sur le cas d'un écoulement fluide 2-D autour d'un obstacle cylindrique. Le champ de pression dans un voisinage du cylindre doit être estimé à partir de quelques mesures de pression effectuées en paroi. En utilisant des positions préalablement fixées des capteurs, des estimateurs adaptés à ces positions sont proposés. Ces estimateurs tirent pleinement parti du très faible nombre de mesures en manipulant des représentations creuses et en exploitant la notion de classes. Des situations où les mesures ne portent pas sur le champ d'intérêt à estimer peuvent également être traitées. Un algorithme de placement de capteurs est proposé et permet une amélioration significative des performances des estimateurs par rapport à des capteurs placés *a priori*.
Plusieurs extensions sont discutées : utilisation de mesures passées, utilisation de commandes passées, estimation du champ d'une quantité d'intérêt reliée de façon non linéaire aux mesures, estimation d'un champ à valeurs vectorielles, etc.

Title : Model-less estimator learning using a limited amount of measurements - Application to fluid flows

Keywords : Model-less estimators, Machine learning, Sparsity, Classification, Sensor placement

Abstract : This thesis deals with sparsity promoting techniques in order to produce efficient estimators relying only on a small amount of measurements given by sensors. These sensor locations are crucial to the estimators and have to be chosen meticulously. The proposed methods do not require dynamical models and are instead based on a collection of snapshots of the field of interest. This learning sequence can be acquired through measurements on the real system or through numerical simulation. By relying only on a learning sequence, and not on dynamical models, the proposed methods become general and applicable to a variety of systems.
These techniques are illustrated on the 2-D fluid flow around a cylindrical body. The pressure field in the neighbourhood of the cylinder has to be estimated from a limited amount of surface pressure measurements. For a given arrangement of the sensors, efficient estimators suited to these locations are proposed. These estimators fully harness the information given by the limited amount of sensors by manipulating sparse representations and classes. Cases where the measurements are no longer made on the field to be estimated can also be considered. A sensor placement algorithm is proposed in order to improve the performances of the estimators.
Multiple extensions are discussed : incorporating past measurements, past control inputs, recovering a field non-linearly related to the measurements, estimating a vectorial field, etc..

Université Paris-Saclay
Ecole Normale Supérieure de Cachan
61 Avenue du Président Wilson / 94230 Cachan, France