Utilisation de l'apprentissage automatique en mécanique des fluides pour la réduction, la reconstruction et la prédiction orientée données du champ de vitesse fluctuante d'un écoulement

Pierre Dubois

Thèse

Pour l'obtention du grade de

Docteur de l'université de Lille

École Doctorale Sciences Pour l'Ingénieur - ED SPI 072

Univ. Lille, CNRS, ONERA, Arts et Metiers Institute of Technology, Centrale Lille, UMR 9014 – LMFL Laboratoire de Mécanique des Fluides de Lille - Kampé de Fériet, 59000 Lille, France

Présentée par

Pierre DUBOIS

Utilisation de l'apprentissage automatique en mécanique des fluides pour la réduction, la reconstruction et la prédiction orientée données du champ de vitesse fluctuante d'un écoulement

---

Sous la direction de

Thomas Gomez

Soutenue le 21 octobre 2021 à Lille

T. Sayadi	CR - Jean le Rond d'Alembert	Rapporteur
L. Cordier	DR - Pprime	Rapporteur
M. Meldi	MC - ISAE ENSMA	Examinateur
G. Mompean	PU - Université de Lille	Examinateur
L. Keirsbulck	PU - UPHF	Examinateur
L. Perret	MC - LHEEA	Encadrant
B. Podvin	CR - LISN	Invitée
T. Gomez	PU - LMFL	Directeur de thèse (invité)

numéro d'ordre ?

Remerciements

La thèse est une épreuve dont je ne soupçonnais pas la difficulté il y a trois ans. L'univers des sciences est extrêmement vaste et je pense m'y être perdu. La transition entre le milieu scolaire et la recherche a souvent pesé sur mon moral et j'ai pensé à l'abandon de nombreuses fois. Le syndrome de l'imposteur a été mon quotidien pendant ma formation (en était-ce vraiment une ?) avec une question récurrente : suis-je légitime et pertinent ? Chaque nouvel apprentissage a été source de frustration car il a remis en doute mes connaissances avec une myriade de questions sans réponses. Des expériences humaines malheureuses et des leitmotivs du type *publish or perish* ont souvent faussé la vision du scientifique que je voulais (veux ?) devenir : une personne ayant le droit de ne pas savoir, ayant le temps d'apprendre, ayant le sens du détail et ayant le sens du partage. Si je ressens principalement de la solitude à la fin du voyage, je suis assez fier du travail que j'ai produit. Je voudrais utiliser ces pages pour remercier les personnes qui m'ont aidé.

Dans la sphère onerienne, j'aimerais remercier quelques collègues. Merci Quentin d'avoir été mon point d'entrée et un confident : si j'ai eu l'occasion de faire cette thèse, c'est parce que tu as mis ta confiance en moi dès que tu m'as recruté en stage ; bon courage pour ta nouvelle aventure ! Merci Laurent pour tes réflexions qui ont contribué à ma culture scientifique ; c'est en cherchant à comprendre tes pensées que j'ai appris de mon côté. Merci Dominique pour ta passion du vélo toujours plaisante à écouter ; tes histoires contribuent à me faire voyager. Merci Jeff pour ta bonne humeur et ton rire communicatif. Merci Brigitte d'avoir été une excellente secrétaire avec qui j'ai pris beaucoup de plaisir à parler. Merci Anne pour ta gentillesse et tes anecdotes toujours enrichissantes ; le séjour à Modane m'a permis de découvrir une belle personne. Merci aussi aux autres collègues avec qui je partage peu mais dont le bonjour me fait toujours plaisir : Christophe (je serais toujours le poète), Geoffrey, Murielle, Vianney, Jérôme, Éric, Cécile, Jacques, etc.

Dans la sphère doctorale, j'ai de nombreux remerciements à faire. Tout d'abord à Célestin qui a partagé mon bureau près de deux ans. J'ai passé de très bons moments avec toi et ta bonne humeur, tes histoires, les délires (partagés avec Adrien, quelles rigolades !) sont d'excellents souvenirs. Tu m'as aidé à relativiser et des amis comme toi, ça vaut de l'or. Ton agrafeuse est aussi entre de bonnes mains. Merci Charles de m'avoir appris tant de choses sans me prendre de haut : tes petits cours au tableau, tes anecdotes, ton côté un peu Guevera et les pauses imposées (mais toujours acceptées) m'ont aidé à démarrer la thèse. J'ai toujours été en confiance avec toi car tu ne me jugeais pas. Merci Giovanni pour ton charme italien et les restaurants que tu m'as fait découvrir. Merci Rolando pour ta façon d'être et ton excellente humeur qui aident à freiner ma timidité. Merci

Lauriane pour ta gentillesse et ton implication dans le club Piou-Piou ; on se rappellera aussi du déménagement tendu mais qui s'est fait dans la bonne humeur. Merci Gwenaël pour tes mots justes et ton expérience : ça a toujours été un plaisir de discuter avec toi de tout (dont l'ENSMA, un sujet que j'adore !) et je te souhaite le meilleur. Merci Elvina avec qui je peux parler de tout et de rien ; on a bien rigolé quand même. D'ailleurs, puis-je préciser que python c'est meilleur que matlab ? Merci à Thomas d'être un scientifique humble, gentil et à l'écoute. C'est toujours un plaisir de discuter avec toi et de voir que les scientifiques ne sont pas tous imbus d'eux mêmes. Tous ces moments avec vous m'ont aidé à passer les petits moments de déprime d'un code qui ne marche pas ou d'un contre-sens dans ma démarche. Dommage que le virus soit passé par là car les sorties entre amis aux restaurants feront parties de mes meilleurs souvenirs. Je remercie aussi mes amis de l'ombre pour leur soutien : Vincent, Robin, Corentin et Simon.

Dans la sphère encadrante, merci Thomas d'avoir pris du temps pour répondre à mes mails de détresse. Si d'un point de vue scientifique nous avons peu échangé, nos échanges sur la définition du scientifique et le syndrome de l'imposteur m'ont aidé. Merci Laurent d'avoir relu intégralement mon manuscrit et de m'avoir guidé. Tu as toute ma gratitude pour le temps et l'énergie que tu m'as consacré.

Je voudrais également remercier Géraldine. Tu as été ma formatrice préférée et j'ai passé d'excellents moments pour préparer un oral pédagogique (on ne l'oubliera pas ce prof d'histoire), pour accroître mon aisance à l'oral et passer ma thèse en 180 secondes. Des heures où on s'est fendu la gueule (pour reprendre ton expression) et qui resteront dans ma mémoire.

Mes plus grands remerciements vont à la sphère familiale. Merci Papa et Maman de m'avoir encouragé sur le chemin de la science qui a débuté avec Faidherbe ; ce n'était pas une expérience facile mais quand je vois le résultat, je suis fier d'avoir eu des parents comme vous. Merci Léna pour nos liens qui se sont resserrés au fil des années ; je suis fier de ce que tu deviens. Merci Pauline pour ta joie de vivre et ton excentricité. Merci à Gérard et Blandine d'être des beaux parents géniaux avec qui j'apprends beaucoup ; merci pour ces parties de manilles, ces repas de famille et les partages (comme la bière) que j'apprécie toujours. Merci à Élodie et Thibault pour leur bonne humeur et leur accueil toujours chaleureux ; être le tonton Pierre de Nathan et Clémence me rend tellement heureux, même si j'ai gagné la réputation d'être le tonton grimace avec qui on peut faire des bêtises.

Je finis mes remerciements par toi Sophie, ma future femme. Merci pour ton soutien dans les moments les plus difficiles. Bientôt dix ans que nous sommes ensemble et de nombreux projets vont voir le jour avec la fin de cette thèse. Tu comptes énormément à mes yeux et je t'aime plus que tout au monde. Tout ce travail n'aurait pas vu le jour sans ta joie de vivre et ton optimisme qui contribuent à mon équilibre. Je te dédie cette thèse mon Amour. Je t'aime.

Table des matières

Nomenclature

Dimensions

n	dimension de l'état complet
r	dimension de l'état réduit
p	nombre de capteurs
l	nombre d'observations de l'état réduit
m	nombre de clichés

Données

$u_{[:,t]}$	état de haute dimension, dimension n
$a_{[:,t]}$	état latent, dimension r
u	matrice des clichés, dimension $n \times m$
a	matrice des états latents, dimension $r \times m$
y	matrice des clichés mesurés, dimension $p \times m$

Exposants et indices

$A_{[i,j]}$	ligne i et colonne j de la matrice A
train	données d'entraînement
test	données de test
val	données de validation
(R)	reconstruction
(P)	prédiction
s	espace d'état (state)
o	espace des observations
b	batch

Opérations sur les matrices

A^T	transposée de la matrice A
A^*	adjoint
$A^\dagger$	inverse de Moore Penrose
$\mathrm{rg}(A)$	rang
$\mathrm{Ker}(A)$	noyau
$\mathrm{Col}(A)$	espace des colonnes de A

Chapitre 2 - Réduction

ϵ	erreur d'auto-encodage **d'un** cliché
$e(.)$	encodeur, application de $\mathbb{R}^n$ dans $\mathbb{R}^r$
$d(.)$	décodeur, application de $\mathbb{R}^r$ dans $\mathbb{R}^n$
$\hat{u}_{[:,t]}$	état complet auto-encodé, dimension n
Φ_r	r premiers modes POD, dimension $n \times r$
W_e	poids d'encodage, dimension $n \times r$
W_d	poids de décodage, dimension $r \times n$
$h_{[:,t]}$	état caché, dimension r
$a_{\mu,[:,t]}$	moyenne de l'état encodé, dimension r
$a_{\sigma,[:,t]}$	écart type de l'état encodé, dimension r

Chapitre 3 - Reconstruction

$\mathcal{H}(.)$	opérateur de mesure générique
$\mathcal{G}(.)$	opérateur de reconstruction générique
$\hat{f}_r(.)$	modèle de reconstruction, application de $\mathbb{R}^p$ dans $\mathbb{R}^r$
$\hat{a}^{(\mathrm{R})}_{[:,t]}$	reconstruction de $a_{[:,t]}$, dimension 3
C	matrice de mesure, dimension $p \times n$
n_c	nombre de cellules
n_v	nombre de composantes
x	centres des cellules répétés n_v fois ; dimension $n \times n_v$
V_k	cellule de Voronoï numéro k
K_c	nombre de clusters (défaut : 500)
$\mu_{[k,:]}$	centroïde numéro k, dimension n_v
θ	bibliothèque de référence en estimation directe, dimension $p \times r$
$s_{[:,t]}$	second membre en estimation directe, dimension p
$\mathcal{D}$	ensemble de données étiquetées pour l'apprentissage
$\mathcal{E}$	erreur quadratique moyenne d'estimation
β	paramètres d'une régression linéaire, dimension $p \times r$
α	hyper-paramètre LASSO
C_B	contrainte de la boîte en SVR
ϵ_m	taille de la marge en SVR (défaut : 0.1)
ξ_t, ξ_t^*	écart à la marge en SVR
α_t, α_t^*	multiplicateurs de Lagrange en SVR
Q_{SVR}	matrice de covariance définie par un noyau en SVR
$\mathcal{Q}$	données disponibles à un nœud de l'arbre en DT
$\mathcal{I}(.)$	fonction d'impureté en DT
$\mathcal{X}(.)$	erreur quadratique moyenne en DT
θ_d	seuil de décision en DT
λ	taux d'apprentissage en GB
$d_{\max}$	profondeur maximale d'un arbre de décision en GB

Chapitre 4 - Prédiction

$\boldsymbol{f}_d(.)$	fonctionnelle de la dynamique
$\mathcal{K}_{\Delta t}(.)$	opérateur de Koopman
$\boldsymbol{F}_{\Delta t}$	flot de la dynamique
K	approximation finie de l'opérateur de Koopman, dimension $l \times l$
$B_{[MC]}$	poids de Koopman approché par moindres carrés, dimension $r \times l$
$g_i(.)$	fonction d'observation numéro i, application de $\mathbb{R}^r$ dans $\mathbb{R}$
$\mathcal{S}$	sous espace généré par les l fonctions d'observations
$G(.)$	**toutes** les fonctions d'observations, application de $\mathbb{R}^r$ dans $\mathbb{R}^l$
G	matrice des données observées, dimension $l \times m$
φ_j	fonction propre j de l'opérateur de Koopman
Φ	trajectoires des fonctions propres identifiées, dimension $l \times m$
ξ	vecteurs propres **gauches** de K, dimension $l \times l$
J	fonction coût pour EDMD DL
$m^{(h)}(.)$	modèle récursif des observations à l'horizon h, application de $\mathbb{R}^l$ dans $\mathbb{R}^l$
$m^h(.)$	modèle direct de l'erreur à l'horizon h, application de $\mathbb{R}^l$ dans $\mathbb{R}^l$
W	paramètres (poids et biais) du réseau de neurones en EDMD DL
H	horizon de prédiction maximal (défaut : 16 pas de temps sans dimension)
$\widehat{G}^{(+h)}(a_{[:,t]})$	prédiction récursive h pas de temps des observations à $t+h$, dimension l
$a^{(P)}_{[:,t]}$	prédiction récursive h pas de temps de l'état latent à $t+h$, dimension r
Q	covariance de l'erreur du modèle dynamique
P	covariance de l'erreur sur la condition initiale
R	covariance de l'erreur sur la mesure, dimension $p \times p$
Ga	gain de Kalman, dimension $r \times r$
F	fréquence d'assimilation de données

Chapitre 5 - Écoulements

U_∞	vitesse caractéristique
D	dimension caractéristique
St	nombre de Strouhal
Re	nombre de Reynolds
Δt^*	temps sans dimension entre deux clichés (défaut : 1)

Chapitre 6 - Résultats

$r_{0.8}$	dimension de l'état réduit à 80% d'énergie en POD
$p_{0.8}$	nombre de capteurs pour que les cellules de Voronoi associées couvrent 80% de la variance
I_y	intensité du bruit sur la mesure
I_a	intensité du bruit sur l'état latent
γ	ratio erreur normalisée sans bruit sur erreur normalisée avec bruit

Divers

$\mathbb{R}$	corps des réels
$\mathbb{C}$	corps des complexes
$\bar{\cdot}$	moyenne temporelle **ou** d'ensemble
Λ	matrice de valeurs propres
σ	activation sigmoïde **ou** écart type
W	paramètres d'un réseau de neurones
$(\boldsymbol{x}, \boldsymbol{y})$	données étiquetées (pour illustration)
$\mathcal{L}$	fonction coût **ou** vraisemblance
Ψ	modes dominants issus d'une SVD
$\mathcal{N}(\mu, \Sigma)$	loi normale de moyenne μ et de covariance Σ
B	nombre de batchs
η	vitesse de descente
R^2	Coefficient de détermination

Acronymes

POD	Proper Orthogonal Decomposition
PCA	Principal Components Analysis
LAE	Linear Auto Encoder
LVAE	Linear Variational Auto Encoder
VAE	Variational Auto Encoder
DR	Direct Reconstruction
LM	Linear Multitask
LS	Linear Singletask
SVR	Support Vector Regression
GB	Gradient Boosting
NN	Neural Network
DMD	Dynamical Mode Decomposition
HODMD	High Order DMD
EDMD	Extended DMD
EDMD RBF	EDMD Radial Basis Function
EDMD DL	EDMD Dictionnary Learning
ASHA	Asynchronous Successive Halving
SINDy	Sparse Identification of Nonlinear Dynamics
NMSE	Normalized Mean Square Error
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion

Introduction

Contexte générale

Au vingt-et-unième siècle, l'apprentissage automatique intègre la vie de tous les jours. Les grandes avancées en matière d'intelligence artificielle permettent de réaliser des tâches qui relevaient du rêve le siècle dernier. Par exemple, on peut automatiquement créer des paréidolies [1] de Van-Gogh, animer Mona Lisa, laisser un ordinateur générer de nouvelles peintures ou colorer une photo en noir et blanc d'Albert Einstein (voir figure 1). Si Clarke (1964) écrivait que les technologies suffisamment avancées seraient indistinguables de la magie, ce n'est pas le cas de ces applications : les méthodes impliquées sont compréhensibles en termes d'algèbre linéaire, d'optimisation et de statistiques.

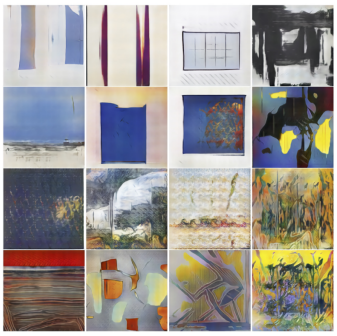
L'utilisation de l'apprentissage automatique ne se limite pas à la science de l'image. En sciences de l'ingénieur, la richesse des données disponibles a également révolutionné la manière de modéliser, prédire et contrôler des systèmes dynamiques tels que la turbulence, l'activité cérébrale, le climat, l'épistémologie [2], la finance, la robotique et l'autonomie. Ces systèmes complexes peuvent ne pas être modélisables empiriquement ou par dérivation du second principe de Newton, principalement à cause de leur caractère non linéaire et de leur comportement multi-échelles en temps et en espace. Pourtant, dans une optique de mesure, de prédiction, d'estimation ou de contrôle de ces systèmes, il est essentiel d'en fournir une caractérisation. En s'inspirant de la biologie, on comprend que la connaissance du système n'a pas besoin d'être *complète* : il suffit de penser à l'aigle qui effectue des manœuvres complexes en environnement perturbé sans connaître les équations de Navier-Stokes ou le champ de vitesse tri-dimensionnel. En formalisant cet apprentissage de réflexes à partir d'expériences, les chercheurs ont pu apporter des éléments de réponses pour le contrôle de la turbulence, le placement optimal de capteurs et d'actionneurs, l'identification de modèles dynamiques parcimonieux et la réduction de dimension. Présentée comme le quatrième paradigme de la découverte scientifique par Hey *et al.* (2009), cette science des données est motivée par les innovations en termes de stockage virtuel des données [3], les capacités de transferts, le matériel de calcul disponible et le faible coût des capteurs.

L'ingénieur d'aujourd'hui dispose d'outils pour calculer des modèles consistants qui se généralisent quantitativement et/ou qualitativement à des portions non mesurées de l'espace d'état, d'applications ou de paramètres. Toutefois, le succès

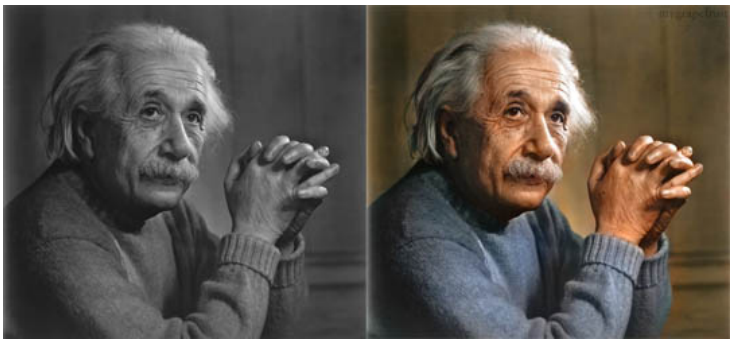
1. Une illusion, de la même manière que l'on reconnaît des formes dans les nuages ou dans une tâche d'encre.
2. Et la COVID-19 n'y a pas échappé.
3. Quasi-illimité via le cloud.

(a) Autoportrait de Van-Gogh passé dans deep dream

(b) Animation deep fake de Mona Lisa



(c) Œuvres générées par une machine. Tiré de Elgammal *et al.* (2017).



(d) Colorisation d'une photo en noir et blanc d'Einstein

Figure *1* – *Applications de l'intelligence artificielle à la science de l'image.*

des méthodes n'est avéré que sur des systèmes académiques tels que l'oscillateur de Duffing, le système de Lorenz 63 ou le sillage d'un cylindre 2D à faible nombre de Reynolds. Pour une mise à l'échelle sur des systèmes plus complexes, il faut d'abord réduire le nombre de degrés de libertés et contrer la malédiction de la dimension. L'objectif est d'extraire des structures cohérentes dans le système, qui sont des tendances de faible rang. En mécanique des fluides, la nature confirme l'existence de telles structures, comme illustré par exemple sur la figure 2 avec l'île Rishiri où l'écoulement de nuages dans le sillage d'un volcan fait le lien avec l'écoulement derrière un cylindre à bas Reynolds. Le film de l'écoulement peut alors se réécrire : plutôt que de faire défiler des images définies par des millions voire des milliards de pixels, il s'agit de faire défiler quelques images propres, appelés modes, dont la contribution change au cours du temps.

L'existence de structures particulières dans les données est le point d'entrée à l'utilisation de la science des données. L'extraction de ces modes fait écho aux efforts des mathématiciens pour simplifier un système par transformation des

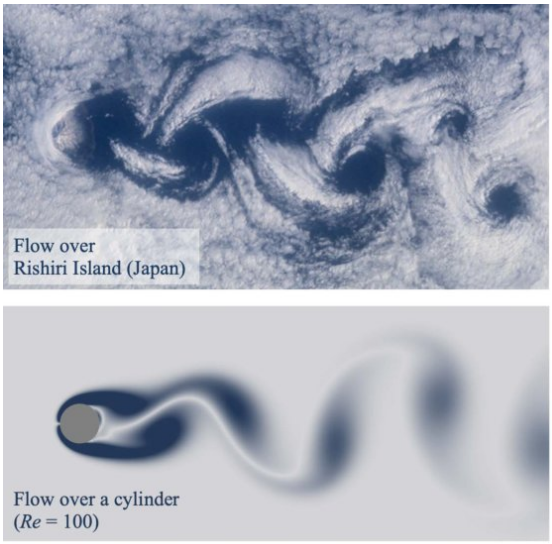


Figure 2 – *Écoulement atmosphérique rencontrant le sommet d'un volcan sur l'île de Rishiri au Japon. La dynamique, pourtant complexe, fait apparaître des structures semblables à la configuration beaucoup plus simple de l'écoulement derrière un cylindre à faible Reynolds.*

coordonnées. L'histoire de ces transformations est riche, avec des développements analytiques tels que la décomposition spectrale [4], la transformation de Fourier [5] ou les fonctions généralisées [6]. La prise en compte de données donne un nouvel élan à ces méthodes, par exemple avec l'analyse en composantes principales. Cette méthode non intrusive [7] de réduction permet d'extraire d'un nuage de points les directions décorrélées de variance maximale. En se limitant aux premières directions, la majeure partie de la variabilité est restituée tandis que les contributions peu énergétiques sont oubliées, fournissant ainsi une approximation de faible rang du système. Un autre exemple concerne l'opérateur de Koopman : introduit en 1931, cet opérateur permet de reformuler un système non linéaire de dimension finie en un système linéaire de dimension infinie. Si la propriété de linéarité est désirable, la dimension infinie de l'espace des observables rend le traitement applicatif difficile. Aujourd'hui, la richesse des données change la donne car elle permet la définition d'approximations finies de l'opérateur pour de nombreuses applications en contrôle. La décomposition en modes dynamique est un exemple d'approximation, massivement utilisée de nos jours pour le post traitement de données physiques ou le traitement vidéo.

4. Base privilégiée pour décrire une application linéaire.
5. Écriture du signal dans une base de fonctions trigonométriques.
6. Théorie des distributions.
7. Le travail s'effectue sur les données, pas sur les équations qui les ont générées.

Avec ces quelques exemples, la science des données montre un haut potentiel. Cependant, il ne faut pas laisser ces outils devenir des boites noires et garder un lien constant avec la physique. Les modèles appris doivent être interprétables et généralisables, à l'image de la seconde loi de Newton. En effet, le modèle de conservation de la quantité de mouvement $F = ma$ est interprétable car simplement basé sur la masse, l'accélération et les forces extérieures ; il est aussi généralisable car découvert avec une pomme tombée d'un arbre et pourtant applicable à de très nombreux systèmes [8]. Mathématiquement, un modèle interprétable et généralisable se traduit par un modèle parcimonieux [9], de faible dimension et robuste [10]. Malgré le potentiel grandissant de l'apprentissage automatique pour les systèmes dynamiques complexes, il faut toujours appliquer les outils intelligemment en suivant le conseil d'Einstein : "rendez les choses aussi simples que possible, mais pas plus simples".

Cadre et objectifs

Cette thèse se consacre à l'étude des méthodes orientés données pour l'estimation du champ de vitesse d'un écoulement. Cette estimation s'entend au sens de la **reconstruction** et de la **prédiction**. Pour le problème de reconstruction, des mesures à l'instant courant sont utilisées pour estimer le champ de vitesse au même instant. Pour le problème de prédiction, le champ de vitesse courant est utilisé pour estimer le champ de vitesse futur. Toutefois, plutôt que de travailler sur le champ de vitesse dans l'espace physique de haute dimension, le travail est effectué dans un espace latent de dimension réduite. La science des données intervient dans chacune de ces étapes :

→ Pour réduire la dimension de l'écoulement, on utilise un auto encodeur qui encode le champ de vitesse en un état latent puis le décode. Les données sont utilisées pour apprendre à maximiser l'information lors de l'encodage et minimiser l'erreur lors du décodage.

→ Pour reconstruire l'état latent à partir d'un vecteur de mesures, on estime la relation de passage entre l'espace de mesures et l'espace latent. Plusieurs formes de modèles sont envisagées, afin d'explorer les outils de l'apprentissage supervisé et de l'algèbre linéaire.

→ Pour prédire le futur de l'état latent, on identifie un modèle dynamique à partir des données. Plusieurs méthodes d'approximation de l'opérateur de Koopman sont étudiées, afin d'avoir un modèle dynamique linéaire fini interprétable par une décomposition spectrale.

Dans la littérature, ces trois thématiques sont *souvent* traitées de manière indépendante, avec des applications sur des écoulements simples. L'objectif de cette thèse est de rassembler les différents outils pour une estimation orientée donnée complète (réduction, reconstruction, prédiction) d'un écoulement. Les outils

8. Cette simple équation a permis (entres autres) aux hommes d'aller sur la Lune.
9. C'est-à-dire peu de termes. En anglais, on utilisera le mot *sparse*.
10. Au bruit, aux perturbations, aux valeurs aberrantes durant l'apprentissage, à de nouveaux scénarios non étudiés, etc.

sont appliqués sur quatre écoulements de complexité croissante afin d'évaluer la scalabilité des méthodes à des écoulements de plus en plus turbulents. Pour ne pas tomber dans le piège de la boîte noire et gagner en robustesse, les modèles qui interviennent sont quasi-systématiquement validés. Au niveau de la réduction, cette robustesse est recherchée au décodage pour limiter l'impact d'une mauvaise estimation de l'état latent lors du retour dans l'espace physique. Au niveau de la reconstruction et de la prédiction, la robustesse s'entend au sens de la généralisation : les modèles appris sur les données d'entraînement doivent être performants sur des données de test, non utilisées lors de l'apprentissage. Mais attention : la généralisation se fait pour des paramètres donnés, par exemple Reynolds et intensité turbulente, et l'apprentissage de modèles couvrant une grille de paramètres sort du cadre de ce manuscrit.

Description du plan

Cette thèse fait appel à des compétences transverses en mathématiques et en mécanique des fluides. La stratégie développée étant fondamentalement applicable à n'importe quel système dynamique où des données sont disponibles, le manuscrit est divisé en trois parties.

La première partie présente les obstacles à la résolution *directe* des problèmes rencontrés en mécanique des fluides. Un état de l'art est proposé, afin de montrer dans quelle mesure l'apprentissage automatique est mis à contribution pour une résolution orientée données de ces problèmes. L'application visée étant celle de l'estimation d'un l'écoulement, une attention particulière est portée sur les méthodes de reconstruction et de prédiction d'écoulements.

La seconde partie est purement méthodologique. Différentes méthodes issues de l'apprentissage automatique sont explorées et illustrées. L'objectif est de faire la synthèse d'une science *nouvelle* et de discuter le potentiel d'application à la mécanique des fluides. Trois chapitres sont présentés :

- → Le premier chapitre est consacré à la réduction de dimension. Le problème d'auto-encodage est formalisé et une méthode importante aux travaux de cette thèse est introduite : le réseau de neurones. Quatre méthodes de réduction sont développées : l'analyse en composantes principales, l'auto encodeur linéaire, l'auto encodeur linéaire variationnel et l'auto encodeur non linéaire variationnel.
- → Le second chapitre traite du problème de reconstruction. En particulier, deux des trois approches possibles d'estimation de l'état latent sont explicitées : l'approche directe et l'approche régressive. Un pan de l'apprentissage automatique est exploré en profondeur : il s'agit de l'apprentissage supervisé, permettant d'apprendre une relation entrée - sortie à partir de données.
- → Le troisième chapitre s'intéresse au problème de prédiction. L'objectif est de synthétiser les outils pour l'identification d'un modèle dynamique à partir

de données. Des méthodes d'approximation de l'opérateur de Koopman sont développées, chacune se différenciant par la manière d'observer l'état latent. On précise également l'algorithme d'assimilation de données qui permet de mettre à jour les prédictions du modèle dynamique par les reconstructions.

La troisième partie du manuscrit est une partie de résultats. Elle se concentre sur l'estimation orientée données de quatre écoulements de complexité croissante : l'écoulement derrière un cylindre 2D à bas Reynolds, une couche de mélange spatiale, l'écoulement derrière un cylindre 3D à Reynolds modéré et l'écoulement derrière une tour plongée dans une couche limite atmosphérique. Trois chapitres sont proposés :

→ Le premier chapitre présente les écoulements et les données en entrée des algorithmes de la partie précédente.

→ Le second chapitre détaille les résultats pour la réduction, la reconstruction et la prédiction des écoulements. La robustesse lors du décodage de l'état latent et la robustesse des méthodes d'estimation régressives sont étudiées. On présente également les scores de prédictions pour différentes approximations finies de l'opérateur de Koopman.

→ Le dernier chapitre fait une synthèse de tous les résultats. On propose également des pistes d'améliorations et des perspectives pour un pas vers des modèles physiques et robustes à d'autres configurations.

Liste de communications

— Un article paru dans *Physica D* et portant sur la prédiction continue du système chaotique de Lorenz 63 (Dubois *et al.*, 2020). Cet article est répertorié en annexe G.
— Un article soumis dans *Journal of Computational Physics*. Les résultats sont pleinement détaillés dans le manuscrit.
— Un article de conférence pour la 3AF. Cet article est répertorié en annexe H.
— En tant que co-auteur, un journal paru dans *Experiments in Fluids*. On utilise des réseaux de neurones pour reconstruire un jet 3D à partir de mesures fil chaud (Ott *et al.*, 2021).
— Un exercice de vulgarisation scientifique, avec ma thèse en 180 secondes. Le discours est disponible sur `https://webtv.univ-lille.fr/video/10928/`.

Première partie

L'utilisation de la science des données en mécanique des fluides

1. L'apprentissage automatique au service de la mécanique des fluides

Ce chapitre présente les avancées récentes de la science des données appliquée à la mécanique des fluides. L'objectif est de comprendre dans quelle mesure les approches orientées données (application de l'apprentissage automatique) peuvent compléter les approches physiques (application des principes). Cet état de l'art est effectué selon un fil rouge : l'application visée est l'utilisation du *machine learning* pour l'estimation *rapide* d'un écoulement urbain. Une première partie présente les tâches principales en mécanique des fluides et les difficultés liées à la physique complexe des écoulements. La seconde partie s'intéresse à la mécanique des fluides orientées données, une science tirant partie de la richesse des données pour résoudre des problèmes d'optimisation. La troisième partie présente plus spécifiquement l'application ayant motivée cette thèse i.e. l'estimation orientée données d'un écoulement urbain pour la navigation sécurisée d'un petit drone. La quatrième partie fait alors la synthèse des méthodes actuelles pour effectuer cette estimation et la dernière partie précise le travail de la thèse.

Sommaire

1.1 | Les problématiques en mécanique des fluides

Dans cette section, on s'intéresse aux missions principales en mécanique des fluides et à la physique complexe d'un écoulement turbulent. L'objectif est de comprendre le contexte d'une transition entre les principes de la physique et les approches orientées données.

1.1.1 | Les missions principales

Les fluides sont présents dans de nombreuses problématiques industrielles telles que la santé, la défense, le transport et l'énergie. Parmi les missions principales pour mieux comprendre la *mécanique* des fluides, on recense :

La **modélisation**. L'objectif est *d'identifier* des modèles interprétables (peu de termes) et généralisables (robuste) pour traduire la physique du système. On peut citer les efforts dans la modélisation de fluides complexes tels que le sang (Bessonov *et al.*, 2016) et les efforts pour la modélisation de systèmes complexes comme le climat où la deuxième loi de Newton n'est pas applicable (Brunton *et al.*, 2016b).

La **fermeture**. C'est une application spécifique de la modélisation. Lorsque l'écoulement est turbulent, les ressources nécessaires pour une résolution directe des équations de Navier Stokes sont importantes. Deux alternatives de calcul existent : la simulation des grandes échelles (LES) et la résolution des équations moyennées (RANS). Dans les deux cas, les équations à résoudre sont *ouvertes* et il est nécessaire de modéliser toute ou partie des échelles. Pour la LES, il s'agit de modéliser les échelles sous maille non résolues par le maillage tandis que pour un calcul RANS, il s'agit de modéliser l'action de *toutes* les fluctuations sur le champ moyen (Duraisamy *et al.*, 2019; Gonzalez et Balajewicz, 2018).

La **réduction**. Le nombre de degrés de libertés du système peut être élevé. Pour des écoulements turbulents, les non linéarités imposent de regarder l'écoulement aux plus petits échelles, augmentant drastiquement la dimension du système. La réduction cherche à identifier les degrés de libertés pertinents (avec parcimonie) pour fournir une description basse dimension de l'écoulement (Rowley et Dawson, 2017). Parmi les résultats fondateurs, on peut citer la dynamique de l'écoulement laminaire autour d'un cylindre qui peut se réduire à la dynamique de trois modes bien choisis (Noack *et al.*, 2003). Pour des écoulements turbulents, l'utilisation d'auto-encodeurs est prometteuse (Xu et Duraisamy, 2020).

Le **contrôle**. L'objectif est de manipuler l'écoulement pour en tirer profit, par exemple minimiser la traînée d'un véhicule. Les dispositifs de contrôle peuvent être passifs ou actifs. Pour la première catégorie, on peut citer des exemples comme les générateurs de vortex (Fouatih *et al.*, 2016), les surfaces imitant la peau des requins (Lang *et al.*, 2014) ou les winglets (Mattos *et al.*, 2003). Pour la seconde

catégorie, on peut s'intéresser à l'injection d'énergie par jet dans une couche limite afin de limiter les décollements (Ott *et al.*, 2019). L'enjeu consiste alors à déterminer les lois de contrôle optimales pour interagir avec l'environnement.

Ces quatre tâches ne s'excluent pas mutuellement et s'écrivent sous la forme d'un problème d'optimisation non linéaire et non convexe. Ces difficultés proviennent de la physique de l'écoulement, faisant intervenir une composante de transport non linéaire à l'origine d'interactions complexes en temps et en espace.

1.1.2 | La physique complexe d'un écoulement

Pour comprendre ces interactions, on se place dans le cas d'un écoulement incompressible de fluide newtonien et sans forces de volumes. La deuxième loi de Newton permet de dériver une équation pour la quantité de mouvement. En ajoutant les équations de conservation de la masse, on obtient un système d'équations aux dérivées partielles (Pope, 2001) :

$$\begin{cases} \boldsymbol{\nabla}.\boldsymbol{u} = 0 \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}.\boldsymbol{\nabla})\boldsymbol{u} = -\boldsymbol{\nabla} p + \dfrac{1}{\mathrm{Re}}\ \boldsymbol{\nabla}^2\boldsymbol{u} \end{cases}$$

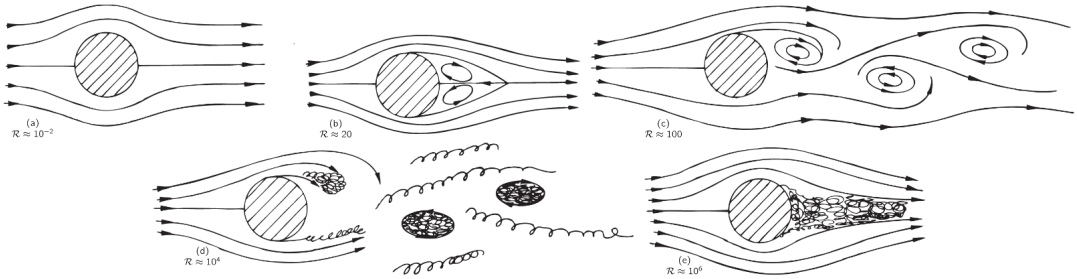
où le champ de vitesse est noté $\boldsymbol{u}$ et le champ de pression est noté p. La forme adimensionnelle de ces équations fait apparaître le nombre de Reynolds Re défini par :

$$\boxed{\mathrm{Re} = \frac{UD}{\nu}}$$

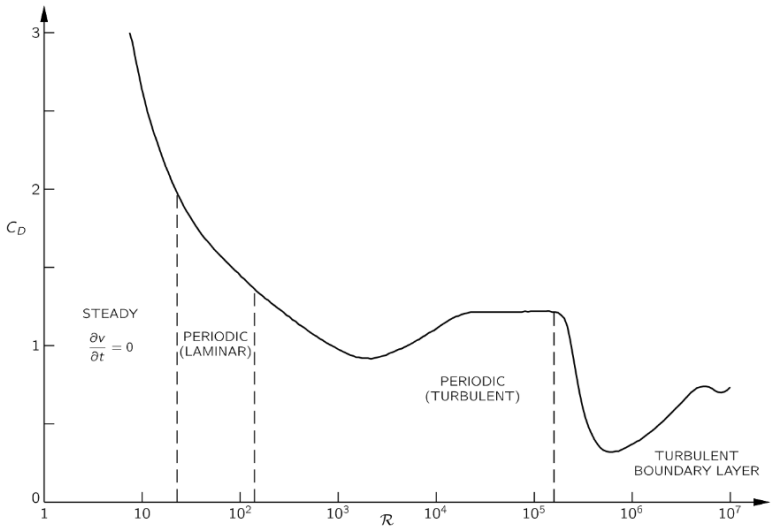
avec U une vitesse caractéristique [1], D une dimension caractéristique [2] et ν la viscosité moléculaire cinématique du fluide. Le nombre de Reynolds évalue le rapport entre les effets d'inertie (terme rouge de l'équation, dit *d'advection*) et le terme de diffusion (terme bleu). C'est un paramètre *bifurcatif* qui pilote l'importance des non linéarités dans l'écoulement (Sreenivasan *et al.*, 1987). L'écoulement derrière un cylindre est particulièrement parlant : la figure 1.1a est reprise d'un cours de Feynman *et al.* (1965) qui illustre l'influence du nombre de Reynolds sur la topologie de l'écoulement. En particulier, plus le nombre de Reynolds est élevé et plus l'écoulement est complexe ou *turbulent*. Les différences de topologie ont un effet sur le coefficient de traînée, dont l'évolution est montrée sur la figure 1.1b.

Lorsque le terme d'advection l'emporte sur le terme de diffusion, l'écoulement est donc difficile à caractériser : les gradients de vitesse [3] sont à l'origine d'une hiérarchie d'échelles, appelées *fluctuations*, qui interagissent entre elles. La figure 1.2a illustre

1. Par exemple la vitesse loin d'une paroi ou la vitesse à l'infini amont.
2. Typiquement celle d'un obstacle, comme le diamètre d'un cylindre.
3. Par exemple imposés par une paroi via la condition d'adhérence ou un obstacle placé dans l'écoulement



(a) Topologie de l'écoulement derrière un cylindre 2D. (a) écoulement rampant à Re = 0.01. (b) Écoulement symétrique à très faible nombre de Reynolds Re = 20. (c) Allée de Von Karman laminaire à Re = 100. (d) Allée de Von Karman turbulente à Re = 10^4. (e) Écoulement pleinement turbulent à Re = 10^6.



(b) Évolution du coefficient de traînée derrière un cylindre en fonction du nombre de Reynolds. Selon le régime d'écoulement, le coefficient de traînée n'a pas la même évolution. De gauche à droite : stationnaire - périodique laminaire - périodique turbulent - turbulent.

Figure *1.1* – *Visualisations de l'écoulement derrière un cylindre 2D (tirés du cours de physique de Feynman et al. (1965), volume 2, chapitre 41)*

le principe de *cascade énergétique* : l'écoulement moyen produit de grosses structures instables qui, par un processus de rupture, génèrent des harmoniques [4]. Les plus

4. Pour s'en convaincre, on peut considérer l'équation de Burgers $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$ et la condition initiale $u(x,0) = A\cos(kx)$. La solution à l'instant δt s'exprime sous la forme de la série de Taylor $u(x,\delta t) = A\cos(kx) + \delta t\frac{A^2k}{2}\sin(2kx) + ...$ qui fait apparaître des harmoniques du nombre d'onde

petites structures créées, dites de Kolmogorov, se dissipent par action de la viscosité moléculaire. Tant que l'écoulement moyen existe, la turbulence s'entretient : les structures restent organisées en augmentant l'entropie de l'environnement plutôt que la leur, évoluant ainsi loin de l'équilibre thermodynamique. Ce processus a été décrit qualitativement par Richardson (1922) et repris par Kolmogorov (1941). Le mathématicien a publié quatre articles pour caractériser le spectre énergétique (voir figure 1.2b). En particulier, il précise les quantités dimensionnantes dans chacune des régions (production, transfert, dissipation) et une analyse dimensionnelle permet d'écrire dans la région inertielle :

$$E(k) = C_K \epsilon^{\frac{2}{3}} k^{-\frac{5}{3}}$$

avec C_K une constante, k l'énergie cinétique contenue dans les fluctuations, ϵ le taux de dissipation de cette énergie et $E(k)$ la densité de k. Cette relation fait apparaître l'exposant $-5/3$, un résultat important en turbulence mais obtenue à partir de considérations expérimentales et dimensionnelles (Vassilicos, 2015). Du point de vue analytique, les avancées sont restées maigres et la preuve d'existence d'une solution régulière aux équations de Navier Stokes est devenue l'un des sept problèmes du millénaire (Lemarié-Rieusset, 2018). Si manipuler ces équations s'est avéré fructueux pour certaines missions (notamment la mécanique des fluides numérique [5]), cela reste illusoire pour d'autres (comme la prédiction temps réelle).

1.1.3 | Le Big Data

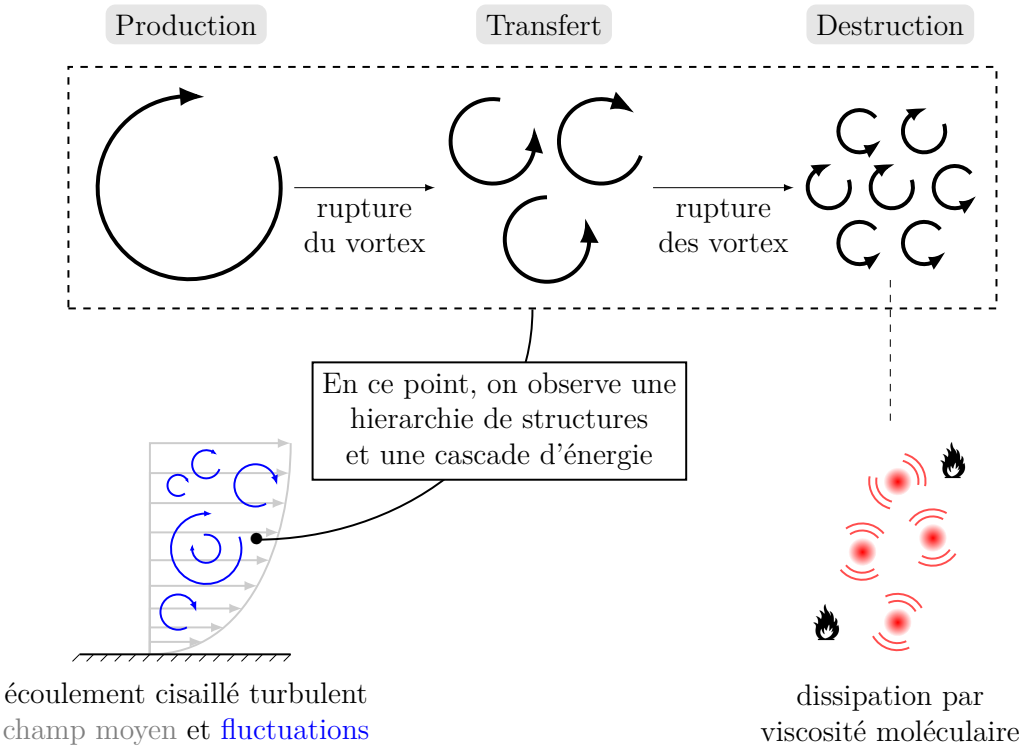
La mécanique des fluides n'échappe pas à la tendance actuelle du *Big Data*. Cette réalité est issue des nombreuses campagnes expérimentales et numériques de ces dernière années (Pollard *et al.*, 2017). Face à la richesse des données, une transition se dessine : plutôt que de construire des modèles à partir de la deuxième loi de Newton, les modèles sont construits à partir de données. C'est ainsi que les techniques récentes ont permis le post traitement et la compression efficace d'écoulements turbulents (Perlman *et al.*, 2007; Schmid, 2011), ouvrant la voie à la **mécanique des fluides orientée données**. Ce manuscrit s'inscrit dans cette philosophie, en explorant certaines des méthodes actuelles pour la **caractérisation** et la **modélisation** purement orientée données d'un écoulement.

1.2 | La mécanique des fluides orientée données

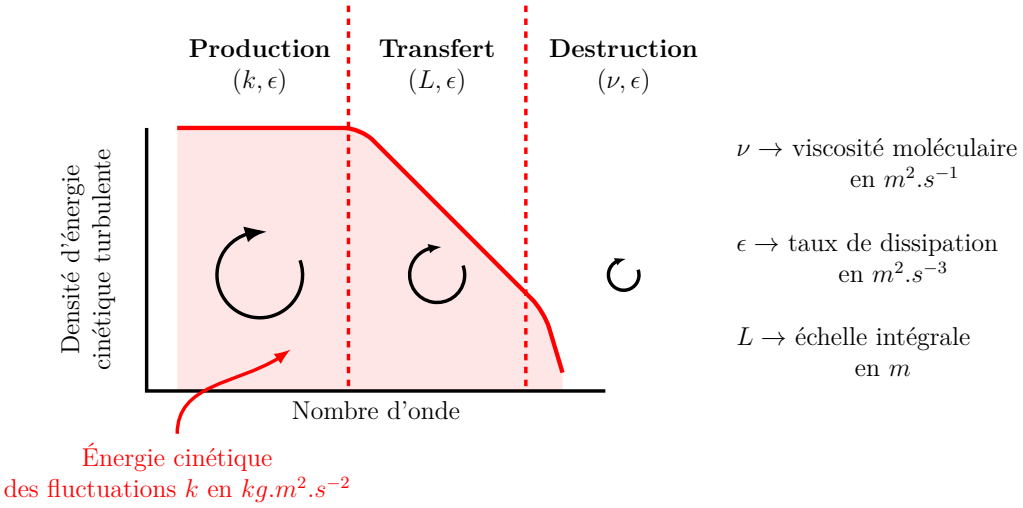
Cette section donne une présentation générale de l'apprentissage automatique appliquée à la mécanique des fluides. En particulier, on précise le contexte historique, les techniques d'extraction de structures propres (réduction) et les méthodes d'identification de modèles réduits (modélisation).

de forçage (Bailly et Comte-Bellot, 2015).

5. Par exemple pour dériver les équations de transports de k et ϵ pour un calcul RANS $k-\epsilon$. Les constantes du modèles restent toutefois empiriques. Il y a aussi tout le travail pour la simulation numérique directe et la simulation des grandes échelles.



(a) Concept de la cascade turbulente. Les grosses structures produisent l'énergie cinétique turbulente à partir du cisaillement de l'écoulement moyen. Ces structures sont instables et donnent naissance à des structures de plus en plus petites, jusque l'échelle de Kolmogorov où l'énergie est dissipée.



(b) Spectre de l'énergie cinétique turbulente observé en un point d'un écoulement turbulent. Les quantités entre parenthèses indiquent les dépendances des structures lors de chaque étape de la cascade.

Figure *1.2* – *Une visualisation de la cascade énergétique pour un écoulement turbulent.*

1.2.1 | L'apprentissage automatique

L'apprentissage automatique ou *machine learning* est un ensemble d'outils pour l'optimisation basée sur les données. On distingue trois catégories d'apprentissage (Brunton *et al.*, 2020) :

- L'apprentissage **supervisé** : des données étiquetées sont disponibles. L'objectif est d'apprendre la relation entre l'entrée et la sortie. Lorsque la sortie est continue, on parle de *régression*. Pour une sortie discrète, on parle de *classification*.
- L'apprentissage **non supervisé** : on cherche les caractéristiques de données non étiquetées, notamment pour réduire leur dimension.
- L'apprentissage **semi-supervisé** : la supervision des données est partielle. Les algorithmes sont surtout destinés à l'apprentissage de réflexes et à la génération de nouvelles données.

La figure 1.3 donne une vision générale des algorithmes et illustre graphiquement la différence entre l'apprentissage supervisé et l'apprentissage non supervisé. Plusieurs des algorithmes seront étudiés en détail dans ce manuscrit.

1.2.2 | Contexte historique

Les prémisses de l'apprentissage automatique sont nées dans les années 1950, avec deux développements distincts : d'un côté la cybernétique [6] avec le concept de rétroaction (Wiener, 1948) et de l'autre l'émulation du cerveau humain avec le développement du perceptron (Rosenblatt, 1958). Rosenblatt introduit la notion de neurone formel qui sera à la base des réseaux de neurones actuels. Toutefois, l'excitation à l'époque est de courte durée : avec une couche de neurones, le classifieur binaire de Rosenblatt est seulement capable d'apprendre des séparatrices linéaires, échouant donc sur le problème XOR [7] (Minsky et Papert, 1969). L'engouement pour cette nouvelle technologie qui souffre des limitations matérielles de l'époque s'effondre au début des années 60, une période qualifiée de premier hiver de l'intelligence artificielle.

Parallèlement, deux étudiants à Berlin développent un algorithme d'optimisation de profils aérodynamiques (Rechenberg, 1964). Il s'agit du premier exemple d'apprentissage automatique en mécanique des fluides expérimentales. Les étudiants cherchent à diminuer la traînée d'une plaque multi-panneaux en jouant sur les angles entre chaque panneau (voir figure 1.4). L'optimisation inclut des aspects stochastiques, les nombres aléatoires étant générés par une planche de Galton. Cette innovation [8] sera à la base de nombreux algorithmes d'optimisation

6. Science qui étudie les mécanismes d'informations pour des systèmes complexes.

7. On considère deux réels x et y. Si x et y sont de signes opposés, la fonction retourne 1. Si x et y sont de mêmes signes, la fonction retourne 0. Dans le plan, il n'est pas possible de séparer linéairement les classes 0 et 1. Il faudra attendre le développement des machines à vecteur supports avec noyau ou les ressources matérielles pour l'implémentation des perceptrons à plusieurs couches.

8. Ajouter de l'aléatoire dans le processus d'optimisation

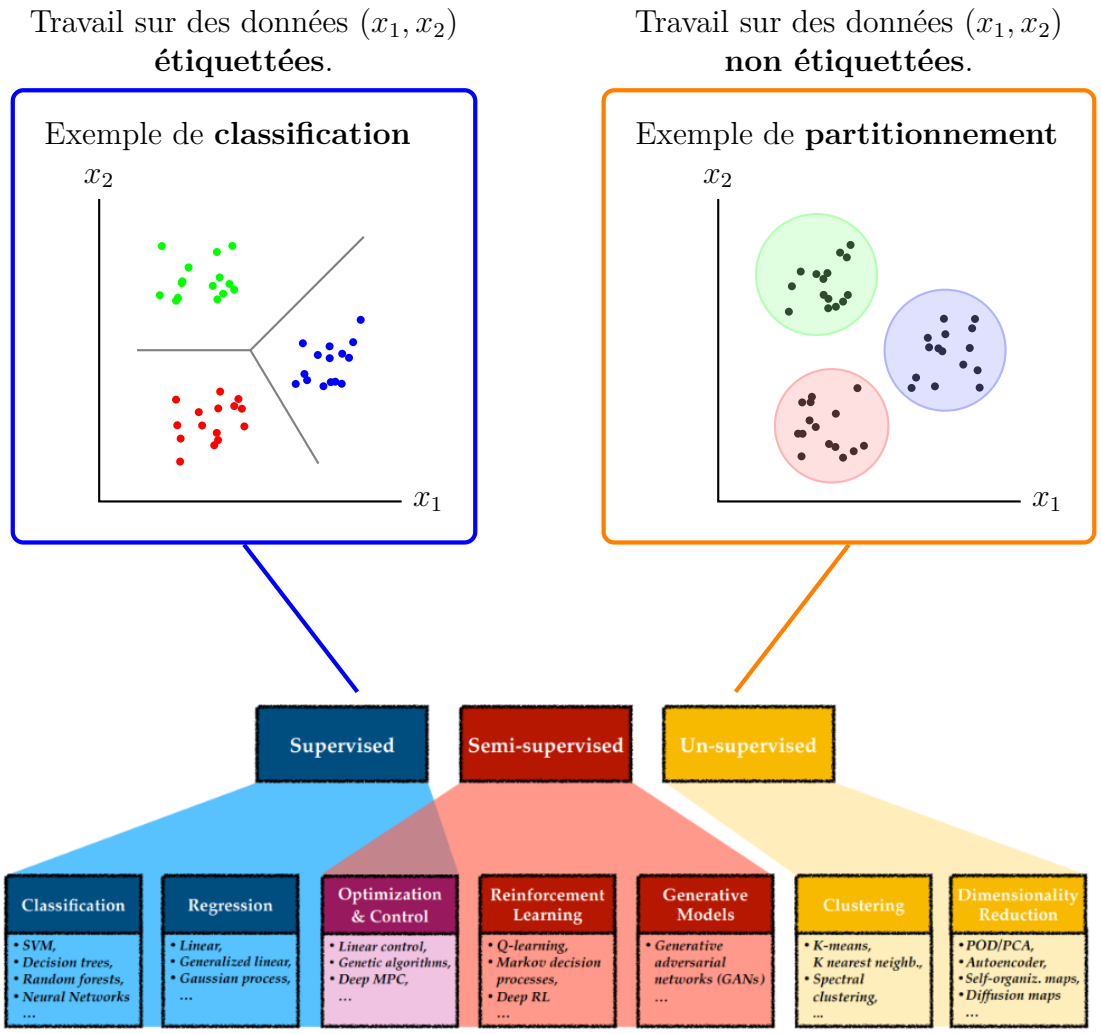


Figure *1.3* – *Vue générale de l'apprentissage automatique et algorithmes principalement utilisés. Sur l'exemple de classification, l'objectif est de déterminer une frontière de décision pour déterminer la classe (ici la couleur) d'un nouveau point. Sur l'exemple de partitionnement, l'objectif est de séparer les données. Une partie de la figure est tirée de Brunton et al. (2020).*

(Bottou, 2012) et stratégies d'évolution (Igel *et al.*, 2007).

Un second hiver de l'intelligence artificielle survient dès 1973, avec le rapport de Lighthill (1973). Selon l'auteur, beaucoup de promesses ont été faites par les chercheurs mais peu de résultats ont été obtenus. Il s'appuie notamment sur les

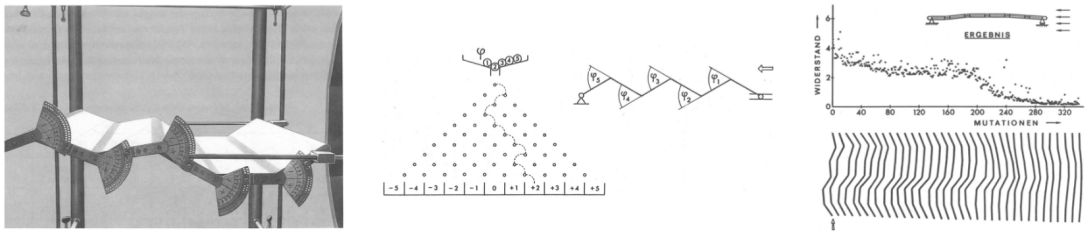


Figure *1.4* – *Optimisation d'une plaque multi-panneaux par une stratégie d'évolution. Le dispositif expérimental est présenté à gauche. La planche de Galton est utilisée pour générer de nouvelles configurations. La configuration obtenue après plusieurs générations n'est pas parfaitement plane et minimise la traînée. Figure tirée de Brunton* et al. *(2020).*

échecs en traitement automatique du langage [9]. Un regain d'intérêt se fait dans les années 80, avec principalement le développement de l'algorithme de rétro-propagation (Rumelhart *et al.*, 1986) pour l'apprentissage de réseaux connexionnistes. Les liens avec la mécanique des fluides deviennent plus forts : les réseaux de neurones sont reliés à la décomposition en modes propres orthogonaux (Baldi et Hornik, 1989) ; ils sont également utilisés pour l'analyse de trajectoires (Teo *et al.*, 1991), l'identification de phases dans des écoulements multi-phasiques (Bishop et James, 1993), l'interpolation de résultats expérimentaux (Ncrgaard *et al.*, 1997) et la modélisation de la turbulence en proche paroi (Milano et Koumoutsakos, 2002). Ce dernier exemple est particulièrement important car il est le premier à utiliser un auto-encodeur profond pour la compression de données en mécanique des fluides.

Les motivations pour appliquer l'apprentissage automatique à la mécanique des fluides se trouvent également dans la nature. De nombreux organismes sont capables de réagir à des environnement instationnaires par des manœuvres ou des déformations complexes. Pourtant, ces organismes n'ont pas connaissance des équations de Navier-Stokes ni du champ de vitesse tri-dimensionnel ; ils ne disposent que de données partielles. On peut citer deux exemples : l'échappée de larves de poissons à une poche de vorticité (Gazzola *et al.*, 2012) et l'exploitation des tourbillons par des poissons pour une nage efficace (Liao *et al.*, 2003). L'apprentissage de ces réflexes est formalisé dans l'apprentissage par *renforcement* où l'on découverte et renforce des lois de contrôles à partir d'expériences (Verma *et al.*, 2018; Paris *et al.*, 2021; Reddy *et al.*, 2018).

Dans ce qui suit, on se concentre plus spécifiquement sur les avancées en matière de réduction de dimension, de modélisation, de fermeture et de contrôle.

9. Fait amusant : quarante ans plus tard, on peut retrouver son discours automatiquement traduit par un ordinateur.

1.2.3 | L'extraction de structures propres

Même pour des systèmes complexes, il existe des structures dominantes de rang faible (Lumley, 1967). Sirovich *et al.* ont publié en 1987 deux articles allant dans ce sens. Le premier concerne les "visages propres" : à partir de l'analyse en composantes principales (PCA) d'une bibliothèque de visages, les auteurs ont déterminé les structures principales présentes dans un visage humain (Sirovich et Kirby, 1987). Le second article, en trois parties, utilise la même méthode sur un écoulement fluide, renommée décomposition en modes propres orthogonaux (POD) (Sirovich, 1987). En sortie de l'algorithme, le film de l'écoulement est réécrit comme une combinaison linéaire de modes (directions spatiales) classés hiérarchiquement par la variance des données qu'ils restituent. En ne conservant que les premiers modes, on dispose ainsi d'une vision *réduite* des instantannés avec l'information temporelle d'un côté et l'information spatiale de l'autre. La figure 1.5 illustre les résultats d'une telle décomposition pour l'écoulement autour d'un profil NACA0012. Les modes 1 et 2 puis 3 et 4 sont semblables ; ils n'apparaissent que décalés dans la direction d'advection. Ce fonctionnement par paire est symptomatique de structures advectées, que la POD (basée sur une décomposition en valeurs singulières) peut difficilement capturer [10] (Taira *et al.*, 2017).

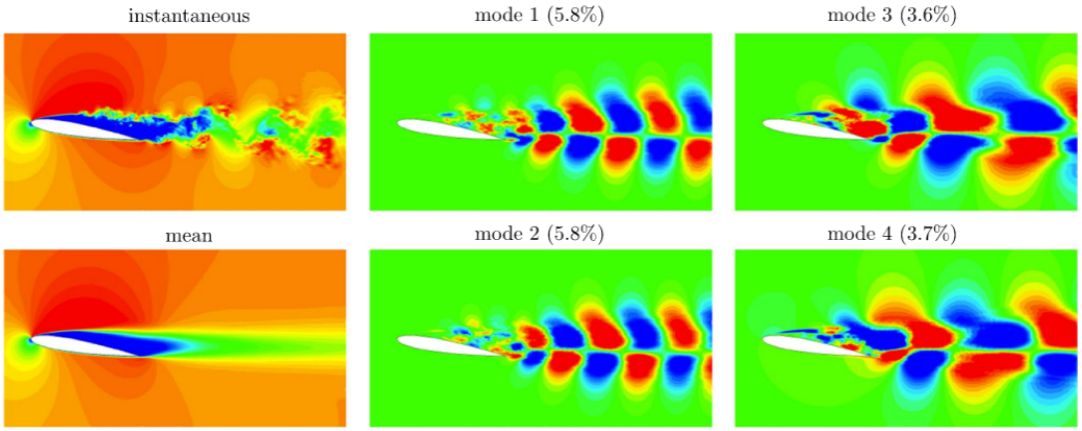


Figure *1.5* – *Décomposition en modes propres orthogonaux de l'écoulement turbulent autour d'un profil NACA0012 à incidence $\alpha = 9°$. Le champ instantané est écrit comme la somme du champ moyen et de modes orthogonaux classés par énergie restituée. En considérant les quatre premiers modes, on restitue approximativement* 20% *de l'énergie cinétique. Tirée de Taira* et al. *(2017).*

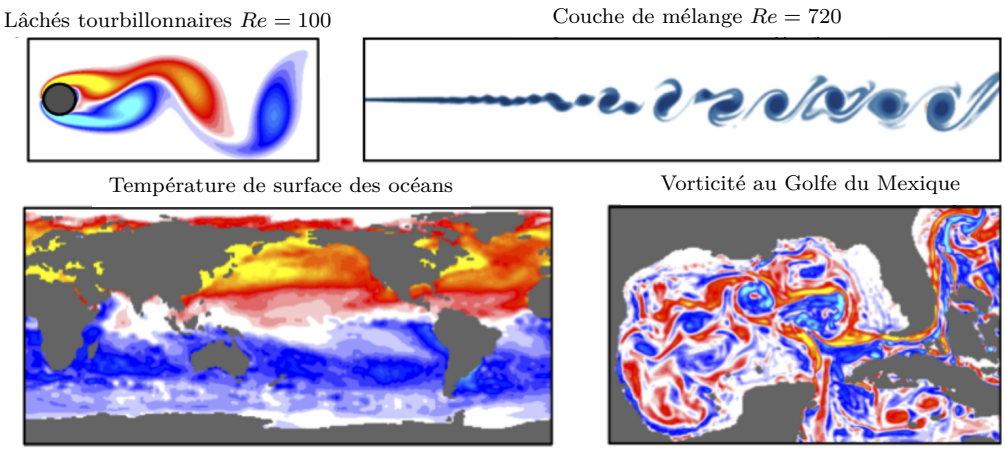
10. On le comprend avec une onde simple : $\sin(x - ct)$ qui se décompose selon $\sin(x)\cos(ct) - \sin(ct)\cos(x)$. On fait apparaître les modes $\sin(x)$ et $\cos(x)$ qui sont la même structure déphasée de $\pi/2$.

La POD est ainsi devenue un standard pour la réduction de bases de données, principalement pour la simplicité de son implémentation, l'orthogonalité des modes, l'optimalité en termes énergétiques et la linéarité de la décomposition [11]. Toutefois, certaines limitations peuvent être soulevées. Premièrement, la POD est uniquement basée sur les corrélations d'ordre deux, ignorant ainsi les moments d'ordre plus élevé. Ce problème peut-être résolu avec des alternatives à l'analyse en composantes principales, comme l'analyse en composantes indépendantes (Shlens, 2014). Deuxièment, les coefficients temporels ont une dynamique faisant apparaître un mélange de fréquences (George, 1988). C'est dans ce contexte que la POD spectrale (SPOD) fut développée, pour exhiber des modes oscillants à une seule fréquence (Towne *et al.*, 2018). Troisièment, les directions principales sont rangées par importance énergétique et non par importance dynamique. La POD équilibrée (BPOD) est une alternative qui détermine deux ensembles de modes (dits équilibrés et adjoints) classés par contrôlabilité/observabilité. De cette manière, les perturbations peu énergétiques mais très observables sont capturées (Rowley, 2005). Quatrièmement, il n'existe pas de règle universelle concernant le nombre de modes POD à conserver. S'il existe bien des astuces mathématiques (Gavish et Donoho, 2014), la sélection est souvent heuristique, basée sur un pourcentage de l'énergie cinétique à restituer. Toutefois, plus l'écoulement est complexe (myriade d'échelles) et plus le nombre de modes à considérer pour restituer une partie de la variance est important (voir figure 1.6 et Callaham *et al.* (2019)). Enfin, la décomposition POD est sensible aux données corrompues par le bruit. C'est en ce sens que la POD robuste (RPOD) a été développée, permettant de détecter les artefacts dans les clichés instantanés (Candès *et al.*, 2011). La figure 1.7 illustre la stratégie sur un cylindre 2D à bas nombre de Reynolds où du bruit poivre-sel a été ajouté aux clichés instantanés (Scherl *et al.*, 2020). Une telle décomposition est prometteuse pour des campagnes expérimentales avec de la vélocimétrie par images de particules haute résolution et haute cadence. Dans le même état d'esprit, il existe aussi la méthode Gappy POD (GPOD) qui fonctionne sur des données masquées [12] (Everson et Sirovich, 1995) et s'applique bien aux données expérimentales (Murray et Ukeiley, 2007).

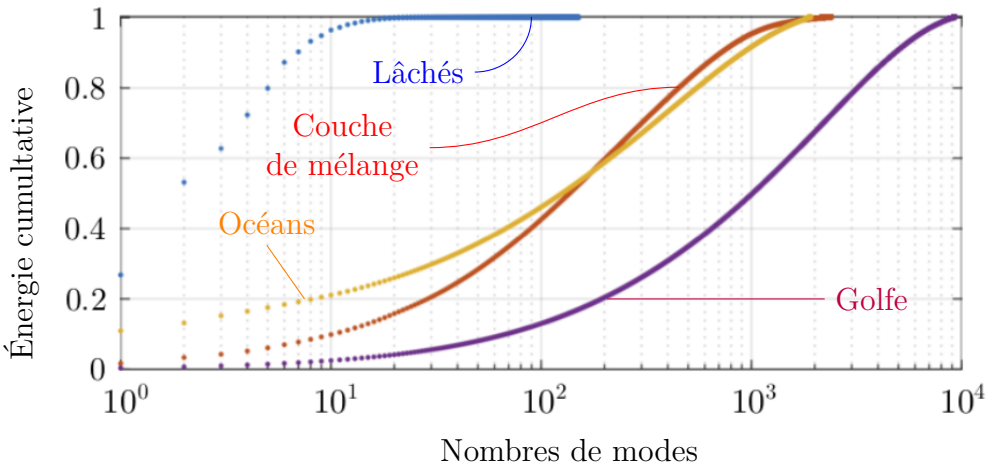
Au delà de l'analyse en composantes principales et ses extensions directes, d'autres techniques de réduction ont été développées. On peut citer les méthodes de partitionnement (Kaiser *et al.*, 2014) et d'apprentissage de variétés (Tauro *et al.*, 2014) pour la classification d'écoulements. Il existe aussi des stratégies pour les problèmes d'ondes progressives (Mendible *et al.*, 2020). L'une des avancées majeures concerne l'utilisation des réseaux de neurones. Depuis 1989, on sait que l'analyse en composantes principales est équivalente à un réseau de neurones linéaire avec une seule couche cachée (Baldi et Hornik, 1989). En complexifiant l'architecture du réseau (nombre de couches, nombre de neurones, activations), il est possible d'auto-encoder le champ de vitesse et d'apprendre une variété de dimension (très) réduite sur laquelle évolue l'état. On parle d'état *latent* car il

11. La superposition assure que les modes calculés respectent les contraintes linéaires telles que l'incompressibilité et la condition d'adhérence. La propriété d'orthogonalité facilite les projections pour la dérivation d'un modèle réduit intrusif.

12. Typiquement des pixels morts



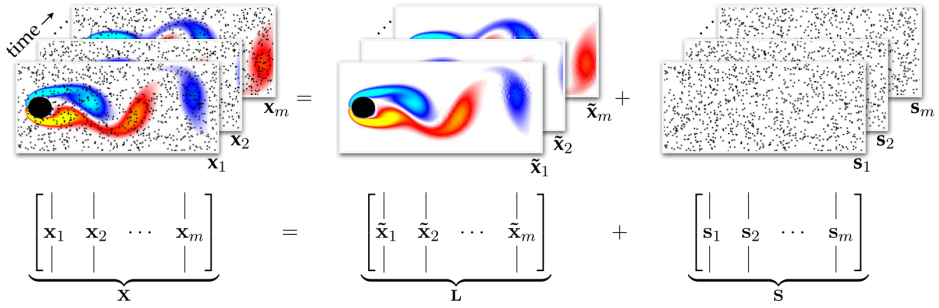
(a) Les quatre configurations étudiées dans Callaham *et al.* (2019).



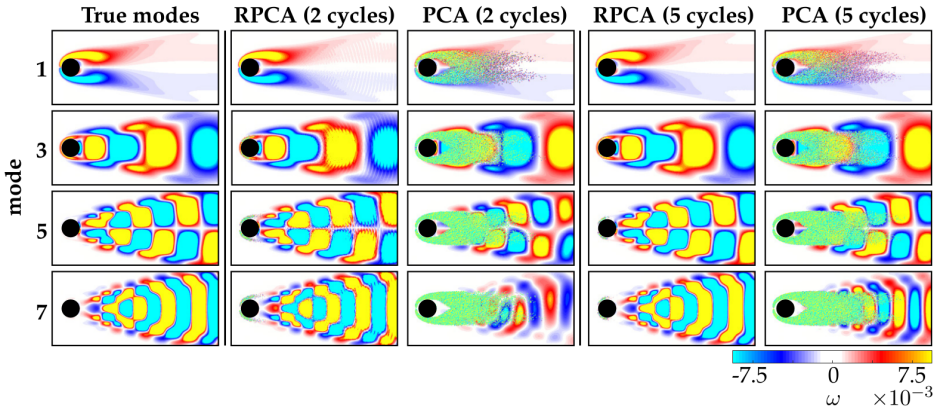
(b) Nombre de modes POD requis pour restituer un pourcentage de la variance. Par exemple, il faut considérer 1000 modes pour restituer 50% de la variabilité des données sur la vorticité dans le Golfe du Mexique.

Figure *1.6* – *Variance des données restituée en fonction du nombres de modes. Plus l'écoulement est complexe, plus le nombre de modes requis est important pour assurer une stationnarité statistique et restituer près de* 100% *de l'énergie. Tiré de Callaham* et al. *(2019))*

s'agit d'une variable cachée, obtenue par des transformations non linéaires d'autant plus complexes que le réseau l'est. La première application d'un tel réseau



(a) Principe de la POD robuste. Les clichés sont décomposés en la somme d'une matrice L (de faible rang, contenant l'information cohérente) et d'une matrice S (parcimonieuse, correspondant au bruit). La POD est ensuite effectuée sur la matrice L.



(b) Résultats obtenus pour des clichés corrompus (1% des signaux de vitesse est bruité). La POD robuste identifie bien mieux les modes attendus que la POD classique. Le nombre de cycles correspond au nombre de périodes de lâchés observés dans la matrice X.

Figure *1.7* – *Comparaison de la RPOD et de la POD sur l'écoulement autour d'un cylindre 2D à* $Re = 100$. *Tiré de Scherl et al. (2020).*

remonte à 2002 (Milano et Koumoutsakos, 2002) et les applications plus récentes font intervenir des structures récurrentes (Gonzalez et Balajewicz, 2018) ou convolutives (Xu et Duraisamy, 2020) avec respect des lois de conservation (Lee et Carlberg, 2019).

Finalement, il existe de nombreuses approches pour réduire une base de données. Plutôt que de travailler avec l'instantané de dimension importante (nombre de points de maillage), on travaille sur un état réduit évoluant selon des directions latentes. La dynamique de l'écoulement est alors réduite à la dynamique de l'état latent et un modèle dynamique de cet état constitue un *modèle réduit* (*Reduced Order Model* ou ROM) de l'écoulement. Dans la suite, on s'intéresse plus

spécifiquement à cette problématique de modélisation.

Remarque : dans ce manuscrit, on fait le choix de se concentrer sur des réductions dont le critère est basé sur les moindres carrés. Dans des approches plus systémiques, des normes différentes [13] peuvent être utilisées (Vuillemin *et al.*, 2013).

1.2.4 | L'identification de modèles réduits

Pour illustrer la stratégie d'identification de modèles réduits, on commence par un exemple canonique : l'écoulement derrière un cylindre 2D à faible nombre de Reynolds. En 1987, il est vérifié expérimentalement que cet écoulement subit une bifurcation supercritique de Hopf [14] au nombre de Reynolds critique $Re_c = 47$ (Zebib, 1987). L'écoulement devient périodique et on observe des lâchés tourbillonnaires s'organisant en une allée de Karman laminaire. Cette observation créée une sorte de paradoxe : la forme normale de la bifurcation de Hopf [15] requiert un terme cubique dans l'équation de la dynamique mais les équations de Navier Stokes ont des non linéarités au plus quadratiques. Il a fallu une quinzaine d'années pour trouver la solution et identifier cette bifurcation (Noack *et al.*, 2003). Les auteurs montrent qu'avec les bonnes mesures du système (deux modes issus d'une décomposition orthogonale et un mode de transition), on peut avoir une description de la dynamique sous la forme de trois équations différentielles ordinaires couplées [16]. Obtenues de manière analytique par une approche *intrusive* [17], ces équations seront identifiées en 2016 avec une méthode purement orientées données appelée SINDy pour *Sparse Identification of Nonlinear Dynamics* (Brunton *et al.*, 2016b). La figure 1.8 illustre la procédure de cette régression dite *symbolique*. Depuis, de nombreuses extensions ont été proposées. Parmi les plus innovantes, on peut citer le travail de Loiseau *et. al* qui ajoutent des contraintes physiques (conservation de l'énergie) et incluent des non linéarités d'ordre plus élevé pour récupérer l'énergie perdue avec une troncature aux premiers modes (Loiseau et Brunton, 2018b). La stratégie est testée sur plusieurs configurations, telles que l'écoulement de cavité et un écoulement qualifié de pinball (Loiseau *et al.*, 2018). On peut aussi citer le travail de Rudy *et al.* (2017) qui étendent la méthode à l'identification d'équations aux dérivées partielles plutôt

13. En l'occurrence $\mathcal{H}_2$ et $\mathcal{H}_\infty$. La première correspond à l'énergie en sortie d'un système lorsque l'on injecte un Dirac. La seconde mesure le gain maximal de la réponse fréquentielle du système.

14. Un point fixe stable devient instable et un cycle limite stable apparaît.

15. Si z est la variable complexe $x + iy$ où (x, y) est l'état du système, la forme normale d'une bifurcation de Hopf est $\frac{dz}{dt} = z(\lambda + i + b|z|^2)$ où λ est un paramètre et $b = \alpha + i\beta$ est un complexe. Une bifurcation super-critique est obtenue pour $\alpha < 0$ et $\lambda > 0$

16. La solution du paradoxe résidait donc dans la séparation des échelles. Les équations de Navier Stokes sont des équations aux dérivées partielles *non linéaires* et pas des équations aux dérivées ordinaires. Le terme quadratique de l'EDP autorise bien la bifurcation de Hopf mais pour s'en rendre compte, il fallait projeter les équations sur le sous espace généré par les deux premiers modes POD et un mode corrigé par le champ moyen, appelé *shift mode*.

17. Le modèle réduit est obtenu par projection des équations de Navier Stokes sur le sous espace généré par les trois modes. Un modèle réduit *non intrusif* serait uniquement construit à partir de données.

que de simples équations différentielles ordinaires. Le travail de Champion *et al.* (2019) est également innovant : l'auteure apprend simultanément l'opération d'auto-encodage **et** la dynamique de l'état latent par SINDy.

Idéalement, on souhaite identifier un modèle dynamique linéaire pour l'interpréter avec une décomposition spectrale. Toutefois, comme pour l'écoulement derrière le cylindre à bas nombre de Reynolds, la dynamique de l'état réduit est très souvent non linéaire. Une des clés pour interpréter la dynamique est l'opérateur de Koopman (voir (Bagheri, 2013) pour un exemple pertinent). Introduit en 1931, cet opérateur permet de faire évoluer linéairement toutes les observations de l'état [18] d'un système (Koopman, 1931). Il y a toutefois un échange de difficultés : on passe d'un système non linéaire mais de dimension finie à un système linéaire de dimension infinie. L'objectif consiste alors à déterminer une approximation finie de l'opérateur, en déterminant un sous ensemble d'observables invariant par l'opérateur de Koopman (Brunton *et al.*, 2016a). En choisissant comme observables les composantes de l'état latent, l'approximation de l'opérateur est connue sous le nom de *décomposition en modes dynamiques* (DMD) (Schmid, 2011). Il s'agit du meilleur [19] modèle linéaire "un pas de temps" pour décrire la dynamique des données (Rowley et Dawson, 2017; Taira *et al.*, 2017; Loiseau, 2020) et les applications pour le post-traitement d'écoulement de jets (Semeraro *et al.*, 2012), de cavité (Lusseyran *et al.*, 2011), de sillage (Tu *et al.*, 2011) et de couche limite (Lee *et al.*, 2012) sont nombreuses. La figure 1.9 illustre le fonctionnement de l'algorithme et les résultats obtenus sur l'écoulement derrière un cylindre 2D. En particulier, on observe que la DMD est sensible aux données corrompues et qu'une réduction robuste est nécessaire pour capter le cycle limite. Si maintenant on choisit comme observables des fonctions non linéaires des composantes de l'état, on parle de DMD étendue (EDMD) (Williams *et al.*, 2015). Tout l'enjeu consiste à bien choisir ces fonctions pour que la dynamique de l'état observé soit linéaire. Le travail de Lusch *et al.* (2018) est particulièrement intéressant : en utilisant l'expressivité des réseaux de neurones, les auteurs déterminent des transformations de l'état telle que la dynamique des nouvelles coordonnées est effectivement linéaire. Les transformations sont paramétrées par un réseau auxiliaire, permettant de couvrir les spectres continus (voir figure 1.10). Une autre approche consiste à utiliser des signaux retardés pour tirer parti du théorème de Takens (1981) [20]. La méthode, dénommée HAVOK pour *Hankel Alternative View Of Koopman* permet d'identifier un système linéaire forcé pour des systèmes chaotiques tel que Lorenz 1963 (Brunton *et al.*, 2017). Enfin, l'algorithme SINDy déjà introduit peut aussi être utilisé pour identifier de manière parcimonieuse des fonctions propres de l'opérateur de Koopman, avec de bons résultats sur des systèmes académiques (Kaiser *et al.*, 2017).

18. Fonctions à valeurs **réelles** sélectionnées dans un espace d'Hilbert, par exemple les fonctions de carrés intégrables.

19. Au sens des moindres carrés

20. Un système chaotique peut être reconstruit à partir d'une séquence d'observations de son état. Sur l'attracteur de Lorenz, l'histoire d'une seule composante permet de construire un attracteur topologiquement identique.

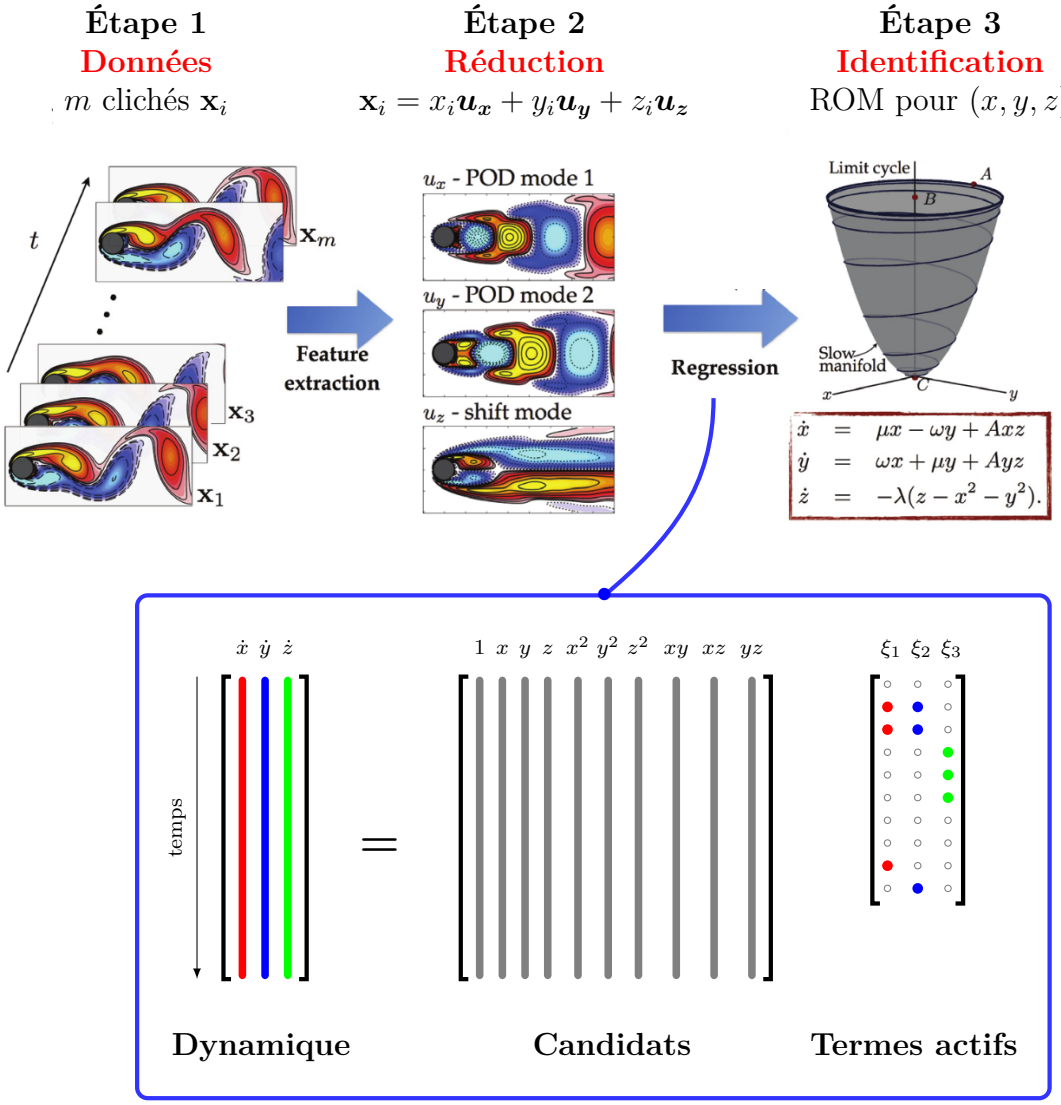


Figure *1.8* – *Modélisation du sillage d'un cylindre 2D à* $Re = 100$. *Les données sont d'abord collectées puis réduites selon trois directions spatiales. Une régression dans une bibliothèque de candidats permet d'identifier un modèle dynamique de la même forme que celui dérivé analytiquement par projection des équations de Navier Stokes (Noack et al. (2003)). L'état réduit étant* (x, y, z), *la variable* z *adhère rapidement à la variété parabolique définie par* $x^2 + y^2$ *depuis le point fixe instable (C) avant de se positionner sur le cycle limite. Vu dans l'espace physique, le point (A) correspond au champ de vitesse instantanné tandis que le point (B) correspond au champ de vitesse moyenne. Figure adaptée du livre Brunton et Kutz (2019) pour une méthode développée dans Brunton et al. (2016b)*

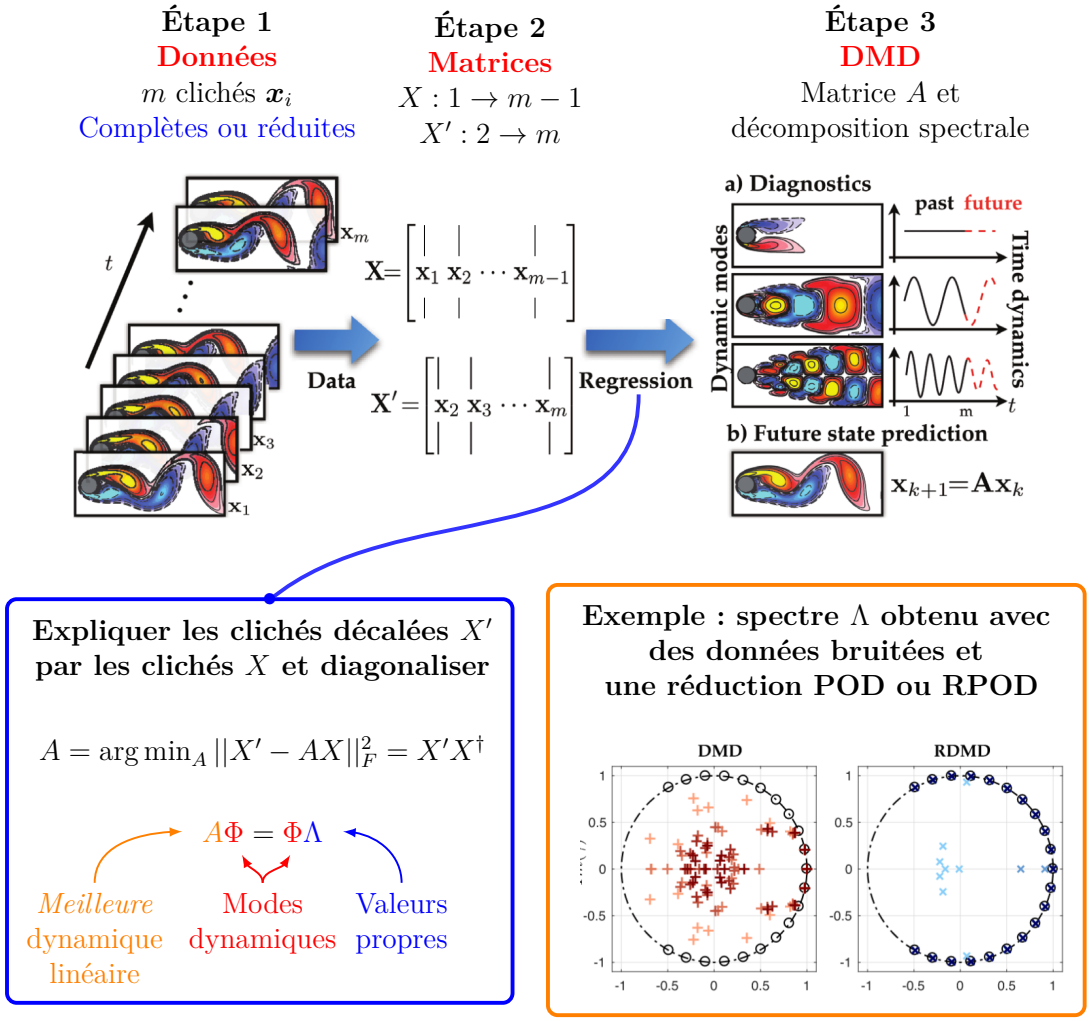
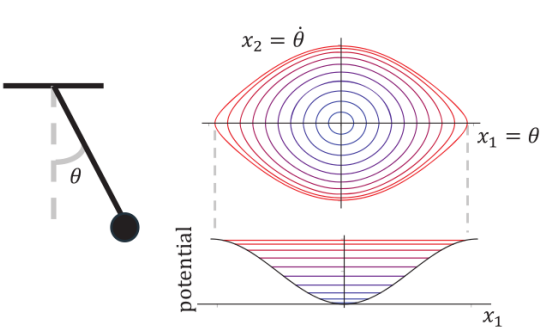
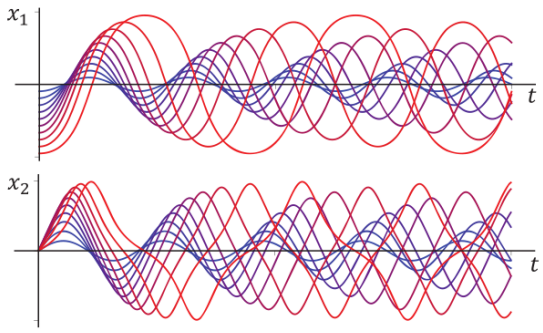


Figure *1.9* – *Illustration de la décomposition en modes dynamiques. Il s'agit d'une régression linéaire entre les clichés décalés dans le temps X' et les clichés X. Une décomposition spectrale de la matrice A renseigne sur la dynamique principale d'évolution en termes de modes Φ et d'évolution Λ. Pour l'écoulement périodique autour du cylindre, les valeurs propres doivent théoriquement se positionner sur le cercle unité (cercle noirs). Avec des données bruitées, il est nécessaire de faire une réduction robuste pour obtenir ce résulat et limiter les valeurs propres défectives. Figure reprenant des éléments de Brunton et Kutz (2019) et Scherl et al. (2020)*

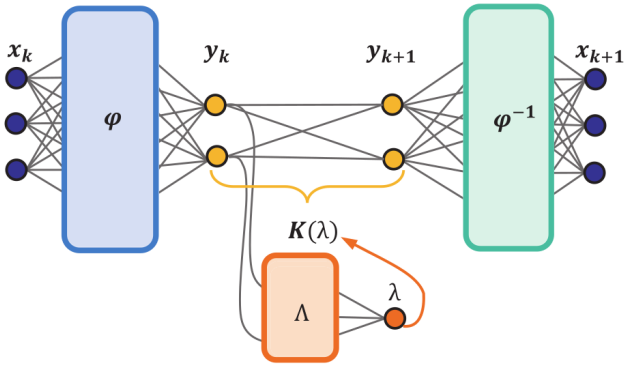
Toutes les méthodes citées permettent d'établir des modèles réduits déterministes interprétables, utilisables comme modèle de substitution pour de la prédiction et



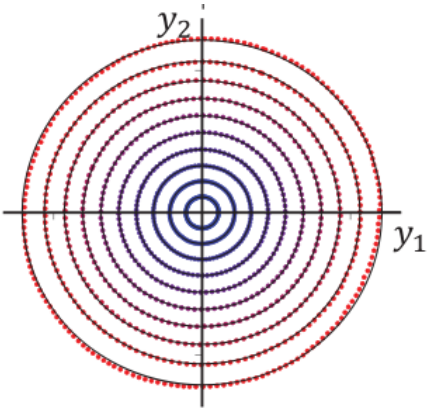
(a) Pendule d'équation $\ddot{\theta} = -\sin(\omega\theta)$



(b) La fréquence d'oscillation est différente selon l'amplitude initiale.



(c) Réseau de neurones paramétré pour déterminer des coordonnées intrinsèques où la dynamique est linéaire.



(d) Dynamique des fonctions propres identifiées.

Figure *1.10* – *Approximation finie de l'opérateur de Koopman par réseau de neurones. L'état complet est encodé (transformation φ), avancé linéairement (modèle $K(\lambda)$) puis décodé (transformation φ^{-1}). Les valeurs propres du modèle dynamique linéaire sont paramétrés par un réseau auxiliaire Λ. Sur l'exemple du pendule, un système à spectre continu, la dynamique des coordonnées intrinsèques est bien linéaire car les orbites sont parfaitement circulaires. Tiré de Lusch et al. (2018).*

du contrôle (Proctor *et al.*, 2016; Bieker *et al.*, 2019). La méthode CROM [21] mérite aussi d'être citée. Cette approche non supervisée permet d'approximer le dual de l'opérateur de Koopman, nommé opérateur de Perron Frobenius. Cet opérateur fait évoluer linéairement les densités de probabilité de l'état et son approximation finie prend la forme d'une chaîne de Markov (Kaiser *et al.*, 2014, 2018). Enfin, les

21. *Cluster-based Reduced Order Model*

méthodes de type boîte noire sont également utiles pour la prédiction de l'état latent, par exemple avec des réseaux de neurones récurrents à cellules de mémoire (Salehinejad *et al.*, 2017; Hochreiter et Schmidhuber, 1997). Nous avons utilisé ces réseaux pour apprendre un modèle dynamique de Lorenz 63 (Dubois *et al.*, 2020) et d'autres auteurs les ont utilisés en mécanique des fluides (Mohan et Gaitonde, 2018; Gonzalez et Balajewicz, 2018; Vlachas *et al.*, 2018).

1.2.5 | Les modèles de fermeture

Une application spécifique des techniques de modélisation concerne la fermeture en mécanique des fluides numérique. Plutôt que de travailler avec un maillage très fin capturant toutes les échelles (DNS), le maillage est relâché et seule une partie des échelles est résolue. L'effet des petites échelles est prise en compte via un modèle pour les tensions de Reynolds (si approche RANS) ou le tenseur sous maille (si approche LES). Dans l'ère du Big Data, ces modèles peuvent maintenant être appris à partir de données, comme largement développée par Duraisamy *et al.* (2019). Un exemple fondateur est présenté dans l'article de Ling *et al.* (2016) où les auteurs utilisent un réseau de neurones pour construire un modèle de viscosité turbulente. Le réseau est construit sur une base de dix tenseurs isotropes particuliers (Pope, 1975) afin de respecter les besoins d'invariance (voir figure 1.11). S'il s'agit du premier exemple de réseau avec contraintes physiques, de nombreuses autres études existent pour les fermetures RANS : utilisation de réseaux artificiels (Singh *et al.*, 2017), de forêts aléatoires (Kaandorp et Dwight, 2020; Wang *et al.*, 2017) ou de machines à vecteurs supports (Ling et Templeton, 2015). En ce qui concerne la LES, on peut citer l'aricle de Maulik *et al.* (2019) pour de la turbulence 2D et surtout l'article de Novati *et al.* (2021) : les auteurs utilisent de l'apprentissage par renforcement profond pour automatiser l'apprentissage de modèle de turbulence. Illustré sur un écoulement de turbulence isotrope, la promesse est faite qu'une telle approche semi-supervisée permettra d'apprendre des modèles de fermeture généralisables.

1.2.6 | Contrôle et renforcement

Dans les sections précédentes, nous avons précisé différents algorithmes parmi toutes les classes de l'apprentissage automatique. Ici, on s'intéresse spécifiquement à l'apprentissage par renforcement, une classe qui fait l'interface entre la théorie du contrôle et la science des données. La figure 1.12 précise le paradigme : un *agent* évolue dans un *environnement* selon une *politique* π qui lui indique *l'action* a à effectuer selon *l'état* s actuel. Pour apprendre cette politique, l'agent mène une série *d'expériences* et renforce sa politique afin de maximiser la *récompense* sur le long terme [22]. Il s'agit d'un apprentissage semi-supervisé car la récompense peut ne

22. Dans le cadre du Q learning, ce processus se traduit par l'apprentissage d'une fonction de qualité. Dans l'approche "off-policy" telle que présentée dans la figure 1.12, la politique de mise à jour est différente de la politique de comportement. En effet, l'action a est choisie de manière gloutonne tandis que l'action A est choisie selon la politique π, qui peut favoriser l'exploration de temps à autre en prenant une action aléatoire.

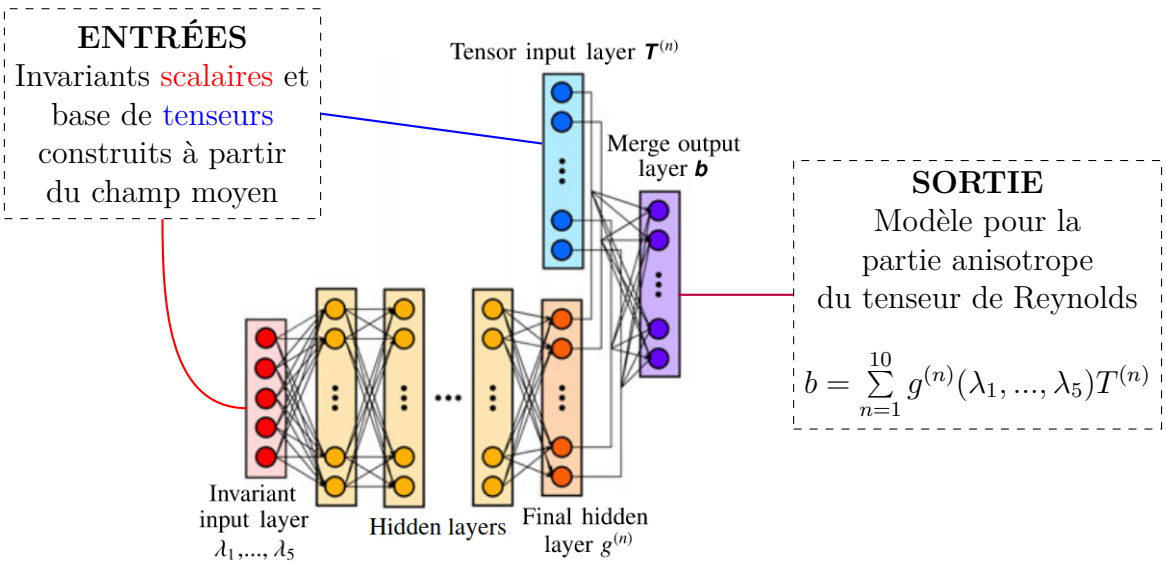


Figure *1.11* – *Modèle de fermeture par réseaux de neurones. Les tenseurs de déformation et de rotation du champ moyen sont utilisés pour construire des invariants* $(\lambda_1, ..., \lambda_5)$ *et une base de tenseurs isotropes* $(T^{(1)}, ..., T^{(10)})$ *assurant que le modèle appris respecte les symétries. Le modèle est construit sur la base de calculs fidèles d'écoulements canoniques. Tiré de Ling* et al. *(2016).*

survenir qu'à la toute fin d'un épisode et être très parcimonieuse [23].

Si l'apprentissage par renforcement est surtout connu dans le domaine des jeux, par exemple pour le jeu de Go (Silver *et al.*, 2017) ou les jeux Atari (Mnih *et al.*, 2015), son utilisation en mécanique des fluides est de plus en plus importante (Garnier *et al.*, 2019). On peut commencer par citer les travaux de Paris *et al.* (2021) et Rabault *et al.* (2019) sur la réduction de traînée autour d'un cylindre à bas nombre de Reynolds : un agent utilise de l'information partielle dans le sillage de l'écoulement pour actionner du soufflage en paroi du cylindre. L'objectif est similaire pour Fan *et al.* (2020) qui renforce une loi de contrôle d'écoulement autour d'obstacles non profilés et obtient une réduction de traînée de l'ordre de 30%. Une autre étude s'intéresse à la manière dont des poissons de queue exploitent la vorticité générée par la nage d'un poisson de tête (Verma *et al.*, 2018). Cet exemple issu de la nature illustre parfaitement les motivation d'un

23. Par exemple, une souris (l'agent) qui doit s'échapper d'un labyrinthe (l'environnement). La souris connaît sa position (l'état) dans le labyrinthe et choisit à chaque carrefour une direction (l'action). La récompense est un morceau de fromage, dont la saveur diminue avec le nombre de déplacements. La souris souhaite apprendre une politique de déplacement mais ne pourra statuer sur la qualité de ses actions qu'à la fin de l'expérience.

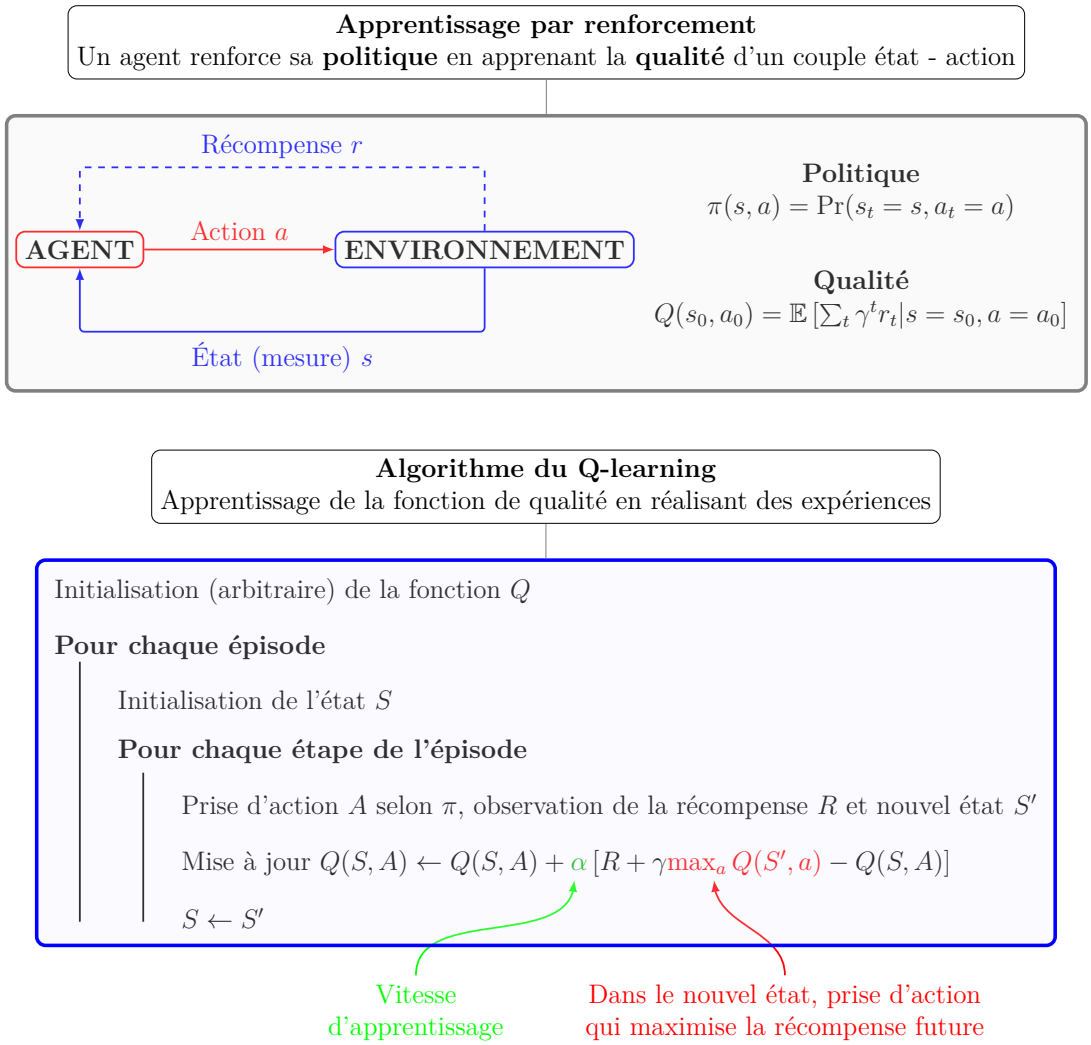
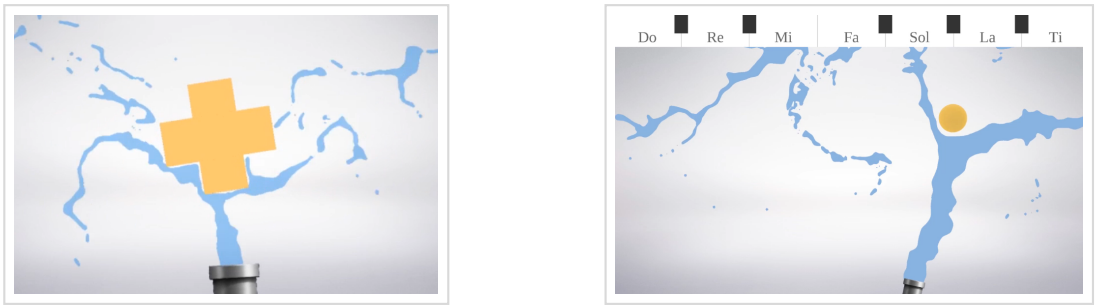


Figure *1.12* – *Principe de l'apprentissage par renforcement et Q-learning. Un agent suit une politique π qui donne l'action a à suivre vraisemblablement pour l'état s. La qualité Q d'une action a_0 prise dans l'état s_0 se mesure par la récompense espérée sur le long terme. L'importance des récompenses futures est actualisée selon le facteur d'oubli γ. L'agent apprend la fonction de qualité à partir d'expériences, ici dans une approche "off-policy". Une étape d'un épisode correspond à l'obtention d'une récompense et la mise à jour de Q. En apprentissage profond, la fonction Q prend la forme d'un réseau de neurones. Pour une revue détaillée des méthodes existantes, voir Garnier et al. (2019).*

apprentissage par renforcement : les poissons (agents) utilisent des informations partielles (capteurs épidermiques) pour exploiter la dynamique complexe d'un environnement dont ils ne connaissent pas les équations. Un autre sujet de recherche concerne le problème Zermolo pour la navigation optimale dans un écoulement turbulent (Biferale *et al.*, 2019). Enfin, on peut citer les applications [24] de Ma *et al.* (2018) pour contrôler une buse et agir sur un corps rigide (voir figure 1.13).



(a) Contrôle de la buse pour que la croix ne chute pas

(b) Contrôle de la buse pour jouer de la musique avec la balle

Figure *1.13* – *Contrôle par renforcement pour apprendre des lois de contrôle de solides rigides. L'article explore différentes applications : équilibre ou mise en rotation d'une croix, jeu de coopération ou de compétition entre buses, lecture d'une partition musicale et franchissement d'obstacles 3D. Tiré de Ma* et al. *(2018).*

L'apprentissage par renforcement est également utilisé en dynamique du vol. Tedrake *et al.* (2009) argumentent sur l'utilisation de l'apprentissage automatique pour imiter la nature et voler comme un oiseau. Les auteurs font une application : une maquette d'avion de deux mètres d'envergure apprend à se percher sur un coin, à la manière d'un oiseau se posant sur un câble électrique [25]. Concernant les drones, on peut citer deux travaux importants. Le premier s'intéresse au renforcement de la loi de contrôle d'un quadrotor pour effectuer des manœuvres complexes (Hwangbo *et al.*, 2017). Le second s'intéresse à l'apprentissage d'une loi de contrôle valable pour différentes configurations de drones hybrides [26] (Xu *et al.*, 2019). Ces méthodes peuvent se révéler très utiles pour la conception de drones

24. À voir via `http://gamma.cs.unc.edu/DRL_FluidRigid/video.mp4`

25. Ce qui est une manœuvre très complexe impliquant de décrocher - donc de détruire la portance - en créant une large zone de recirculation.

26. Drones qui peuvent passer d'une configuration hélicoptère à une configuration avion. La première configuration est plus stable et manoeuvrable. La seconde est plus performante sur de longues distances. Établir une loi de contrôle pour ces systèmes est très dépendant du design du drone, ce que le travail de Xu cherche à éviter.

autonomes. Enfin, on peut citer le travail de Reddy *et al.* (2018) qui optimisent la politique de vol d'un planeur en écoulement turbulent. La stratégie est déterminée à partir d'expériences effectuées avec une maquette de planeur pendant plusieurs jours. La figure 1.14 illustre quelques résultats de cette étude.

Finalement, l'apprentissage par renforcement est une réalité en mécanique des fluides, avec de nombreuses applications ces dernières années. Cette approche semi-supervisée permet d'apprendre des lois de contrôle sans connaissance complète [27] de l'environnement, un paradigme inspiré de la nature et prometteur lorsque la physique nous échappe. Notons toutefois que c'est une approche lente [28] et que ce n'est pas la seule approche possible pour du contrôle. Beaucoup de méthodes d'identifications de modèles sont étendues au contrôle. On peut notamment citer l'identification parcimonieuse avec contrôle (Brunton *et al.*, 2016c), la décomposition modale dynamique avec contrôle (Proctor *et al.*, 2016) et le cadre plus général KRONIC (*Koopman Reduced Order Nonlinear Identification and Control*) qui utilise la théorie de Koopman pour appliquer les outils de contrôle des systèmes linéaires (Kaiser *et al.*, 2017).

1.2.7 | Synthèse

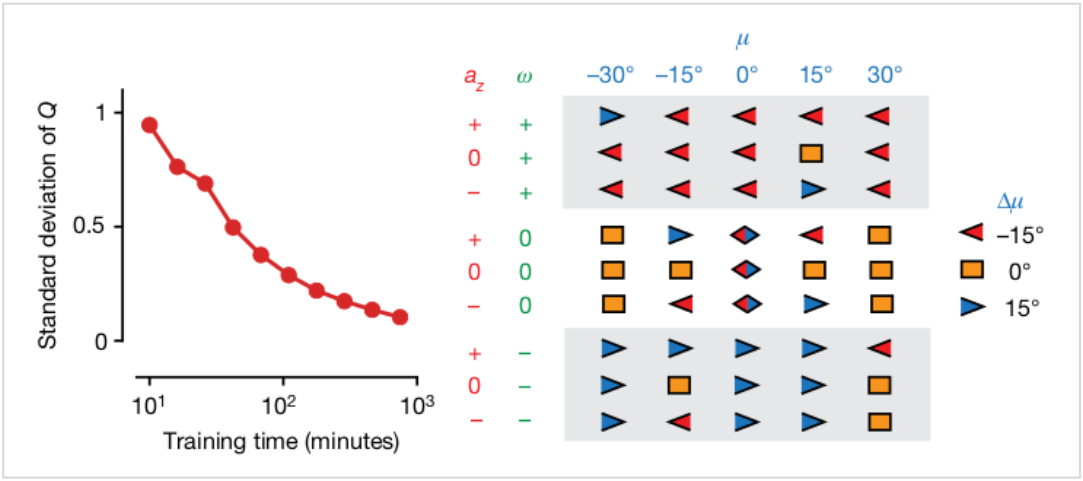
La disponibilités des données a ouvert la voie vers une nouvelle façon de penser la mécanique des fluides. De nombreuses avancées ont été faites grâce à l'apprentissage automatique pour la **modélisation** (SINDy et opérateur de Koopman), la **réduction** (auto-encodeurs), le **contrôle** (apprentissage par renforcement) et la **fermeture** (réseau de neurones artificiel contraint). Les méthodes sont toutefois validées sur des cas académiques [29] avec seulement quelques applications. Dans ce manuscrit, on va s'intéresser à la scalabilité de certaines des méthodes pour des écoulements de complexité croissante. En particulier, on se demande **dans quelle mesure les méthodes de réduction et de modélisation peuvent se combiner pour estimer un écoulement en temps réel**. Ce raisonnement est né d'un souhait : fournir une estimation de la carte du vent *instantanée* et *future* à un petit drone évoluant en milieu urbain. Dans la suite de cet état de l'art, on explicite les difficultés d'une telle tâche, les clés orientées données disponibles dans la littérature et le travail de thèse.

27. Par exemple, l'algorithme du Q-learning est purement *model free* et la fonction de qualité est apprise par expériences, sans connaître l'environnement. Il existe une autre classe de méthodes, dîte *model based*, où l'environnement est modélisé de manière déterministe ou probabiliste.

28. Dans une approche *model free*, il faut énormément de données.

29. Écoulement derrière un cylindre, système chaotique de Lorenz, oscillateur de Van Der Pol, attracteur de Rössler, oscillateur de Duffing, turbulence 2D, etc.

(a) Une maquette de planeur (2 mètres d'envergure) évolue au dessus d'un champ. La figure de gauche illustre l'une des trajectoires. La maquette a accès aux gradients de vitesse vertical (flèches bleues), l'accélération verticale et le moment de roulis. Les expériences sont réalisées avec un contrôleur de vol *initial* permettant de moduler l'inclinaison et l'assiette pour maximiser la portance.



(b) Apprentissage de la nouvelle politique de vol. À gauche, évolution de la fonction de qualité en fonction des expériences réalisées (1000 minutes d'expériences, étalées sur une quinzaine de jours). À droite, politique de vol finale. Chaque symbole correspond à la *meilleure* action à prendre selon l'état d'inclinaison μ, l'accélération verticale a_z et le moment de roulis ω. Ainsi, le contrôleur *final* indique un changement d'inclinaison de $-15°$ ou $15°$ pour un planeur dans l'état $(a_z < 0, \omega = 0, \mu = 0°)$.

Figure *1.14* – *Q-learning pour améliorer la politique de vol d'un planeur. Une maquette de planeur équipée réalise des expériences qui seront utilisées pour apprendre la nouvelle politique. L'objectif est de déterminer l'action optimale sur l'angle d'inclinaison pour espérer un gain d'altitude sur le long terme. Figures tirées de Reddy et al. (2018)*

1.3 | Application visée : navigation en écoulement urbain

Dans le contexte d'une *mobilité aérienne urbaine*, de nouveaux systèmes de transport aérien au sein des villes sont en développement. Par exemple, de petits

drones autonomes pourraient être utilisés pour assurer des missions de livraison [30], de surveillance et d'inspection de bâtiments. Mais la tâche n'est pas aisée : les écoulements urbains sont très fluctuants, avec des niveaux de turbulence élevés (Sheng *et al.*, 2018; Hui *et al.*, 2009; Li *et al.*, 2019). Pour éviter les obstacles géométriques et fluidiques [31], un drone autonome devrait prendre des décisions *intelligentes* en temps réel.

Une première approche consisterait à renforcer le comportement du drone avec les méthodes présentées dans la section 1.2.6. L'idéal serait d'apprendre la politique en situation réelle mais cela soulève de nombreuses questions pratiques [32]. Une seconde approche consisterait à définir des lois de commande robustes (Perozzi *et al.*, 2018) prenant en entrée une estimation de la carte du vent. Disposant d'une telle carte et d'un modèle dynamique de l'écoulement, il serait possible de prédire le futur de l'écoulement et présager des zones dangereuses. Pour des missions autonomes, ce travail ne saurait se faire uniquement *offline* avec les équations de Navier Stokes (un travail déjà effectué par Galway *et al.* (2012) et basé sur la combinaison de cartes CFD). Le travail devrait se faire *online* avec un modèle dynamique réduit dont les conditions initiales seraient séquentiellement estimées par le drone à partir de mesures de l'environnement [33].

Pour mieux comprendre la difficulté de la tâche, on regarde plus en détail la physique de l'écoulement. Pour cela, on considère la partie basse de la couche limite atmosphérique (Kaimal et Finnigan, 1994) qui constitue la couche limite urbaine (Barlow, 2014). Celle-ci se décompose en une sous couche rugueuse (proche paroi) et une sous couche inertielle. Dans la sous couche rugueuse, l'écoulement dépend entièrement des éléments de rugosité et sa structure est complexe (Kastner-Klein et Rotach, 2004; Blackman *et al.*, 2017). Dans la sous couche inertielle, les flux turbulents sont constants et le champ de vitesse moyenne prend la forme d'une loi logarithmique ou puissance (Buschmann et Gad-el Hak, 2003) dont les paramètres sont normés selon le terrain (ESDU, 1982). Pour un prisme monté dans une telle couche limite, l'écoulement est fortement instationnaire et dépend de nombreux facteurs [34] (Martinuzzi et Tropea, 1993). La figure 1.15 illustre la topologie de l'écoulement instantanné derrière un bâtiment élancé (McClean et Sumner, 2014). On y observe un tourbillon en fer à cheval en amont du bâtiment et des décollements géométriques qui interagissent pour former un sillage. Pour un diamètre proche de la hauteur de la tour, le sillage prend la forme d'un tourbillon en arche transporté vers l'aval. Pour une hauteur au moins

30. Un chemin sur lequel Amazon Prime Air semble bien engagé
31. Comme des rafales de vent, des mouvements ascendants, des bouffées, etc.
32. Le drone pourrait se casser. Il faut aussi disposer d'un environnement urbain sans population pour éviter les dangers, ou reproduire un tel environnement en soufflerie. La deuxième solution n'est pas une mince affaire : il faut générer les bonnes statistiques de la turbulence et disposer d'une soufflerie assez grande pour y faire voler un drone.
33. Des informations représentatives de l'écoulement mais potentiellement bruitées.
34. Dont le rapport d'aspect du prisme, le dérapage de l'écoulement (Becker *et al.*, 2002) et le nombre de Reynolds. À noter que dans le cas d'une couche limite turbulente, la dépendance au nombre de Reynolds disparaît au delà d'une valeur critique (Snyder et Castro, 2002).

deux fois supérieures à la hauteur, la symétrie est rompue et on observe des allées de Karman (Wang et Zhou, 2009).

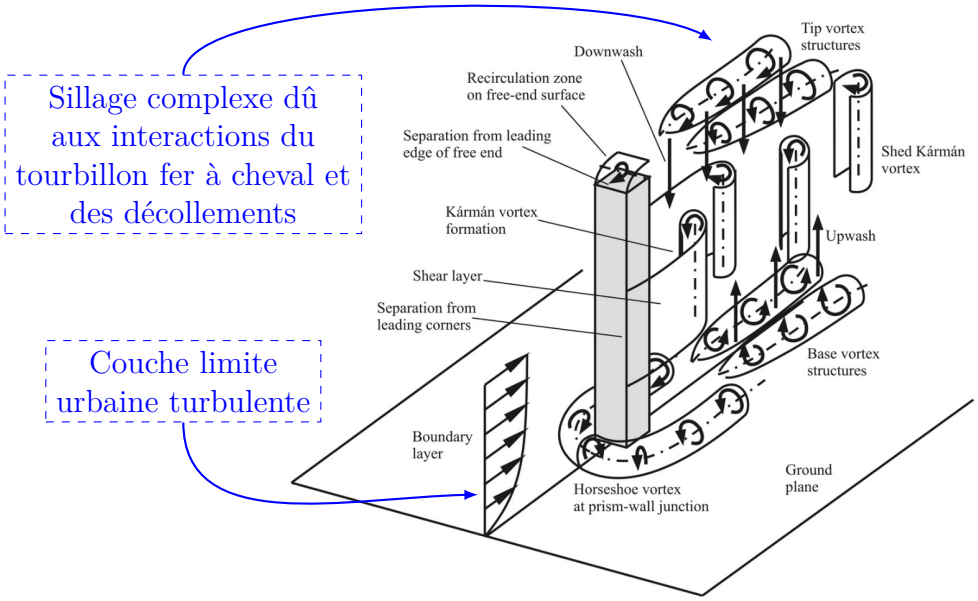


Figure *1.15* – *Topologie de l'écoulement autour d'un bâtiment élancé. Tiré de McClean et Sumner (2014).*

Plusieurs obstacles compliquent donc une estimation *online* de l'écoulement. Premièrement, la dynamique est tri-dimensionnelle, fortement instationnaire et avec évenements rares du fait d'une activité turbulente intense. Il en résulte un état complet de très haute dimension. Deuxièmement, le sillage est transporté vers l'aval. Or les méthodes de réduction *simples* (comme la POD) ne sont pas adaptées aux problèmes d'ondes progressives. Un travail important est donc requis pour extraire une variété [35] latente pertinente. Idéalement, il faudrait classifier les zones de l'écoulement et faire des réductions locales, comme suggérées par les approches CROM de Kaiser *et al.* (2014) et NIROM (*Non Intrusive Reduced Order Model*) de Xiao *et al.* (2019). Troisièmement, établir une base de données sur laquelle apprendre est un travail à part entière, avec des réflexions autour du maillage, des méthodes numériques (Blocken, 2018) et de la couche limite atmosphérique. De plus, la base de données doit couvrir différentes conditions d'entrées (nombre de Reynolds, angle de dérapage, configuration urbaine) pour espérer déployer des modèles (dynamique et d'estimation) généralisables. Du point de vue de la donnée, cela augmente drastiquement le coût de calcul et les besoins en termes de stockage. Du point de vue des modèles, cela augmente la complexité mathématique et les besoins en mémoire vive [36]. Quatrièmement, le modèle d'estimation doit s'appuyer

35. Structure topologique qui ressemble localement à un espace euclidien.
36. Du point de vue mathématique, le modèle doit apprendre à générer des données pertinentes

sur de la connaissance partielle, par exemple des mesures LIDAR (*LIght Detection And Ranging*) du champ de vitesse (Floors *et al.*, 2013). Le choix, le nombre et le positionnement des capteurs est crucial pour capturer les éléments essentiels de la dynamique. Enfin, disposer du modèle d'estimation n'est que la première étape du processus ; le petit drone doit être capable de l'utiliser intelligemment en temps réel, ce qui passe par la définition de lois de commande.

Les outils présentés dans la section 1.2 apportent quelques éléments de réponses sur l'étape de réduction (absolument nécessaire) et sur l'identification d'un modèle dynamique (requise pour estimer la carte de vent future). La section suivante ajoute de nouvelles clés concernant l'estimation orientées données à partir de mesures ponctuelles. Le trousseau de clés en main, nous présenterons alors les objectifs de la thèse et le cadre d'étude.

1.4 | Estimation orientée données d'un écoulement

Cette section se consacre aux méthodes d'estimation d'un écoulement, regroupées dans la littérature sous le nom de *méthodes de reconstruction*. L'objectif est de retrouver l'état complet du système (par exemple champ de vitesse) à partir de sa mesure. La reconstruction intervient dans de nombreuses applications comme le contrôle actif pour la réduction de traînée (Paris *et al.*, 2021; Pfeiffer, 2016) l'identification de sillages de bateaux (Graziano *et al.*, 2016), les sciences climatiques (Kalnay, 2003) ou la modélisation du flux cardiaque (Sankaran *et al.*, 2012). Les méthodes de reconstruction sont également utilisées en mécanique des fluides numérique : des clichés complets sont récupérés *a posteriori* à partir de signaux ponctuels sauvegardés sur le disque pendant la résolution (Carlberg *et al.*, 2019).

Les méthodes de reconstruction se classent en trois catégories (Callaham *et al.*, 2019). La première est **l'estimation directe** où l'état complet est écrit comme une combinaison linéaire d'éléments de référence. Chaque estimation nécessite de résoudre un problème d'optimisation basé sur le vecteur de mesures, ce qui limite l'utilisation à des applications *offline*. On utilise le terme *direct* car l'estimation ne s'appuie sur aucun modèle, juste une matrice de mesures et des éléments de référence. La seconde est **l'estimation régressive** où les outils de l'apprentissage supervisé sont utilisés pour complètement apprendre la relation de passage entre l'espace des mesures et l'espace d'état. De cette manière, une estimation requiert simplement l'évaluation du modèle. Tout l'enjeu consiste à choisir une bonne forme pour le modèle et à ne pas le sur-apprendre. La troisième est **l'assimilation de données**. La mesure est utilisée pour mettre à jour la prédiction d'un modèle.

pour de nouvelles configurations (nombre de Reynolds, angle de dérapage, configuration urbaine) non couverts par la base de données, comme étudié dans Morton *et al.* (2021) sur l'écoulement autour d'un cylindre en rotation.). Du point de vue de l'ordinateur, il faut charger en mémoire des matrices de taille importante.

La méthode est construite sur le principe d'inférence bayésienne : la distribution *a posteriori* de l'état est proportionnelle au produit de la distribution *a priori* de l'état et de la *vraisemblance* de la mesure. Chacune des stratégies est illustrée sur la figure 1.16.

Les sections suivantes présentent plus en détail chacune des méthodes ainsi que leur utilisation en mécanique des fluides.

1.4.1 | La reconstruction directe

Avant de formellement introduire le problème, digressons sur la compression d'images. Écrite dans l'espace des fréquences, une image a une représentation parcimonieuse car peu de fréquences sont actives parmi toutes les harmoniques possibles. Si l'on dispose d'une partie des pixels (i.e. une mesure de l'image) et que l'objectif est de reconstruire l'image, il est donc plus judicieux de reconstruire dans l'espace des fréquences [37]. C'est le principe de *l'acquisition comprimée*, illustré sur la figure 1.17.

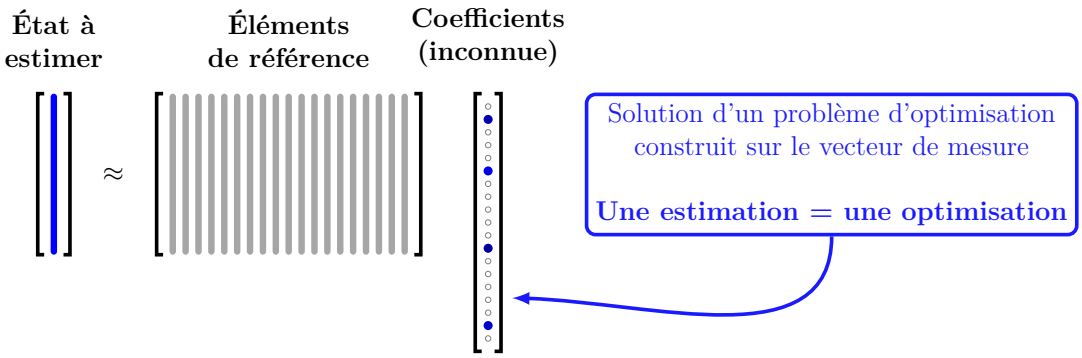
Dans un cadre plus général, si on note x le signal initial, C la matrice de capteurs, $y = Cx$ la mesure et Ψ la base (connue) dans laquelle le signal est parcimonieux (notation s), on doit résoudre le problème d'optimisation :

$$\hat{s} = \arg\min_{s} ||s||_0 \text{ tel que } y = Cx = C\Psi s$$

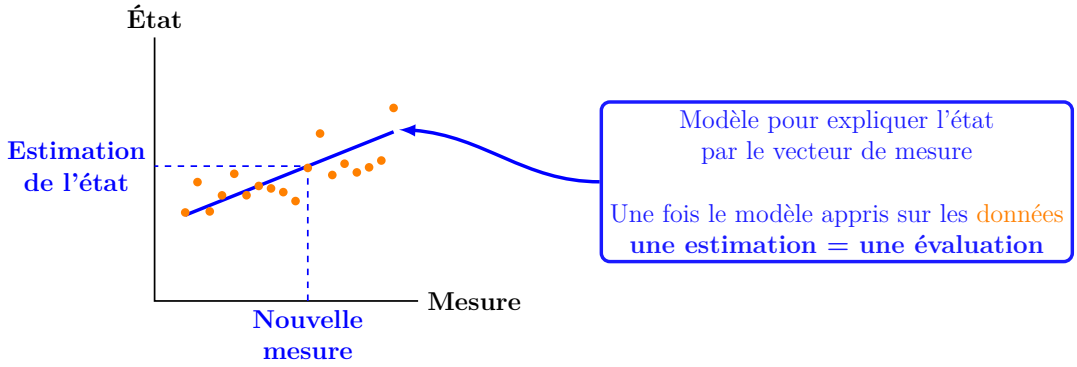
Ici, $\hat{s}$ désigne l'estimation du signal compréssé, ensuite utilisé pour estimer le signal plein x avec $\hat{x} = \Psi\hat{s}$. La norme l_0 représente la cardinalité de s que l'on souhaite minimiser. C'est toutefois une norme non convexe, requérant une recherche combinatoire de la solution. Sous certaines conditions, le problème d'optimisation peut heureusement être relaxé en faisant intervenir la norme l_1 qui promeut la parcimonie dans la solution [38] (Candes *et al.*, 2006; Spielman *et al.*, 2012). Pour ce qui nous intéresse, le signal est plutôt appelé *état* tandis que la matrice de transformation correspond aux *éléments de référence*. La figure 1.18 montre un exemple d'utilisation en mécanique des fluides, où cinq capteurs de vorticité sont

37. C'est une représentation obtenue par transformée de Fourier spatiale. Si l'image contient $M \times N$ pixels et que le pixel (m, n) de l'image est noté $f[m, n]$, la transformée s'écrit $F[k, l] = \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[m, n] \exp\left(-j2\pi \left\{\frac{mk}{M} + \frac{nl}{N}\right\}\right)$. L'inverse est donné par $f[m, n] = \frac{1}{\sqrt{MN}} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} F[k, l] \exp\left(j2\pi \left\{\frac{mk}{M} + \frac{nl}{N}\right\}\right)$. Intuitivement, on teste dans chaque direction la présence de cosinus ou sinus *spatiaux*, la longueur d'onde minimale étant fixé par la taille d'un pixel. C'est exactement pareil qu'un signal temporel où l'on chercherait la présence de cosinus ou sinus *temporels* dont la fréquence minimale serait fixé par la résolution de l'acquisition.

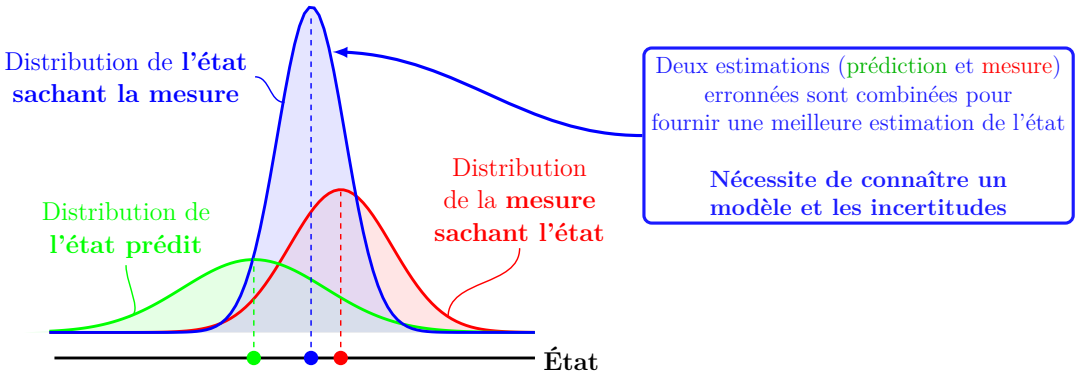
38. La matrice $C\Psi$ doit satisfaire le principe d'isométrie restreinte pour que la solution au problème l_1 converge avec une très forte probabilité vers la solution l_0. On rappelle que l_0 désigne le nombre d'éléments non nuls dans le signal tandis que l_1 est la somme des valeurs absolues de toutes les composantes.



(a) Estimation directe. L'état est estimé par la combinaison linéaire d'éléments de référence, par exemple des modes d'écoulements issus d'une décomposition modale.



(b) Estimation régressive. L'état est estimé par un modèle appris de manière supervisée. Dans l'illustration, le modèle explique linéairement l'état par les mesures.



(c) Estimation par assimilation de données. La distribution de l'état prédit est mise à jour grâce à la distribution de la mesure. Ici, la mesure est l'état complet et les distributions sont normales, facilitant l'application du théorème de Bayes.

Figure *1.16* – *Illustration graphique des méthodes d'estimation de l'état d'un système par sa mesure.*

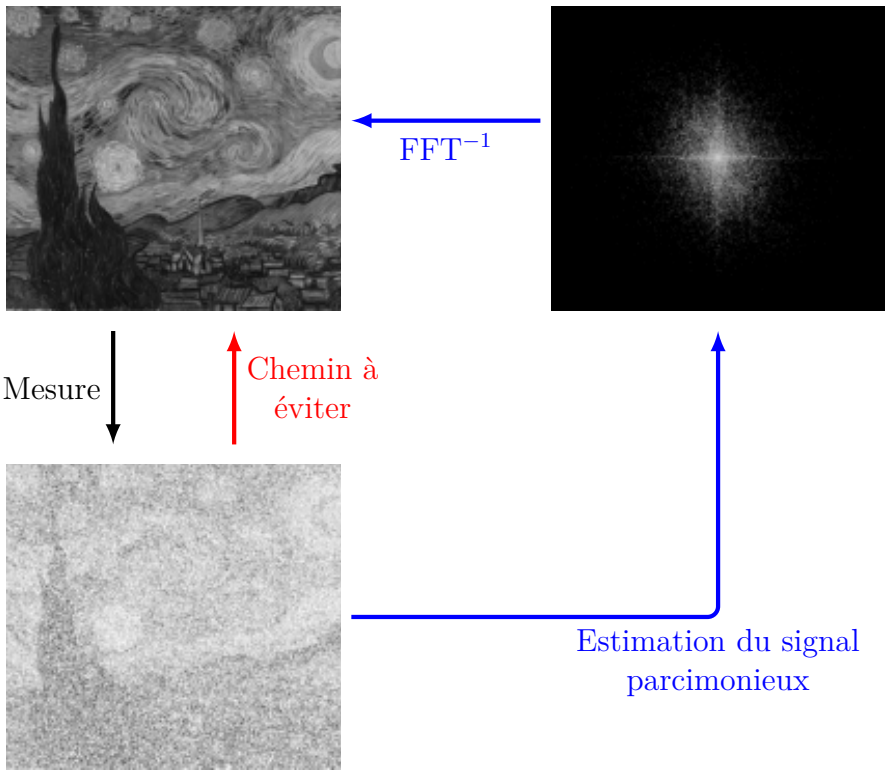


Figure *1.17* – *Principe de l'acquisition comprimée. On veut estimer le signal plein (ici l'image) à partir de sa mesure (ici une sélection aléatoire de pixels). Lorsque l'on connaît une base où le signal est parcimonieux (l'image écrite dans l'espace des fréquences), il est judicieux d'utiliser la mesure pour estimer la représentation creuse puis repasser dans l'espace initial (chemin bleu). Cela diminue la complexité du problème par rapport à une estimation naïve (chemin rouge). Remarque : FFT est l'acronyme de Fast Fourier Transform.*

utilisés pour estimer le champ de vorticité complet en aval d'un cylindre [39] (Callaham *et al.*, 2019). Tout l'enjeu consiste à définir les bons éléments de référence Ψ et la bonne matrice de mesure C.

39. L'article dont est tirée la figure est très formateur. L'auteur présente toutes les variantes du problème d'optimisation et étudie la robustesse au bruit de la reconstruction parcimonieuse. Il présente aussi une stratégie de reconstruction locale pour scinder le problème d'optimisation en plusieurs petits problèmes.

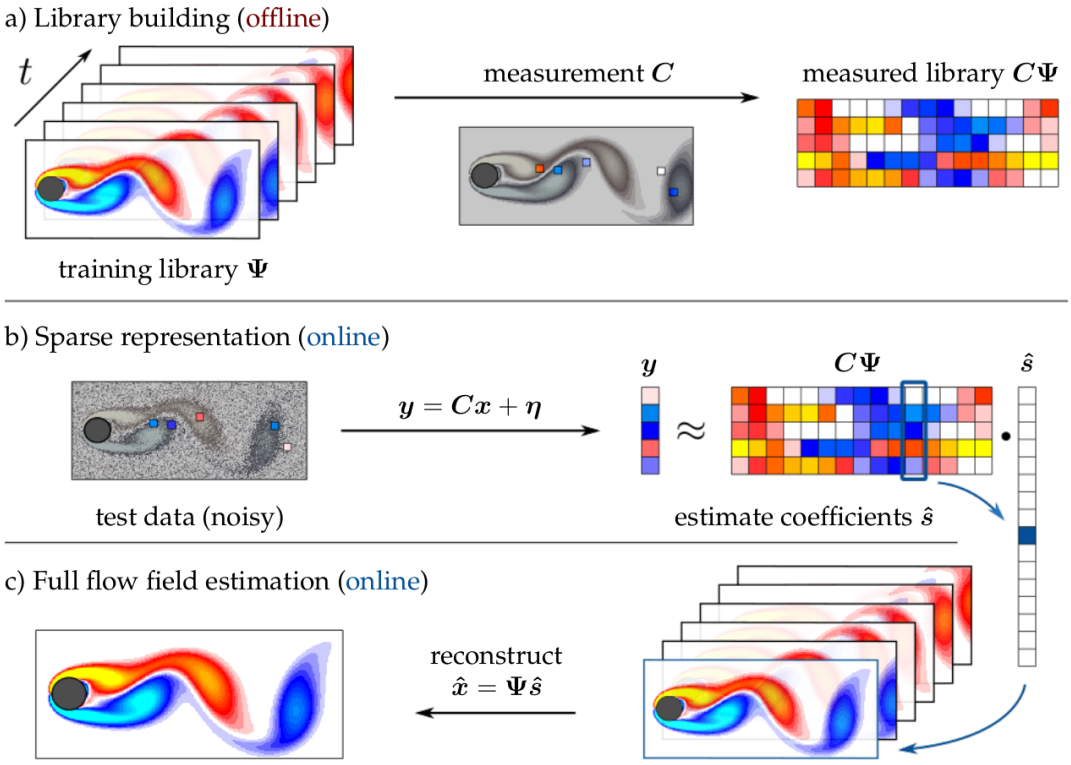


Figure *1.18* – *Reconstruction directe illustrée sur l'écoulement laminaire autour d'un cylindre. La mesure bruitée du champ de vorticité (cinq capteurs) est écrite comme la combinaison linéaire d'éléments de référence, ici les clichés mesurés de l'écoulement. Le vecteur de coefficients est ensuite utilisé pour estimer le cliché d'où provient le vecteur de mesure. Figure tirée de Callaham et al. (2019)*

Le choix des éléments référence

Les éléments de référence sont souvent issus d'une décomposition modale générique (par exemple la transformation de Fourier) ou sont adaptés à la donnée (Taira *et al.*, 2017). Pour le deuxième cas, il est important de citer l'algorithme *gappy POD* proposé par Everson et Sirovich (1995). Dans l'article, les auteurs reconstruisent une image bardée de pixels morts à partir d'images propres, obtenues par analyse en composantes principales. En mécanique des fluides, l'algorithme a été utilisé pour diverses tâches dont la reconstruction directe d'écoulements instationnaires (Willcox, 2006; Venturi et Karniadakis, 2004) et la reconstruction d'écoulements 3D à partir de mesures 2D (Podvin *et al.*, 2006; Murray et Ukeiley, 2007). Le dictionnaire peut également être construit en imposant le besoin de parcimonie pour le signal compréssé. C'est le cas des

algorithmes GOBAL [40] et K-SVD [41] qui encodent les données de manière sparse comparé à une décomposition orthogonale classique. Enfin, on peut citer l'utilisation des réseaux de neurones pour constuire les éléments de référence. Al Mamun *et al.* (2018) utilisent par exemple un réseau ELM [42] pour auto-encoder l'état d'un écoulement et les modes de références sont les poids du réseau. Les auteurs montrent qu'une tel dictionnaire, bien que non interprétable, permet une reconstruction robuste au positionnement des capteurs par rapport à un dictionnaire POD.

Le placement des capteurs

Dans de nombreuses applications, le placement des capteurs est aléatoire. De cette manière, la probabilité que le vecteur de mesures $\boldsymbol{y}$ reflète les termes actifs du signal compressé $\boldsymbol{s}$ est élevée (Candes et Tao, 2006). Mathématiquement, cela se justifie par le *principe d'isométrie restreinte* qui assure que la transformation $C\psi$ agit comme une *isométrie*. En choisissant aléatoirement les capteurs, les matrices C et ψ sont vraisemblablement incohérentes [43] et la solution du problème convexe l_1 converge vers la solution du problème non convexe l_0 (voir page 90 de Brunton et Kutz (2019) pour une illustration et Candes et Tao (2006) pour la démonstration). Parmi les autres méthodes de placement, on peut citer celles qui minimisent le conditionnement de $M = \theta^T\theta$ avec $\theta = C\psi$. L'inversion de la matrice M est requis lorsque l'utilisation de la norme l_2 est plus adéquat (Jayaraman *et al.*, 2019) comme dans le cas d'une reconstruction par *gappy POD* (Willcox, 2006; Yildirim *et al.*, 2009). La minimisation du nombre de condition peut se baser sur une décomposition QR de la matrice M mais la complexité calculatoire augmente rapidement et la méthode est finalement peu utilisée (Manohar *et al.*, 2018a). Enfin, il existe des méthodes de placement physique de capteurs, par exemple aux *extrema* des modes dominants d'une décomposition orthogonale (Cohen *et al.*, 2003) ou aux points dynamiquement intéressants en termes d'observavilité et de contrôlabilité (Manohar *et al.*, 2018b).

40. *Goal-Oriented BAsis Learning*

41. Décomposition en valeurs singulières (SVD) pour construire K éléments qui encodent de manière parcimonieuse les données. C'est une version améliorée de l'algorithme des K-moyennes.

42. *Extreme Learning Machine*. Il s'agit d'un réseau avec une seule couche cachée, dont les paramètres tels que les poids peuvent ne pas être mis à jour. Les poids entre la couche d'entrée et la couche cachée sont initialisés aléatoirement et une régression moindres carrés permet de déterminer les poids entre la couche cachée et la couche de sortie.

43. La cohérence entre ces deux matrices est définie par $\mu(C,\psi) = \sqrt{n}\max_{i,j}\left[C_{[i,:]}\psi_{[:,j]}\right]$. Dans cette formule, $C_{[i,:]}$ est la i-ème ligne de la matrice de mesure et $\psi_{[:,j]}$ est le j-ème mode ; la cohérence est donc le produit scalaire. Plus ce nombre est proche de l'unité (borne inférieure) et plus les matrices sont incohérentes : les composantes d'un vecteur singulier de $C\psi$ sont semblables. Plus ce nombre se rapproche de $\sqrt{n}$ (borne supérieure) et plus les matrices sont cohérentes : les vecteurs singuliers sont très parcimonieux. Dans ce deuxième cas, les termes actifs du vrai signal compressé ne sont pas captés par la mesure et l'estimation du signal par la mesure sera mauvaise.

1.4.2 | La reconstruction par régression

La reconstruction par régression a été introduite en mécanique des fluides en 1975 sous le nom *d'estimation stochastique* (SE). L'objectif était d'analyser des bases de données pour déterminer les structures cohérentes d'un écoulement turbulent (Adrian, 1975). Dans cette méthode, la quantité d'intérêt (par exemple champ de vitesse fluctuante $\boldsymbol{u}$) est estimée par la moyenne conditionnelle :

$$\mathbb{E}\left[\boldsymbol{u}|\boldsymbol{y_1},...,\boldsymbol{y_p}\right]$$

où $\boldsymbol{y}_i$ désigne un événement conditionnel [44] de l'écoulement au point $(\boldsymbol{x}_i, t)$. Un développement de Taylor fournit une approximation linéaire (LSE), quadratique (QSE) ou d'ordre plus élevé (HOSE) du champ de vitesse fluctuante. Par exemple, l'estimation quadratique s'écrit :

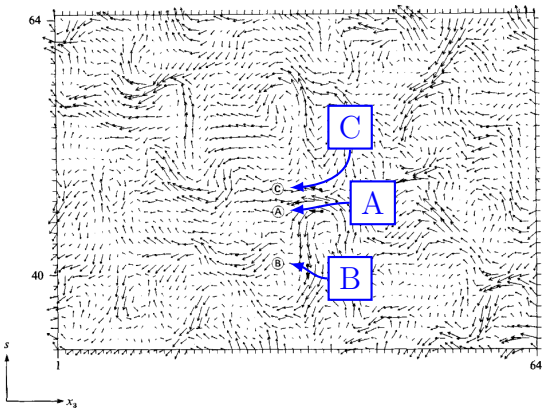
$$\boldsymbol{u}(\boldsymbol{x},t) \approx \sum_{i=1}^{p} A(\boldsymbol{x},\boldsymbol{x}_i)\boldsymbol{y}_i(t) + \sum_{i=1}^{p}\sum_{j=1}^{p} B(\boldsymbol{x},\boldsymbol{x}_i,\boldsymbol{x}_j)\boldsymbol{y}_i(t)\boldsymbol{y}_j(t)$$

Mathématiquement, il s'agit d'une régression linéaire multi-variée. Les coefficients de la régression sont appris en minimisant l'erreur quadratique moyenne entre les clichés d'entraînement et l'estimation par le modèle. Comme exemple fondateur, on peut citer Adrian et Moin (1988) où les auteurs utilisent l'estimation stochastique linéaire sur une DNS d'un écoulement cisaillé et isolent des structures en épingles à cheveux (voir figure 1.19). À l'époque, l'un des avantages mis en avant pour l'estimation stochastique était qu'elle repose sur des statistiques mesurables avec seulement deux sondes.

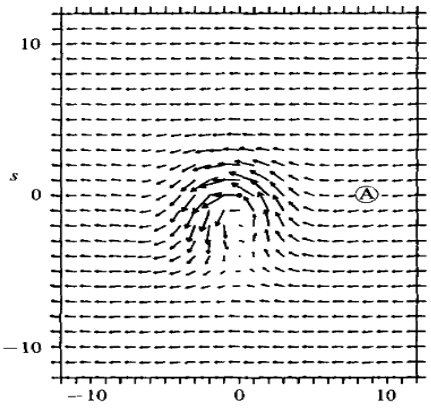
La méthode a évoluée depuis. En introduisant de la décomposition orthogonale propre (Berkooz *et al.*, 1993), l'estimation ne porte plus sur le champ de vitesse mais sur les coefficients de la décomposition (méthode SE-POD, voir Durgesh et Naughton (2010)). Avec l'utilisation de fenêtres d'observation, l'estimation ne s'appuie plus sur les mesures au temps courant mais aussi sur leur histoire (méthode SE-MTD, voir Caraballo *et al.* (2007)). Parmi les principales applications, on trouve l'étude de la turbulence isotrope (Adrian, 1979), de couches limites turbulentes (Guezennec, 1989), de jets axisymétriques (Bonnet *et al.*, 1994), d'expansions brutales (Cole et Glauser, 1998) et de cavités (Murray et Ukeiley, 2003). On peut également citer l'augmentation *a posteriori* de données PIV [45] à partir de signaux résolus en temps (Tu *et al.*, 2013). À noter que pour le placement de capteurs, les travaux sont moins riches que pour l'acquisition comprimée. Généralement, les mesures sont prises dans les zones d'intérêt de l'écoulement, comme pour la couche cisaillée dans Adrian et Moin (1988). Sur des applications plus élaborées, les capteurs sont choisis de manière à expliquer le champ de vitesse *plus* qu'ils ne s'expliquent linéairement entre eux, pour éviter les multi-colinéarités. Ce principe est formalisé avec le facteur d'inflation de la variance (*Variance*

44. Une mesure pas nécessairement homogène du champ à estimer. Par exemple, des mesures de pression pour estimer le champ de vitesse.

45. *Vélocimétrie par Images de Particules*



(a) DNS du champ de vorticité et événements conditionnels A, B et C.



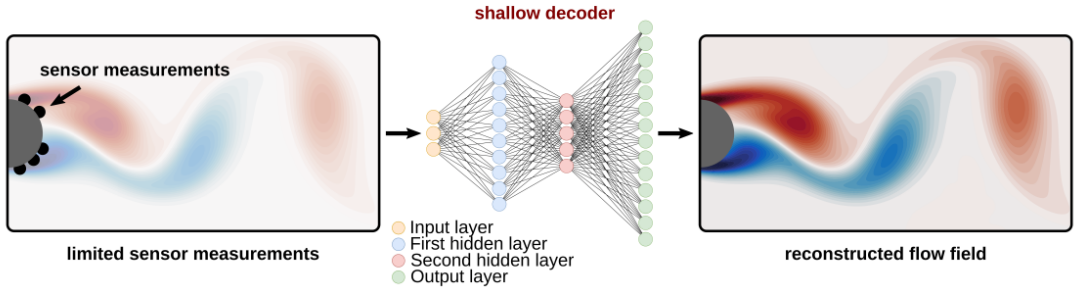
(b) LSE sur l'événement A.

Figure *1.19* – *L'une des premières applications de l'estimation stochastique. L'auteur identifie une structure en épingle à cheveux dans un écoulement cisaillé. Figures extraites de Adrian et Moin (1988)*

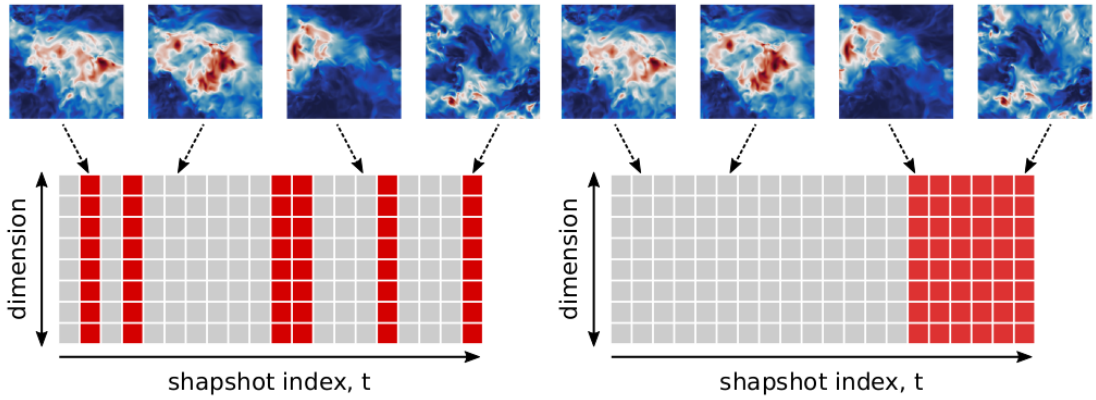
Inflation Factor ou VIF) que l'on souhaite proche de l'unité (Arnault *et al.*, 2016).

Aujourd'hui, l'estimation ne se limite plus à la régression linéaire. Récemment, les chercheurs s'intéressent à des méthodes plus *flexibles* de l'apprentissage automatique. Le réseau de neurones superficiel Erichson *et al.* (2020) est un exemple d'estimation stochastique où la relation entrée - sortie est une boîte noire. La figure 1.20 résume les résultats de l'article. Pour l'écoulement laminaire autour d'un cylindre, les auteurs utilisent l'information pariétale pour reconstruire le champ de vorticité. Pour des écoulements plus complexes tels que de la turbulence homogène isotrope, le réseau superficiel n'est pas capable d'extrapoler les futurs champs de vitesse. Le réseau est plutôt adaptée aux tâches d'interpolation et à la super-résolution [46]. Cet article est important car il reformule l'estimation stochastique sous la forme d'apprentissage supervisé. Depuis, d'autres méthodes régressives telles que les forêts aléatoires ou les machines à vecteurs supports ont été testé (Fukami *et al.*, 2020; Carlberg *et al.*, 2019; Matsuo *et al.*, 2021). On peut aussi citer le travail dans la super-résolution d'écoulements avec des approches semi-supervisées telles que les réseaux génératifs antagonistes (*Generative Adversarial Network* ou GAN, voir Kim *et al.* (2019)) mais la motivation principale n'est pas la physique [47].

46. Principe du logiciel RAISR de Google.
47. L'objectif est plutôt de faire une animation réaliste. Il y a d'ailleurs un auteur des studios Pixar.

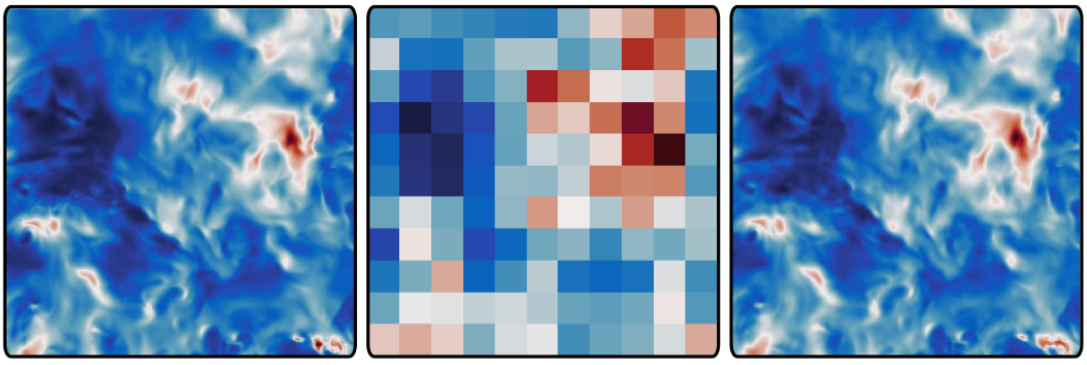


(a) Reconstruction régressive par réseau de neurones de l'écoulement laminaire autour d'un cylindre. Cinq capteurs de pression pariétale sont utilisés pour estimer le champ de vorticité.



(A) Interpolation (B) Extrapolation

(b) Les deux problèmes d'estimation. Pour un problème d'interpolation, les clichés de test sont situés entre les clichés d'apprentissage. Pour le problème d'extrapolation, les clichés de test sont situés après les clichés d'apprentissage.



(A) Snapshot (B) Low resolution (C) Shallow Decoder

(c) Exemple d'estimation pour un problème d'interpolation. La mesure du cliché réel (A) est une image très basse résolution (B). Le réseau de neurones estime l'image de haute résolution (C).

Figure *1.20* – *Estimation stochastique par réseau de neurones superficiel. Résultats tiré de Erichson et al. (2020), un article fondateur des méthodes de reconstruction par apprentissage supervisé.*

1.4.3 | L'estimation par assimilation de données

La dernière méthode de reconstruction est l'assimilation de données. Étant donné une condition initiale pour le système, un modèle dynamique (déterministe ou probabiliste) est utilisé pour prédire l'état futur. Selon les incertitudes du modèle (lié à sa stabilité), les erreurs peuvent s'accumuler et détériorer la qualité de la prédiction sur le long terme. L'assimilation de données vise à contrer cette accumulation d'erreurs en assimilant des mesures (potentiellement bruitées) du système.

L'assimilation de données a principalement été développée en météorologie et océanographie. En mécanique des fluides, elle donne un cadre mathématique rigoureux pour combiner des prédictions numériques avec des données expérimentales. Cela permet notamment de mieux décrire les entrées du solveur numérique, comme les conditions initiales, les conditions limites et les paramètres de modèles (Kato *et al.*, 2015; Gronskis *et al.*, 2013; Symon *et al.*, 2020). Deux approches existent : les approches *séquentielles* et les approches *variationnelles* (Mons *et al.*, 2016).

Dans les approches séquentielles, la densité *a priori* de l'état est mise à jour lorsque la mesure est disponible. La densité *a posteriori* de l'état est calculée avec le théorème de Bayes (Wikle et Berliner, 2007). La mise-à-jour fait apparaître un gain, dit de Kalman (Kalman, 1960), qui combine les erreurs de mesure et de prédiction. Différentes versions du filtre de Kalman existent, selon que le modèle dynamique est linéaire, différentiable ou non linéaire (Chowdhary et Jategaonkar, 2010). En mécanique des fluides, Suzuki (2012) a implémenté un filtre de Kalman étendu pour compléter des données PIV d'un écoulement de jet à partir d'une DNS bidimensionnelle de l'écoulement. Colburn *et al.* (2011) ont utilisé le filtre de Kalman ensembliste pour estimer les grandeurs statistiques d'un écoulement turbulent de canal. Enfin, Sousa *et al.* (2018) ont utilisé un filtre de Kalman pour mettre à jour les conditions initiales d'une simulation numérique d'écoulement urbain (voir figure 1.21). L'assimilation séquentielle des données est prometteuse pour des applications temps réel. Toutefois, le succès de la méthode repose sur la connaissance des incertitudes et d'un modèle dynamique. Les approches orientées données permettent d'aborder ces deux problèmes, comme illustré sur le système de Lorenz 63 (Dubois *et al.*, 2020).

Dans les approches variationnelles, une fonction coût est minimisée. Toutes les mesures dans une fenêtre d'observation sont utilisées [48]. La stratégie n'est pas adaptée pour du temps réel et sert plutôt à l'augmentation des données (Le Dimet et Talagrand, 1986). En mécanique des fluides, les approches variationnelles ont été appliquées aux équations de Navier Stokes, par exemple avec Gronskis *et al.* (2013) qui ont déterminé les conditions limites optimales d'une DNS à partir de mesures expérimentales. On peut aussi citer les travaux de Symon pour assimiler le terme

48. Donc les observations peuvent être passées **et** futures. L'assimilation de données variationnelle est donc plus adaptée à l'augmentation de données, pas à la mise en jour en temps réelle.

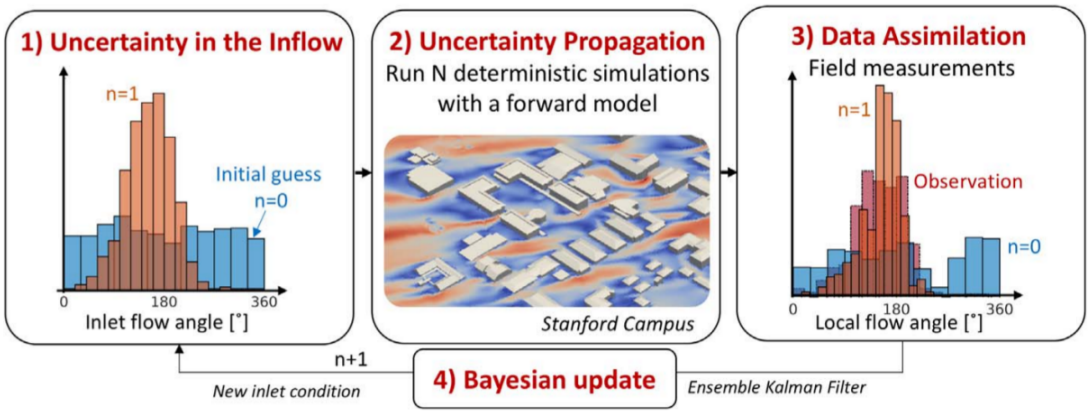


Figure *1.21* – *Assimilation de données par filtre de Kalman pour mettre à jour la carte de vent en entrée d'une simulation numérique d'écoulement urbain. Des mesures locales du vent sont utilisées pour déterminer la distribution* a posteriori *de la carte de vent, dont la distribution* a priori *est fournir par CFD. Figure tirée de Sousa* et al. *(2018)*

de forçage dans les équations moyennées (Symon *et al.*, 2017) et les travaux de Mons *et al.* (2016) pour assimiler des conditions d'entrées instationnaires. À noter que le filtre de Kalman existe aussi dans une formulation variationnelle, que l'on appelle *lissage* de Kalman.

1.4.4 | Synthèse

L'estimation d'un écoulement par des mesures ponctuelles peut se faire de trois manières. Dans l'approche directe, le champ à estimer est écrit comme une combinaison linéaire d'éléments de référence. La recherche d'une solution parcimonieuse s'inscrit dans la théorie de l'acquisition comprimée, qui donne les clés pour le choix d'une grille de capteurs. Toutefois, une estimation du champ requiert une optimisation. Dans l'approche régressive, l'opérateur de passage entre l'espace de mesures et l'espace d'état est complètement appris. Née de l'estimation stochastique, l'estimation régressive a de belles perspectives grâce aux méthodes d'apprentissage supervisé. Dans l'approche d'assimilation de données, des mesures sont utilisées pour mettre à jour les prédictions d'un modèle dynamique. Les filtres de Kalman permettent une mise à jour séquentielle des prédictions mais l'algorithme repose sur un modèle dynamique (potentiellement à apprendre, selon les méthodes de la section 1.2.4) et sur la connaissance des incertitudes. C'est la méthode la plus adaptée pour faire de la prédiction continue.

1.5 | Le travail de thèse

Conclusions de l'état de l'art

L'apprentissage automatique est un ensemble d'outils *pour* l'optimisation à partir de données. Son utilisation en mécanique des fluides est une réalité et de nombreux travaux *académiques* fournissent des réponses aux questions de réduction de dimension, de contrôle, de fermeture et de modélisation. Toutefois, les applications restent limitées à des cas d'écoles comme l'écoulement laminaire autour d'un cylindre. Dans le cadre de la mobilité aérienne urbaine, on se demande si les méthodes développées peuvent être utilisés pour estimer un écoulement urbain. C'est un problème complexe car l'écoulement est tridimensionnel et fortement turbulent. Sur des écoulements plus simples, trois stratégies d'estimation ont déjà été testées : l'estimation directe, l'estimation par régression et l'estimation par assimilation de données.

Objectifs du manuscrit

Dans tout le manuscrit, on distingue le problème de *reconstruction* du problème de *prédiction*. Dans le premier cas, il s'agit d'estimer l'état courant du système à partir de mesures courantes. Dans le second cas, il s'agit d'estimer le futur de l'état à partir des états courant et passé. L'objectif du manuscrit est de formuler une stratégie complète d'estimation d'un écoulement, basé sur les trois briques de *réduction*, *reconstruction* et *prédiction*. De cette manière, les différents outils parsemés dans la littérature seront unifiés et reformulés dans un contexte purement orienté données.

Différentes combinaisons de méthodes seront testées sur quatre écoulements de complexité croissante. La première configuration testée est l'écoulement laminaire autour d'un cylindre. Il s'agit du cas de référence pour la validation des méthodes implémentées. La seconde configuration est une couche de mélange spatiale laminaire dont l'intérêt réside dans l'instabilité de Kelvin-Helmotz. La troisième configuration est un écoulement pleinement turbulent autour d'un cylindre rectangulaire. La dernière configuration est un écoulement urbain, qui sera estimé dans différents plans.

Innovations

Plusieurs innovations sont proposées dans le manuscrit. Elles résultent de questionnements qui n'ont pas trouvé de réponse dans la littérature. Les innovations principales sont listées ci-dessous :

1. Pour la réduction, on s'intéresse aux auto-encodeurs variationnels. L'objectif est de régulariser l'espace latent pour gagner en robustesse lors du décodage d'états latents bruités ou mal estimés.
2. Pour la reconstruction, on étudie différentes formes de modèles de l'apprentissage supervisé. Les hyper-paramètres sont systématiquement validés et la robustesse au bruit de mesures est étudiée.

3. Pour la prédiction, on compare différentes approximations de l'opérateur de Koopman. On propose aussi un algorithme pour l'apprentissage des observables.

Deux travaux supplémentaires sont présentés en annexes G et H. Ils seront rapidement référés dans le manuscrit. Il s'agit :

1. D'un article publié dans *Physica D* qui traite de la prédiction continue du système de Lorenz 63. On y apprend un modèle dynamique par réseau de neurones récurent et on met à jour les prédictions par assimilation de données régressive (par opposition à une approche générique comme le filtre de Kalman).
2. D'un article de conférence pour la 3AF. On y explique comment des modèles de reconstruction et de prédiction purement orientés données sont utilisés dans le cadre de l'assimilation de données.

Avant de commencer

Cet état de l'art montre que la thèse est pluridisciplinaire avec un besoin de connaissances en mathématiques appliquées et en mécanique des fluides. Pour cette raison, la seconde partie du manuscrit est purement basée sur des développements mathématiques. **Le lien avec la mécanique des fluides n'est pas explicite car les méthodes sont potentiellement applicables à n'importe quel système**. La troisième partie du manuscrit présente les quatre écoulements de travail et les résultats. De nombreux tableaux de valeurs sont donnés en annexe C et D car ils sont peu interprétables tels quel. L'exposé présente les tendances générales et suit la logique de la partie méthodologique. Il faut aussi préciser que tout le travail s'effectue à des paramètres fixés. Les méthodes génératives pour couvrir différents paramètres (nombre de Reynolds, intensité turbulente, etc.) sortent du cadre de ce manuscrit. La robustesse s'entend comme généralisation des méthodes aux données de test, non observés durant l'apprentissage, mais issus du même calcul numérique. L'ajout de contraintes physiques n'est également pas au programme.

Outils de programmation

Pour produire ce travail, environ 15000 lignes de code sous python 3 ont été écrites. L'apprentissage de tous les modèles a nécessité environ 400 heures CPU. Une utilisation extensive des bibliothèques suivantes a été faite :

— *numpy*. Pour l'algèbre linéaire et la gestion des données.
— *sklearn*. Pour la majeure partie des algorithmes d'apprentissage automatique.
— *keras*. Pour les réseaux de neurones.
— *cvxpy*. Pour les problèmes d'optimisation en reconstruction directe.
— *matplotlib*. Pour les tracés.
— *sherpa*. Pour l'optimisation massive d'hyper-paramètres.
— *scipy*. Pour l'interpolation d'une grille non ordonnée à une grille ordonnée.
— *pickle*. Pour la sérialisation des résultats.
— *torch*. Pour l'algorithme EDMD DL.
— *subprocess*. Pour séquencer les exécutions.

— *statsmodel*. Pour des calculs statistiques.

Concernant les simulations numériques, elles ont été réalisés avec le code volumes finis *OpenFOAM*.

Quelques sources

Les lecteurs qui ne sont pas des mathématiciens appliqués pourront trouver de très bonnes explications sur les chaînes *youtube* suivantes :

— *3B1B*. De très bonnes explications sur l'algèbre linéaire. Très utile pour comprendre la décomposition spectrale et la dualité entre une application linéaire et sa représentation matricielle. Également un cours sur les réseaux de neurones.

— *Stat Quest*. Une excellente chaîne pour l'apprentissage automatique [49].

— *Steve Brunton*. De bonnes explications sur les sujets de recherche évoqués dans cet état de l'art.

— *Fluid mechanics 101*. De bonnes explications pour comprendre la mécanique des fluides numériques.

49. Triple BAM !

Deuxième partie

Méthodes orientées données pour l'estimation d'un système dynamique

2. Les auto-encodeurs pour réduire la dimension du problème

Dans ce chapitre, on s'intéresse à des méthodes de réduction non intrusives pour réduire la dimension d'une base de données. L'objectif est de réécrire l'état de haute dimension à partir de structures *pertinentes* et de rang faible extraites des données. La première section s'intéresse à formaliser ce problème d'auto-encodage. La deuxième section présente l'analyse en composantes principales pour réduire une base de données selon des directions orthogonales. La troisième section est consacrée aux méthodes de réduction par auto-encodeurs neuronaux. On y introduit le réseau de neurones et différentes formes d'auto-encodeurs, dont l'auto-encodeur variationnel. Enfin, on fait la synthèse des méthodes qui seront utilisées dans la troisième partie du manuscrit et on précise les métriques pour quantifier la qualité d'un auto-encodage.

Sommaire

2.1 | Le problème de réduction de dimension

Dans cette section, on introduit la problématique d'auto-encodage et on précise les notations pour le reste du chapitre.

2.1.1 | La compressibilité des signaux

Avant de formaliser le problème de réduction, on essaye de comprendre pourquoi les signaux sont compressibles. Pour cela, on considère une image de 20×20 pixels noirs et blancs. Au total, il y a 2^{400} images possibles, ce qui est plus grand que le nombre de nucléons dans l'univers observable [1]. Dans l'espace des images d'un mégapixel et sans limite de couleurs, le nombre d'images possibles est encore plus grand. La dimension est $n = 10^6$ car une image quelconque est canoniquement reconstruite en superposant un million d'images constituées chacune d'un seul pixel coloré différent. À la fin d'une superposition aléatoire, les chances que l'image constituée soit naturelle sont minces : on observe vraisemblablement du bruit [2]. Autrement dit, beaucoup de directions sont redondantes et pour construire une image naturelle, il aurait été plus judicieux de superposer quelques images *principales* non définies par un seul pixel. Si l'on transpose à la physique, l'état du système est décrit par un vecteur de $\mathbb{R}^n$. Plutôt que d'écrire l'état sous la forme $\sum_{i=1}^{n} x_i \boldsymbol{e}_i$ avec $\boldsymbol{e}_i$ la i-ième direction canonique de $\mathbb{R}^n$, on approxime l'état par $\sum_{i=1}^{r} a_i \boldsymbol{\phi}_i$ où les modes $\boldsymbol{\phi}_i$ sont $r \ll n$ vecteurs de $\mathbb{R}^n$ décrivant *optimalement* l'espace. Définir cette optimalité et calculer les directions propres, c'est l'objectif de la réduction.

2.1.2 | Formalisation de la réduction

Soit un état de haute dimension $x \in \mathbb{R}^n$. On cherche à auto-encoder cet état c'est-à-dire *l'encoder* puis le *décoder*. Soit une famille d'encodeurs E et une famille de décodeurs D. L'auto-encodeur optimal est défini par la paire :

$$(e, d) = \underset{(e,d)\in E\times D}{\arg\min}\ \epsilon\left[x, d \circ e(x)\right]$$

où ϵ est une fonction coût qui mesure l'écart entre l'état réel et l'état auto-encodé. Tout l'enjeu consiste à bien choisir les candidats (familles E et D) et la fonction coût ϵ pour maximiser l'information lors de l'encodage et minimiser l'erreur lors du décodage.

1. Grâce à des calculs de masse, on estime qu'il y a 10^{80} nucléons dans l'univers observable.
2. Cette idée était déjà introduite en 1944 avec la nouvelle *La Bibliothèque de Babel* de Borges. L'auteur décrit une bibliothèque immense avec tous les livres de 410 pages possibles, chaque page étant formée de 40 lignes d'environ 80 caractères. Une partie infime de la bibliothèque contient tous les ouvrages déjà écrits et ceux à venir. Les autres sont complètement illisibles.

2.1.3 | Notations

Dans le contexte de l'apprentissage automatique, le problème d'optimisation est résolu à partir de données non étiquetées (procédure non supervisée). Suite à des observations, on dispose d'une matrice de clichés $u \in \mathbb{R}^{n\times m}$ avec n la dimension de l'état et m le nombre d'échantillons. Les clichés sont rangés par colonnes et on les suppose ordonnés dans le cas d'un système dynamique. La matrice est également **centrée** c'est-à-dire que la moyenne de tous les clichés donne le vecteur nul.

L'encodeur et le décodeur sont appris avec une première partie des données que l'on qualifie *d'apprentissage* ou *d'entraînement*. L'autre partie forme l'ensemble de *test* et sert à vérifier que l'auto-encodeur se généralise à de nouvelles données. Pour un découpage 70/30, on définit les matrices $u^{\text{train}} \in \mathbb{R}^{n\times m_{\text{train}}}$ et $u^{\text{test}} \in \mathbb{R}^{n\times m_{\text{test}}}$ avec $m_{\text{train}} = 0.7m$ et $m_{\text{test}} = 0.3m$. Un état quelconque est noté $u_{[:,t]} \in \mathbb{R}^n$ où l'indice $[:,t]$ précise que l'on travaille avec toutes les composantes d'un seul cliché.

Dans la suite, on présente deux approches classiques : l'analyse en composantes principales et les auto-encodeurs neuronaux. On profitera de la seconde approche pour donner une présentation générale des réseaux connexionnistes.

2.2 | L'analyse en composantes principales

Dans cette section, on présente l'analyse en composantes principales (PCA). L'objectif est de déterminer une base orthogonale dans laquelle les données sont décorrélées. En se concentrant sur les directions de variance maximale, on sacrifie une partie de l'énergie mais on dispose d'une vision de faible rang des données. Ici, trois approches sont présentées : la décomposition spectrale de la matrice de covariance, les quotients de Rayleigh et la décomposition en valeurs singulières (SVD) de la matrice des clichés.

2.2.1 | Décomposition spectrale de la matrice de covariance

Décorréler les données d'entraînement revient à diagonaliser la matrice de covariance (Taira *et al.*, 2017). Il s'agit d'une matrice $n \times n$ dont le coefficient (i, j) indique la covariance [3] entre le signal i et le signal j. Elle est définie par :

$$C^{\text{train}} = u^{\text{train}}[u^{\text{train}}]^T$$

La matrice de covariance est vraisemblablement pleine dans la base canonique de $\mathbb{R}^n$. La décomposition spectrale supprime les moments croisés d'ordre deux et s'écrit :

3. Une covariance positive indique qu'une augmentation du premier signal est synonyme d'une augmentation du deuxième signal *en moyenne*. Une covariance négative indique que les tendances sont opposées.

$$C^{\text{train}}\Phi = \Phi\Lambda$$

avec $\Phi \in \mathbb{R}^{n\times n}$ la matrice des vecteurs propres et $\Lambda \in \mathbb{R}^{n\times n}$ la matrice diagonale des valeurs propres. La matrice de covariance étant définie positive, le théorème spectral indique que les valeurs propres sont positives et que les vecteurs propres sont orthogonaux [4]. On décide de classer les valeurs propres par ordre décroissant c'est-à-dire :

$$\Lambda_{11} \geq \Lambda_{22} \geq ... \geq \Lambda_{nn}$$

Les vecteurs propres forment une base de $\mathbb{R}^n$. Dans cette base dite *principale*, les données sont décorrélées et la variance dans la direction $\Phi_{[:,i]}$ est Λ_{ii}. Un nouvel état $u_{[:,t]}$ s'écrit $u'_{[:,t]}$ avec $u_{[:,t]} = \Phi u'_{[:,t]}$. La matrice Φ étant unitaire, son inverse est son adjoint donc on a $u'_{[:,t]} = \Phi^T u_{[:,t]}$.

On tronque la transformation aux r premières directions. On définit la matrice $\Phi_r \in \mathbb{R}^{n\times r}$ à partir des r premières directions principales. Les r premières composantes de $u'_{[:,t]}$ forment *l'état latent* que l'on note $a_{[:,t]} \in \mathbb{R}^r$. Cela définit l'opération d'encodage :

$$\textbf{Encodage} \rightarrow a_{[:,t]} = e\left(u_{[:,t]}\right) = [\Phi_r]^T u_{[:,t]}$$

En écrivant l'état latent dans l'espace de dimension n, on obtient une approximation de rang r de l'état. Cela définit l'opération de décodage :

$$\textbf{Décodage} \rightarrow \hat{u}_{[:,t]} = d\left(a_{[:,t]}\right) = \Phi_r a_{[:,t]} = \Phi_r [\Phi_r]^T u_{[:,t]}$$

avec $\hat{u}_{[:,t]}$ l'approximation de rang r de l'état $u_{[:,t]}$. En considérant $r = n$, la condition d'orthogonalité permet d'écrire $\Phi_r[\Phi_r]^T = I_{n\times n}$ et l'on retrouve $\hat{u}_{[:,t]} = u_{[:,t]}$. Dans le cas $r \leq n$, seule la condition $\Phi_r^T[\Phi_r] = I_{r\times r}$ est acquise. L'énergie cumulative restituée par les r directions est calculée avec :

$$E(r) = \frac{\sum_{i=1}^{r} \Lambda_{ii}}{\sum_{i=1}^{n} \Lambda_{ii}}$$

ce qui permet de quantifier l'information perdue par l'approximation de rang r. La figure 2.1 donne un exemple d'analyse en composantes principales dans le cas $n = 2$.

4. Soit S une matrice rectangulaire. Alors SS^T est une matrice carrée symétrique, comme la matrice de covariance. On note v un vecteur propre de SS^T associé à la valeur propre λ. On a $SS^T v = \lambda v$ donc $v^T SS^T v = \lambda v^T v$ d'où $||S^T v||_2^2 = \lambda ||v||_2^2$ donc $\lambda \geq 0$. Si maintenant on considère les couples (λ_i, v_i) et (λ_j, v_j), on peut écrire $v_j^T SS^T v_i = v_j^T \lambda_i v_i$ donc $(\lambda_j - \lambda_i) v_j^T v_i = 0$. Cela indique que des valeurs propres différentes impliquent des vecteurs propres orthogonaux. Cette démonstration fait apparaître les termes du quotient de Rayleigh, que l'on retrouve dans la formulation historique de la PCA.

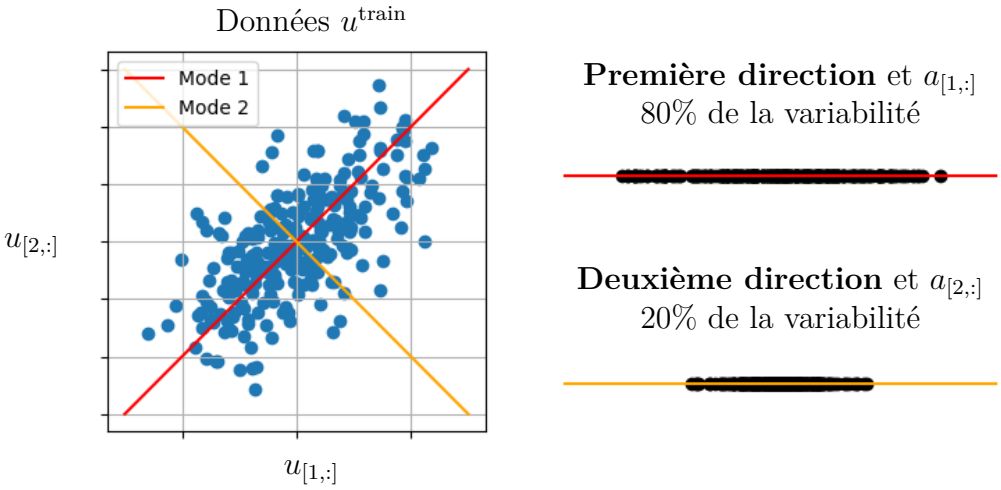


Figure *2.1* – *Illustration de l'analyse en composantes principales. Plutôt que de travailler avec les directions canoniques, on travaille avec la base orthonormée qui restitue hiérarchiquement la variance. La première composante principale représente 80% de la variabilité totale. La seconde composante principale restitue la variabilité résiduelle.*

2.2.2 | Lien avec les quotients de Rayleigh

Historiquement, la PCA a été introduite en lien avec des quotients de Rayleigh (Hotelling, 1933). Ici, les directions sont construites hiérarchiquement selon un critère énergétique. La première direction doit maximiser la variance des données projetées sur cette direction. Mathématiquement, il faut résoudre le problème d'optimisation :

$$\Phi_{[:,1]} = \underset{\phi \in \mathbb{R}^n}{\arg\max} \left\{ \frac{\phi^T C^{\text{train}} \phi}{\phi^T \phi} \right\}$$

Les autres directions doivent maximiser la variance des données résiduelles. À l'étape k du processus, on retire la covariance déjà résolue et on construit la matrice :

$$C_k^{\text{train}} = X - \sum_{i_r=1}^{k-1} \Phi_{[:,i_r]} \left[\Phi_{[:,i_r]} \right]^T X$$

On cherche alors la direction qui maximise la variance des données résiduelles projetées :

$$\Phi_{[:,k]} = \underset{\phi \in \mathbb{R}^n}{\arg\max} \left\{ \frac{\phi^T C_k^{\text{train}} \phi}{\phi^T \phi} \right\}$$

Pour la direction k, la valeur maximale du quotient est Λ_{kk} et on l'obtient avec le vecteur propre unitaire associé.

2.2.3 | Décomposition en valeurs singulières

La matrice des clichés d'entraînement admet une décomposition unique (dite en *valeurs singulières*) sous la forme :

$$\textbf{SVD} \rightarrow u^{\text{train}} = U\Sigma V^*$$

avec $U \in \mathbb{C}^{n\times n}$, $\Sigma \in \mathbb{C}^{n\times m}$ et $V \in \mathbb{C}^{m\times m}$. Les matrices U et V sont des matrices unitaires [5]. Dans le cas classique où la dimension est plus grande que le nombre de données, la décomposition se réécrit :

$$u^{\text{train}} = \begin{bmatrix}\hat{U} & \hat{U}^{\perp}\end{bmatrix} \begin{bmatrix}\hat{\Sigma} \\ 0_{(n-m)\times m}\end{bmatrix} V^*$$

avec $\hat{U} \in \mathbb{C}^{n\times m}$, $\hat{U}^{\perp} \in \mathbb{C}^{n\times(n-m)}$ et $\hat{\Sigma} \in \mathbb{C}^{m\times m}$. Les colonnes de $\hat{U}^{\perp}$ décrivent un espace orthogonal et supplémentaire aux colonnes de $\hat{U}$. La matrice $\hat{\Sigma}$ est diagonale et ses composantes sont appelées *valeurs singulières*. Selon Eckart et Young (1936), l'approximation optimale de rang r (au sens des moindres carrés) de u^{train} est la troncature au rang r de sa décomposition en valeurs singulières. Autrement dit :

$$\widetilde{U}\widetilde{\Sigma}\widetilde{V}^* = \underset{\substack{L \text{ tel que} \\ \text{rg}(L)=r}}{\arg\min} ||u^{\text{train}} - L||_F^2$$

où $||.||_F^2$ est le carré de la norme de Frobenius [6] et permet le calcul de la somme quadratique des résidus. Les matrices $\widetilde{U} \in \mathbb{C}^{n\times r}$, $\widetilde{\Sigma} \in \mathbb{C}^{r\times r}$ et $\widetilde{V} \in \mathbb{C}^{r\times m}$ sont respectivement les r premières colonnes de $\hat{U}$, les r premières valeurs singulières et les r premières lignes de V.

La décomposition en valeurs singulières permet de retrouver la décomposition spectrale de la matrice de covariance. Pour les corrélations temporelles, on peut écrire :

$$u^{\text{train}}\left[u^{\text{train}}\right]^* = U \begin{bmatrix}\hat{\Sigma}^2 & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & 0_{(n-m)\times(n-m)}\end{bmatrix} U^*$$

Cela permet d'identifier U aux directions principales Φ et $\hat{\Sigma}^2$ aux valeurs propres Λ. Si le but final est de construire une approximation de rang r de la matrice des clichés, faire la décomposition spectrale n'est pas idéal : le coût de calcul pour diagonaliser la matrice $n \times n$ est important. La *méthode des clichés* (ou

5. $UU^* = U^*U = I_{n\times n}$. De telles matrices sont appréciées car elles agissent comme des isométries linéaires et conservent les produits scalaires. Les matrices de rotation sont des matrices unitaires.

6. Pour la matrice M, on a $||M||_F = \sqrt{\sum_{i,j} M_{[i,j]}^2}$

des *snapshots* en anglais) est plus adaptée : on calcule la matrice de covariance spatiale $[u^{\text{train}}]^* u^{\text{train}}$, on effectue la décomposition spectrale pour avoir les valeurs propres $\hat{\Sigma}^2$ et les vecteurs propres V puis on récupère les r premières directions principales via $\widetilde{U} = u^{\text{train}} \widetilde{V} \hat{\Sigma}^{-1}$ (Sirovich, 1987). À noter qu'un critère numérique pour déterminer r existe mais nous ne rentrons pas dans les détails (Gavish et Donoho, 2014)

Interprétation

Les colonnes de U forment une base orthonormale pour l'espace des colonnes de u^{train}. Les colonnes de V forment une base orthonormale pour l'espace des lignes de u^{train}. Autrement dit, les colonnes de U représentent les **directions spatiales dominantes** des clichés d'entraînement et les colonnes de V représentent les **directions temporelles dominantes**. Supposons que u^{train} soit la représentation matricielle d'une application linéaire de $\mathbb{R}^m$ dans $\mathbb{R}^n$ munis de leurs bases canoniques. La décomposition en valeurs singulières permet d'écrire cette transformation comme la composée d'une rotation (via V^*) avec des étirements/compressions (via Σ) et une nouvelle rotation (via U). Pour une interprétation en mécanique des fluides, on renvoie à la note de bas de page [7].

2.2.4 | Synthèse

Dans l'analyse en composantes principales, le décodeur est une matrice rectangulaire dont les colonnes (appelés modes) sont orthogonales. L'encodeur se déduit en transposant cette matrice. Les modes peuvent être construits par décomposition spectrale de la matrice de covariance, par maximisation de quotients de Rayleigh ou par décomposition en valeurs singulières de la matrice

7. Supposons que la matrice des clichés corresponde à des champs de vitesse. D'un point de vue application linéaire, l'espace de départ est $\mathbb{R}^m$ de sorte qu'une direction canonique est un instant donné. L'espace d'arrivée est $\mathbb{R}^n$ de sorte qu'une direction canonique est un champ de vitesse nul partout sauf en un point. La j-ième colonne de u^{train} indique comment combiner les champs de vitesse canoniques (définis par un pixel) pour retrouver l'observation du champ de vitesse à l'instant j. L'opération est linéaire : le cliché complet est une superposition de calques plus ou moins colorés des champs canoniques, les couleurs étant les composantes lues dans le vecteur colonne. Procéder ainsi pour refaire le film de l'écoulement n'est pas judicieux : les images sont reconstruites dans l'ordre et chaque image requiert de superposer n calques peu interprétables pour visualiser le cliché. La SVD rend la tâche moins fastidieuse. Les colonnes de V indiquent les combinaisons de clichés à reconstruire pour ne pas reconstruire des clichés redondants. Par exemple, il faut reconstruire simultanément la superposition des clichés 7, 12 et 14. Dans le nouveau système de temps, les variations spatiales d'un cliché ne sont pas corrélées avec les variations spatiales des autres clichés. La matrice U indique les calques à utiliser pour les reconstructions. Il ne s'agit plus de superposer pixels par pixels mais plutôt de superposer des structures cohérentes qui permettront de visualiser rapidement le cliché. Plus on ajoute de calques, plus la résolution est meilleure mais quelques calques suffisent pour restituer une grande partie de l'énergie. Dans le nouveau système d'espace, les variations temporelles d'un calque ne sont pas corrélées avec les variations temporelles d'un autre calque. Finalement, les clichés intéressants $u^{\text{train}} V$ sont reconstruits selon $U\Sigma$. Mais dans un monde où le temps est fléché, reconstruire les clichés $u^{\text{train}} V$ a peu de sens. On reconstruit plutôt les clichés dans le sens normal du temps, en conservant toutefois l'idée des calques. Mathématiquement, on effectue $u^{\text{train}} = U\Sigma V^*$. Dans ce cas, ΣV^* contient les couleurs à utiliser pour chaque calque afin de reconstruire les données u^{train}.

des clichés. Dans tous les cas, les directions principales sont des directions de variance maximale et la troncature au rang r est la meilleure approximation possible au sens des moindres carrés. De par la simplicité de son implémentation [8], c'est généralement la première méthode de réduction testée.

Remarque 1 : dans le cadre de la mécanique des fluides, la méthode sera plutôt appelée POD pour *Proper Orthogonal Decomposition*.

Remarque 2 : il existe une formulation continue de l'analyse en composantes principales, souvent appelée *décomposition de Karhunen Loève*. Cette approche est surtout utile pour construire des modèles réduits intrusifs (Rowley et Dawson, 2017; Noack *et al.*, 2003).

2.3 | Réduction par réseau de neurones

L'analyse en composantes principales est une méthode de réduction linéaire et orthogonale. Elle n'est donc pas adaptée à certaines bases de données où les tendances principales sont non linéaires [9] comme illustré sur la figure 2.2. Ici, on s'intéresse aux auto-encodeurs neuronaux qui permettent une réduction aussi *flexible* que souhaitée en jouant sur l'architecture des réseaux de neurones. Dans la suite, on donne une présentation générale des réseaux connexionnistes et on illustre le théorème d'universalité. Ensuite, on fait le lien entre l'auto-encodeur linéaire et l'analyse en composantes principales puis on développe les auto-encodeurs variationnels.

2.3.1 | Présentation générale des réseaux connexionnistes

Un réseau de neurones est un ensemble d'unités interconnectés. Chaque unité, appelée *neurone formel*, réagit à un stimulus via une fonction d'activation comme illustré sur la figure 2.3. Les neurones sont regroupés en couches et on distingue la couche d'entrée (excitation initiale), les couches cachées (traitement) et la couche de sortie (réponse). Selon l'organisation des couches et le type d'activation, le réseau de neurones prend un nom différent et peut répondre à des tâches spécifiques (voir page 222 de Brunton et Kutz (2019) pour des détails pédagogiques). Dans ce manuscrit, on travaillera uniquement avec des **réseaux de neurones à propagation avant** où l'information est propagée de la couche d'entrée vers la couche de sortie sans retour possible. Avec un nombre important de couches (une dizaine), le réseau est qualifié de *profond*. Cette technologie est à la base de l'intelligence artificielle (LeCun *et al.*, 2015).

8. Une ligne de code en python : $u, s, vh =$ numpy.linalg.svd(u^{train})
9. On renvoit au tutoriel Shlens (2014) pour toutes les limitations de l'analyse en composantes principales.

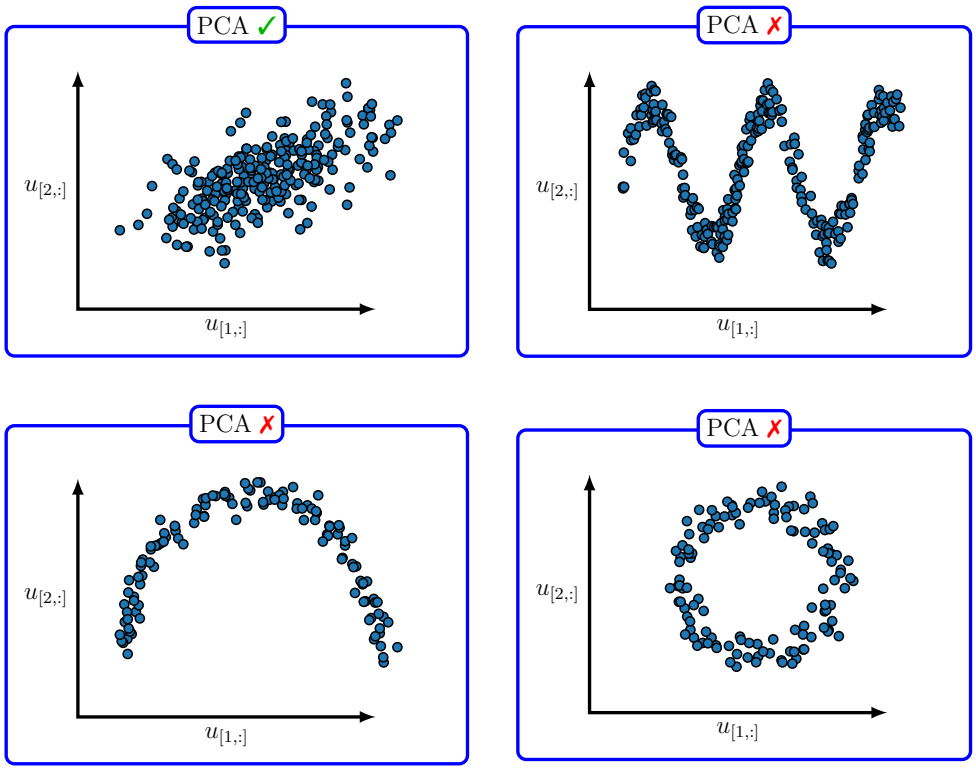
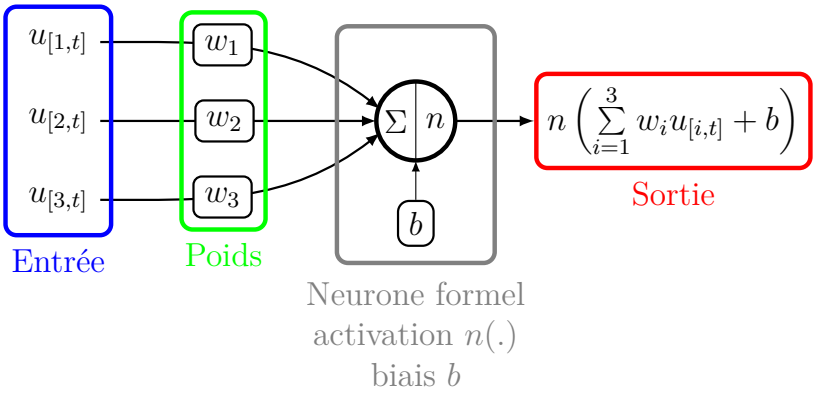


Figure 2.2 – *Exemples prototypes où l'analyse en composantes principales est adaptée ou échoue.*

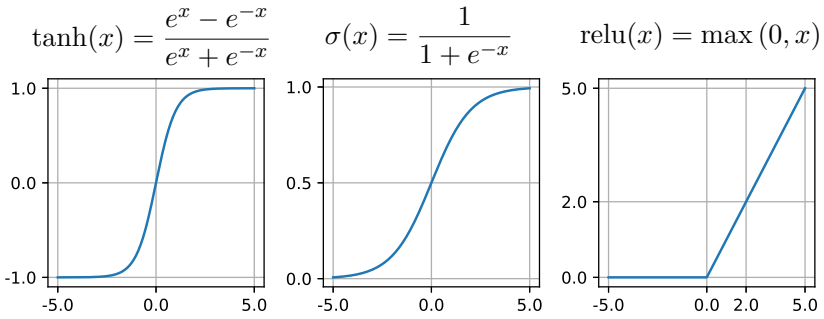
Le théorème d'approximation universelle

Un réseau de neurones à propagation avant avec une seule couche cachée et un nombre suffisant de neurones sigmoïdaux peut approximer *n'importe quelle fonction*. C'est le théorème d'approximation universel dit de Cybenko (1989) et généralisé à d'autres fonctions d'activations par Hornik *et al.* (1989). Ce résultat est assez intuitif pour une fonction de $\mathbb{R}$ dans $\mathbb{R}$. Sur la figure 2.4, on considère un réseau à propagation avant avec un neurone dans la couche d'entrée, deux neurones dans la couche cachée et un neurone dans la couche de sortie. Les neurones cachés s'activent selon une sigmoïde tandis que le neurone de sortie s'active linéairement. La sortie est une combinaison linéaire des deux sigmoïdes dont les positions sont déterminées par les paramètres (poids et biais) du réseau. En augmentant le nombre de neurones cachés (donc le nombre de paramètres), on augmente le nombre de sigmoïdes et les non linéarités dans la sortie. Le théorème généralise cette observation aux entrées et sorties multi-dimensionnelles.

Le théorème d'approximation universelle fait des réseaux à propagation avant les candidats idéaux pour les problèmes d'apprentissage supervisé. Dans ce qui suit, on utilise les notations classiques pour ce type de problèmes ; on reviendra aux notations standards du manuscrit dans les sections suivantes. Ici, on dispose d'un



(a) Exemple de neurone formel. L'unité effectue pondère une somme d'entrées (Σ) et s'active plus ou moins en fonction de l'excitation.



(b) Exemples classiques de fonction d'activation. De gauche à droite : tangente hyperbolique, sigmoïde, fonction linéaire rectifiée

Figure *2.3* – *Illustration du neurone formel et exemples d'activations.*

ensemble de m données étiquetées $\{\boldsymbol{x}_k, \boldsymbol{y}_k\}_{k\in[1..m]}$ où $\boldsymbol{x}_k$ et $\boldsymbol{y}_k$ sont respectivement le k-ième vecteur d'excitation et le k-ième vecteur de sortie. L'objectif est d'apprendre les poids et biais d'un réseau pour estimer le champ $\boldsymbol{f}$ tel que $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. Les paramètres W sont optimisés en minimisant un coût $\mathcal{L}$ qui évalue l'écart entre les sorties d'apprentissage et les estimations par le réseau. Traditionnellement, le coût est l'erreur quadratique et se calcule selon :

$$\mathcal{L}\left(W, \{\boldsymbol{x}_k, \boldsymbol{y}_k\}_{[1..m]}\right) = \sum_{k=1}^{n} ||\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k||_2^2$$

où $||.||_2$ est la norme l_2 (somme des composantes au carré) et $\hat{\boldsymbol{y}}_k$ est la sortie calculée avec $\boldsymbol{x}_k$ en entrée et les paramètres courants. Pour minimiser le coût, une descente de gradient est utilisée. Celle-ci nécessite de calculer le gradient de $\mathcal{L}$ par rapport aux poids et au biais. Pour des réseaux de neurones profonds où l'expression analytique du coût est complexe, le calcul des gradients n'est pas trivial. Heureusement, l'algorithme de rétro-propagation fut développé par Rumelhart *et al.*

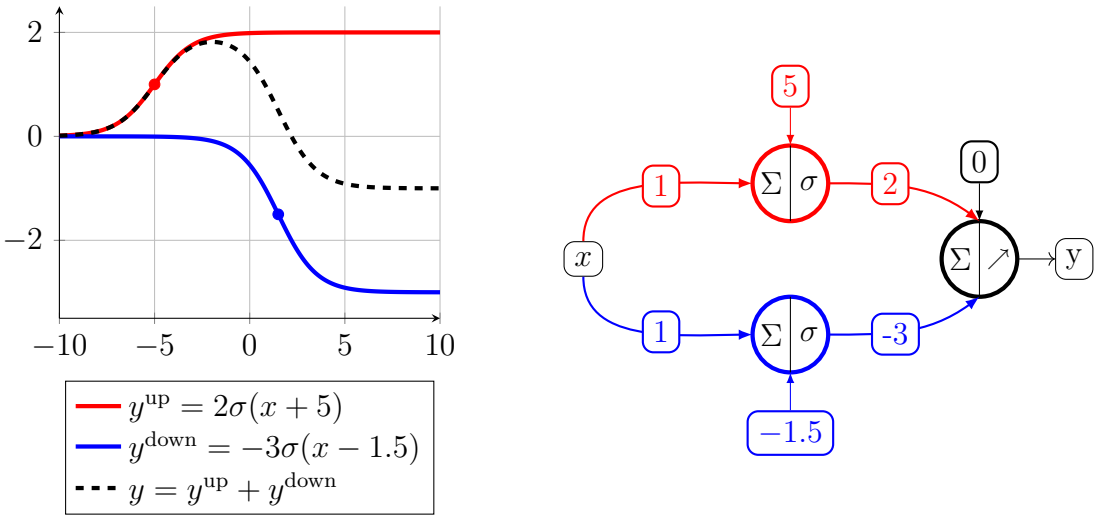


Figure 2.4 – *Illustration d'un réseau de neurones simple. La branche haute positionne une sigmoïde d'amplitude* 2 *en* $x = -5$. *La branche basse positionne une sigmoïde d'amplitude* 3 *en* $x = 1.5$. *La sortie est la somme de ces sigmoïdes. Avec plus de neurones cachés, on peut reconstruire n'importe quel fonction.*

(1986).

Algorithme de rétro-propagation

Soit un réseau de neurones à propagation avant. Un échantillon d'entraînement est noté $(\boldsymbol{x}_k, \boldsymbol{y}_k)$. Au niveau de la couche l, le vecteur d'entrées $[\boldsymbol{i}^l]_k$ est traité par les neurones cachés selon l'activation n_l identique pour chaque neurone. Le vecteur traité est noté $[\boldsymbol{o}^l]_k$. À la dernière couche L, le coût pour l'échantillon est $\mathcal{L}(\boldsymbol{x_k}, \boldsymbol{y}_k)$ que l'on note $\mathcal{L}_k$. La contribution du neurone j de la couche l à l'erreur $\mathcal{L}_k$ est notée $[\delta_j^l]_k$. La matrice des poids entre la couche $l-1$ et l est notée $\boldsymbol{w}^l$. Le vecteur des biais des neurones de la couche l est noté $\boldsymbol{b}^l$. L'algorithme de rétro-propagation permet de calculer le gradient de $\mathcal{L}_k$ par rapport aux paramètres $\boldsymbol{w}^l$ et $\boldsymbol{b}^l$. En utilisant extensivement la règle de dérivation en chaîne, on établit l'algorithme suivant :

Algorithme de rétro-propagation

1. **Activation linéaire pour la couche d'entrée** :

$$[\boldsymbol{o}^l]_k = \boldsymbol{x}_k$$

2. **Propagation avant** :

$$\forall l : 1 \to L \text{ calcul de } [\boldsymbol{o}^l]_k = n_l\left([\boldsymbol{i}^l]_k\right) = n_l\left(\boldsymbol{w}^l[\boldsymbol{o}^{l-1}]_k + \boldsymbol{b}_l\right)$$

3. **Calcul des erreurs dans la dernière couche** :

$$[\boldsymbol{\delta}^L]_k = \frac{\partial \mathcal{L}_k}{\partial\,[\boldsymbol{i}^L]} = \nabla_o \mathcal{L}_k \odot n'_L\left([\boldsymbol{i}^L]_k\right)$$

L'opérateur $\odot$ est le produit d'Hadamard, ∇_o est le gradient par rapport à la sortie $\boldsymbol{o}^L$ et n'_L est la dérivée de l'activation de sortie.

4. **Rétro-propagation de l'erreur** :

$$\forall l : L-1 \to 1 \text{ calcul de } [\boldsymbol{\delta}^l]_k = \left(\left[\boldsymbol{w}^{l+1}\right]^T [\boldsymbol{\delta}^{l+1}]_k\right) \odot n'_l\left([\boldsymbol{i}^l]_k\right)$$

5. **Calcul des gradients** :

$$\frac{\partial \mathcal{L}_k}{\partial w^l_{ji}} = [o_i]^{l-1}_k [\delta^l_j]_k \qquad \text{et} \qquad \frac{\partial \mathcal{L}_k}{\partial b^l_j} = [\delta^l_j]_k$$

avec w^l_{ji} le poids entre le neurone i de la couche $l-1$ et le neurone j de la couche l et $[o_i]^{l-1}_k$ le traitement de $[\boldsymbol{i}]^{l-1}_k$ par le neurone i de la couche $l-1$.

Une fois les gradients calculé, la mise à jour des paramètres $W = \{\boldsymbol{w}^l, \boldsymbol{b}^l\}_{l\in[1..L]}$ s'écrit synthétiquement :

$$\boxed{W \leftarrow W - \eta \frac{\partial \mathcal{L}_k}{\partial W}}$$

avec η la vitesse d'apprentissage [10]. Dans ce cas précis où une descente est effectuée pour chaque nouvel échantillon, la descente est dite *stochastique*. Pour gagner en stabilité, la fonction coût est plutôt évaluée sur des groupes de données appelés *batch*. Traditionnellement, on choisit des batchs dont la taille est une puissance de deux car certaines expériences ont montré que ça pouvait accélérer le calcul, à l'image de la transformée de Fourier rapide. Lorsque tous les batchs ont été utilisés, on dit qu'un *epoch* d'apprentissage a été effectué (Ruder, 2016).

Remarque 1 → pour un coût quadratique $\mathcal{L}_k = \frac{1}{2}||\boldsymbol{y}_k - [\boldsymbol{o}^L]_k||^2_2$, l'étape 3 de l'algorithme se réécrit $[\boldsymbol{\delta}^L]_k = \left(\boldsymbol{y}_k - [\boldsymbol{o}^L]_k\right) \odot n'_L\left([\boldsymbol{i}^L]_k\right)$.

Remarque 2 → les paramètres $\boldsymbol{w}^l$ sont souvent initialisés par un schéma de

10. Si η est élevé, la descente est brutale et il y a risque de divergence. Si η est petit, la descente est lente mais plus stable.

Glorot i.e. ils sont échantillonnés d'une gaussienne de moyenne nulle et de variance fonction du nombre de liaisons entre la couche l et la couche $l+1$ (Glorot et Bengio, 2010).

Remarque 3 → pour des réseaux plus complexes que les réseaux à propagation avant, la rétro-propagation est reformulée. Pour les réseaux récurrents, on parle de rétro-propagation dans le temps (Salehinejad *et al.*, 2017).

Remarque 4 → pour des réseaux profonds, le gradient peut exploser ou disparaître avec les fonctions d'activations classiques. C'est ce qui explique le développement de la fonction linéaire rectifiée ou des cellules mémoire pour les réseaux récurrents (Hochreiter, 1998).

Remarque 5 → d'un point de vue machine, l'architecture du réseau ne doit pas être définie avec des objets statiques comme des tableaux dont la dimension est fixée. En travaillant avec des objets dynamiques (taille fantôme), on s'autorise à ne pas charger toutes les données en mémoire. Dans la communauté d'intelligence artificielle, ces objets sont appelés *tenseurs*, d'où le nom des bibliothèques classiques *TensorFlow* et *pyTorch*.

Quelques précisions

Dans le manuscrit, on applique les réseaux de neurones pour auto-encoder l'état $u_{[:,t]}$ de haute dimension. Pour les descentes de gradient, on utilisera l'agorithme *ADAptive Momentum* (Kingma et Ba, 2014) qui met à jour η en fonction des changements de direction du gradient [11]. Concernant l'arrêt de l'apprentissage, on introduit l'ensemble de *validation*. Il correspond aux vingt derniers pourcents de l'ensemble d'entraînement et sert à estimer l'erreur de test. Lorsque le coût de validation raugmente ou évolue peu pendant plusieurs epochs, l'apprentissage est stoppé. On aura l'occasion de reparler de cet ensemble lors de la validation croisée des méthodes de reconstruction. Pour ce chapitre, les hyper-paramètres des réseaux ne seront pas validés et l'ensemble de validation ne sert qu'à contrôler l'apprentissage.

2.3.2 | L'auto encodeur linéaire

L'auto-encodeur linéaire est un réseau de neurones avec une seule couche cachée (r neurones) et dont toutes les fonctions d'activations sont linéaires. On appelle ce réseau LAE pour *Linear Auto Encoder*.

11. Tant que le gradient est dans la même direction que ceux précédents, on accélère la vitesse d'apprentissage. Cela fonctionne comme un rocher dans la pente d'une montagne : sa vitesse augmente s'il roule dans la même direction mais s'annule lorsqu'il rebrousse chemin. C'est la notion de quantité de mouvement ou *momentum*.

Fonction coût

Les poids d'encodage et de décodage sont respectivement notés $W_e \in \mathbb{R}^{n\times r}$ et $W_d \in \mathbb{R}^{r\times n}$. Les neurones cachés sont biaisés avec le vecteur $b_h \in \mathbb{R}^r$ tandis que les neurones de sortie sont biaisés avec le vecteur $b_o \in \mathbb{R}^n$. L'ensemble de ces paramètres permet de définir les opérations :

$$\textbf{Encodage} \rightarrow a_{[:,t]} = e\left(u_{[:,t]}\right) = W_e u_{[:,t]} + b_h$$

$$\textbf{Décodage} \rightarrow \hat{u}_{[:,t]} = d\left(a_{[:,t]}\right) = W_d a_{[:,t]} + b_o$$

Les poids et les biais sont obtenus par calcul itératif, en minimisant l'erreur quadratique entre les états réels u^{train} et leur auto-encodage $\hat{u}^{\text{train}}$. Le coût pour un échantillon est définie par :

$$\epsilon = ||u_{[:,t]} - \hat{u}_{[:,t]}||_2^2 = \sum_{i_n=1}^{n} \left(u_{[i_n,t]} - \hat{u}_{[i_n,t]}\right)^2$$

Avec ce choix de fonction coût, un lien fort existe avec l'analyse en composantes principales. On résume les résultats ci-dessous.

Lien avec l'analyse en composantes principales

Supposons que les biais soient nuls. Dans ce cas, l'objectif est de trouver les poids W_e et W_d pour que le coût total soit minimal. En introduisant la norme de Frobenius, on peut compacter la notation :

$$\mathcal{L}(W_e, W_d) = ||u^{\text{train}} - W_d W_e u^{\text{train}}||_F^2$$

On suppose la matrice W_d connue. On peut alors montrer que les poids d'encodage satisfont :

$$W_e = [W_d]^\dagger = \left[W_d^T W_d\right]^{-1} W_d^T$$

où l'opération † est l'inverse généralisé de Moore Penrose. Elle traduit que pour un nouvel état $u_{[:,t]}$, l'auto-encodage optimal est obtenu par projection orthogonale sur le sous espace des colonnes de W_d :

$$\hat{u}_{[:,t]} = W_d \left[W_d^T W_d\right]^{-1} W_d^T u_{[:,t]}$$

Dans le cas où les colonnes du décodeur sont orthogonales $\left(W_d^T W_d = I_{n\times n}\right)$, on retombe sur la formulation de l'analyse en composantes principales. Ces résultats restent valides en ajoutant les biais des neurones dans le problème d'optimisation (Plaut, 2018).

2.3.3 | L'auto-encodeur variationnel

Pour estimer l'état de haute dimension, une stratégie possible est 1) estimer l'état latent de dimension réduite et 2) utiliser le décodeur. On parle d'approche *générative* car on essaye de générer l'estimation $\hat{u}_{[:,t]}$ à partir d'un point de l'espace latent. Le décodeur étant appris à partir de données, deux configurations sont possibles.

1. L'estimation de l'état latent est *exactement* identique à un état rencontré lors de l'apprentissage de l'auto-encodeur. Dans ce cas, il suffit de répéter l'exemple pour avoir une estimation pertinente de l'état de haute dimension.
2. L'estimation de l'état latent est *différente* de tous les états rencontrés lors de l'apprentissage de l'auto-encodeur. Dans ce cas, l'estimation de l'état de haute dimension sera vraisemblablement étrange si le décodeur n'est pas robuste.

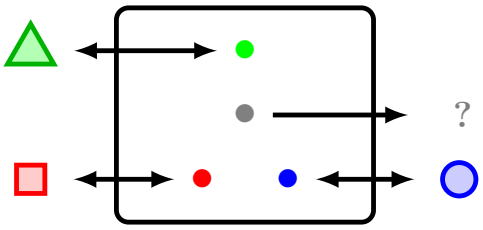
Pour améliorer la robustesse du décodeur, il faut régulariser l'apprentissage de l'auto-encodeur et *structurer* l'espace latent. Mathématiquement, on doit s'assurer que l'espace appris soit *continu* et *complet*. La première propriété traduit que deux états proches dans l'espace latent doivent être décodés de manière similaire. La seconde propriété traduit qu'un échantillon de l'espace latent doit être décodé en une donnée qui fait sens pour le système considéré. Une stratégie possible que nous allons étudier est l'auto-encodeur variationnel où l'encodeur et le décodeur sont des objets *probabilistes*. Les détails mathématiques sont donnés dans l'annexe A.

Intuition de l'auto-encodeur variationnel

Pour structurer l'espace latent, les états d'apprentissage sont encodées en *distributions* plutôt qu'en *points*. Les distributions sont encouragées à se superposer afin de créer un gradient de l'information dans l'espace latent. De cette manière, on pourra générer des données pertinentes en décodant des points différents de ceux rencontrés lors de l'apprentissage. Cette intuition est schématisée sur la figure 2.5. Pour faciliter le calcul de la contrainte de régularité [12], on impose que la distribution d'un point encodé soit normale. L'architecture d'un réseau de neurones variationnel se définit donc par :

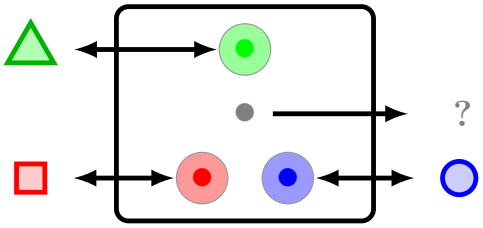
1. Un encodeur constitué de deux sous réseaux. Le premier traite l'état de haute dimension $u_{[:,t]} \in \mathbb{R}^n$ et détermine la moyenne de l'état latent $a_{\mu[:,t]} \in \mathbb{R}^r$. Le second traite l'état de haute dimension et détermine l'écart type de l'état latent $a_{\sigma[:,t]} \in \mathbb{R}^r$. Le vecteur $a_{\sigma[:,t]}$ est de dimension r car on impose l'indépendance des composantes de l'état latent ; cela réduit le nombre de paramètres et revient à dire que l'état latent suit une gaussienne multivariée de covariance diagonale.
2. Un décodeur qui est un réseau de neurones. En entrée, il reçoit un échantillon de la distribution normale $\mathcal{N}\left(a_{\mu,[:,t]}, a^2_{\sigma,[:,t]}\right)$. L'état décodé est $\hat{u}_{[:,t]}$.

12. Elle est précisée plus loin et aussi détaillée dans l'annexe A.



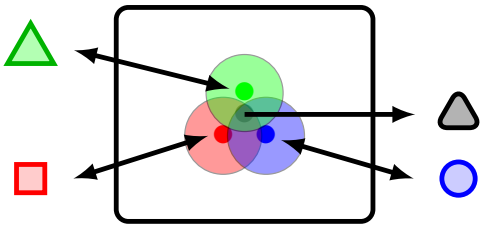
Données d'entraînement bien décodées
Mais comment décoder le
nouveau point ?

(a) Espace latent non structuré.



Données encodées en distributions
Continuité localement améliorée
Espace incomplet

(b) Espace latent non complet.



Distributions qui se superposent
On génère une forme géométrique
pertinente (triangle arrondi)

(c) Espace latent structuré.

Figure *2.5* – *Pour structurer l'espace latent, les données d'entraînement (trois formes géométriques) sont encodées en distributions plutôt qu'en points. Les distributions sont encouragées à se superposer pour créer un gradient de l'information. Cela permet de générer de nouvelles données pertinentes (triangle arrondi).*

Fonction coût

L'enjeu consiste à définir un coût ϵ qui fasse le compromis entre la qualité de l'auto-encodage et la régularité de l'espace latent. Afin d'assurer la complétion et la continuité de l'espace, les distributions sont encouragées à se superposer, comme illustré sur la figure 2.5. Pour ce faire, chaque état encodé est contraint à suivre une loi normale centrée réduite. En rajoutant la contrainte d'auto-encodage, la fonction coût pour un échantillon s'écrit :

$$\epsilon = ||u_{[:,t]} - \hat{u}_{[:,t]}||_2^2 + D_{\mathrm{KL}}\left(\mathcal{N}\left[a_{\mu[:,t]}, a_{\sigma[:,t]}^2\right] \;||\; \mathcal{N}\left[0_r, I_{r\times r}\right]\right)$$

où D_{KL} est la divergence de Kullback-Leibler et mesure les similarités entre les deux lois normales. Ce terme est développée dans Odaibo (2019) et donne :

$$\epsilon = ||u_{[:,t]} - \hat{u}_{[:,t]}||_2^2 - \frac{1}{2}\sum_{i_r=1}^{r}\left[1 + \log\left(a_{\sigma[i_r,t]}^2\right) - a_{\sigma[i_r,t]}^2 - a_{\mu[i_r,t]}^2\right]$$

Les réseaux LVAE et VAE

On considère deux auto-encodeurs variationnels. Le premier est appelée LVAE pour *Linear Variational Auto Encoder* : toutes les activations sont linéaires sauf pour le calcul de l'écart type qui utilise une sigmoïde. Le second est juste appelé VAE pour *Variational Auto Encoder*. Il s'agit du LVAE mais certaines activations linéaires sont remplacées par des tangente hyperbolique. La figure 2.6 résume les architectures retenues, pensées comme de simples extensions de l'analyse en composantes principales.

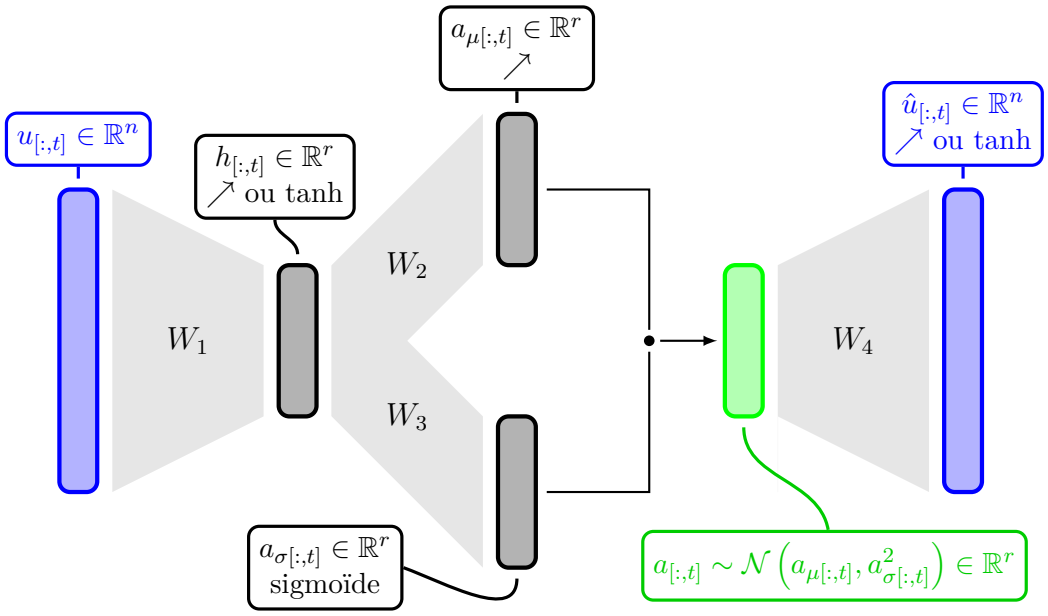


Figure 2.6 – *Réseaux de neurones variationnels étudiés. L'état complet est encodé en une distribution normale. Avant de calculer ses paramètres, une transformation linéaire (LVAE) ou non linéaire (VAE) est effectuée. L'état latent est échantillonné de la distribution puis décodé linéairement (LVAE) ou non linéairement (VAE). Pour ne pas alourdir la figure, les biais ne sont pas indiqués. ↗ signifie linéaire.*

Précisons les équations pour le VAE. La première étape est de transformer l'état de haute dimension $u_{[:,t]}$ en un état caché $h_{[:,t]}$ avec :

$$h_{[:,t]} = \tanh\left(W_1 u_{[:,t]} + b_h\right)$$

La deuxième étape consiste à calculer les paramètres de la distribution :

$$\textbf{Moyenne} \rightarrow a_{\mu[:,t]} = W_2 h_{[:,t]} + b_2$$

$$\textbf{Écart type} \rightarrow a_{\sigma[:,t]} = W_3 h_{[:,t]} + b_3$$

Pour la troisième étape, on échantillonne une loi normale centrée réduite et on forme l'état latent :

$$a_{[:,t]} = z \odot a_{\sigma[:,t]} + a_{\mu[:,t]} \text{ avec } z \sim \mathcal{N}(0_r, I_{r\times r})$$

Pour la dernière étape, on décode l'état latent avec une tangente hyperbolique :

$$\textbf{Décodage} \rightarrow \hat{u}_{[:,t]} = \tanh\left(W_4 a_{[:,t]} + b_4\right)$$

On remarquera que l'état latent est calculé à partir d'une loi $\mathcal{N}(0, I_{r\times r})$ plutôt qu'une loi $\mathcal{N}\left(a_{\mu[:,t]}, a^2_{\sigma[:,t]}\right)$. Dans la littérature, on parle d'astuce de reparamétrisation pour faciliter la rétro-propagation lors de l'apprentissage (Kingma *et al.*, 2015).

Remarque : $a_{[:,t]} \sim \mathcal{N}\left(a_{\mu[:,t]}, a^2_{\sigma[:,t]}\right)$ n'est effectuée que pendant l'apprentissage. Lorsque le modèle est déployé on utilise indépendamment l'encodeur ou le décodeur. Dans le premier cas, l'état latent moyen est la donnée la plus intéressante. Dans le second cas, l'état latent est une donnée utilisateur mise en entrée d'un réseau déterministe.

Structures dominantes

Les poids entre la couche latente et la couche de sortie sont les directions de décodage. Ces modes sont non ordonnés (contrairement à l'analyse en composantes principales) et peu interprétables pour la réduction non linéaire.

Dans le cas de l'auto-encodeur variationnel, on effectue une décomposition en valeurs singulières sur les poids W_4 pour visualiser des modes. On parle de modes *dominants* rangés dans les colonnes de $\Psi_{W_4} \in \mathbb{R}^{n\times r}$ tel que :

$$W_4 = \Psi_{W_4} \Sigma_{W_4} V^*_{W_4}$$

Où les matrices Σ_{W_4} et V_{W_4} sont telles que définies dans la section 2.2.3.

2.3.4 | Synthèse

Les réseaux de neurones permettent de construire des auto-encodeurs aussi flexibles que l'on souhaite. Pour ce travail, les réseaux considérés sont relativement simples et construits sur la base de l'analyse en composantes principales. Le LAE est une extension non orthogonale de la PCA. Le LVAE permet de régulariser l'espace latent. Le VAE permet de régulariser l'espace latent et d'auto-encoder non

linéairement. Dans l'idéal, il faudrait optimiser les hyper-paramètres du réseau au risque d'augmenter drastiquement le nombre de paramètres. On a fait le choix de rester humble pour comprendre la structure des équations.

2.4 | Qualité de l'auto-encodage

Les différentes méthodes d'auto-encodage étant implémentées, on souhaite comparer leur qualité. Dans ce manuscrit, différents outils sont utilisés.

2.4.1 | L'erreur normalisée moyenne

L'erreur moyenne normalisée permet de *quantifier* la qualité de l'auto-encodage sur les données d'entraînement et les données de test. Elle est définie par une version locale et une version globale.

Erreur locale

Si on désigne par $u^b \in \mathbb{R}^{n\times m_b}$ un batch de données, l'erreur normalisée moyenne est le vecteur $\mathrm{NMSE} \in \mathbb{R}^n$ tel que :

$$\forall i_n \in [1..n],\ \mathrm{NMSE}_{i_n} = \frac{\sum_{t=1}^{m_b}\left(u^b_{[i_n,t]} - \hat{u}^b_{[i_n,t]}\right)^2}{\sum_{t=1}^{m_b}\left(u^b_{[i_n,t]} - \overline{u}^b_{[i_n,:]}\right)^2} \in [0,1]$$

où $\overline{u}^b \in \mathbb{R}^n$ est la moyenne d'ensemble des données, calculée sur tous les clichés du batch. La formule fait apparaître l'erreur quadratique moyenne au numérateur et la variance totale au dénominateur. **Si** $\mathrm{NMSE}_{i_n} \to 0$, l'erreur moyenne est faible par rapport à la variabilité des données et peu d'informations sont perdues lors de l'auto-encodage du signal i_n. **Si** $\mathrm{NMSE}_{i_n} \to 1$, beaucoup d'informations sont perdues lors de l'auto-encodage. C'est le cas lorsque le signal i_n varie peu mais que son encodage puis décodage restitue (relativement) de grandes variations.

Erreur globale

En effectuant une moyenne sur toutes les composantes de l'état, on obtient une erreur moyenne :

$$\overline{\mathrm{NMSE}} = \frac{1}{n}\sum_{i_n=1}^{n} \mathrm{NMSE}_{i_n}$$

Une bonne méthode d'auto-encodage assure des erreurs similaires pour les données d'entraînement et les données de test. Dans le cas contraire, il y a deux possibilités. **Soit** l'auto-encodeur est *sur-appris*. Dans ce cas, les modes représentent plus les données d'entraînement qu'ils ne représentent les données de test et $\mathrm{NMSE}_{\mathrm{test}} \gg \mathrm{NMSE}_{\mathrm{train}}$. Pour les réductions neuronales, on peut diminuer l'écart

en modifiant les hyper-paramètres du réseau et en neutralisant certains neurones durant l'apprentissage. **Soit** il y a *fuite* des données. Les données de test sont une redite de *bons* échantillons des données d'entraînement. Cette mauvaise préparation des données se réflète dans $\text{NMSE}_{\text{test}} \ll \text{NMSE}_{\text{train}}$.

Cas particulier de la mécanique des fluides

En mécanique des fluides, la dimension du système n est le nombre de points de maillage multiplié par le nombre de composantes de vitesse. Les cartes d'erreur locale sont tracées par composantes.

2.4.2 | Les réductions linéaires

Pour vérifier que les sous espaces des réductions linéaires PCA et LAE coïncident, on compare les covariances des états latents dans les bases principales. Pour l'analyse en composantes principales, on a :

$$\textbf{PCA} \rightarrow a^{\text{train}} = \left[\Phi_r\right]^T u^{\text{train}} \implies a^{\text{train}} \left[a^{\text{train}}\right]^T = \Phi_r^T \left[u^{\text{train}} \left\{u^{\text{train}}\right\}^T\right] \Phi_r$$

Cette matrice est diagonale par construction. Concernant la réduction linéaire non orthogonale et sans considérer les biais, la covariance de l'état latent est telle que :

$$\textbf{LAE} \rightarrow a^{\text{train}} = W_e u^{\text{train}} \implies a^{\text{train}} \left[a^{\text{train}}\right]^T = W_e \left[u^{\text{train}} \left\{u^{\text{train}}\right\}^T\right] W_e^T$$

Ici, la matrice est vraisemblablement pleine. La base principale du décodeur W_d est déterminée par décomposition en valeurs singulières, selon $W_d = \Psi_d \Sigma_d V_d^T$. Or les poids du décodeur sont reliés aux poids d'encodage par l'inverse de Moore Penrose donc [13] $W_e = W_d^\dagger = V_d \Sigma_d^{-1} \Psi_d^T$. Cela permet de récrire la matrice de covariance :

$$\textbf{LAE} \rightarrow a^{\text{train}} \left[a^{\text{train}}\right]^T = V_d \Sigma_d^{-1} \left\{\Psi_d^T \left[u^{\text{train}} \left\{u^{\text{train}}\right\}^T\right] \Psi_d\right\} \Sigma_d^{-1} V_d^T$$

La matrice entre accolades est la covariance de l'état latent écrite dans la base principale du décodeur. C'est une matrice *approximativement* [14] diagonale dont les composantes doivent tendre vers celles de la matrice PCA. Dans la troisième partie de ce manuscrit, le même travail sera effectué sur les composantes LVAE pour vérifier que le sous espace appris est aussi le même que pour la PCA.

13. $W_e = \left[W_d^T W_d\right]^{-1} W_d^T = \left[V_d \Sigma_d \Psi_d^T \Psi_d \Sigma_d V_d^T\right]^{-1} V_d \Sigma_d \Psi_d^T$. Or les matrices Ψ_d et V_d sont unitaires et l'inverse d'un produit de matrices carrés est le produit (dans le sens contraire) des inverses. Donc $W_e = V_d \left[\Sigma_d^2\right]^{-1} \Sigma_d \Psi_d^T = V_d \Sigma_d \Psi_d^T$

14. Les poids du réseau de neurones peuvent ne pas être totalement convergés.

2.4.3 | Visualisation de l'espace latent

La promesse de l'auto-encodeur variationnel est de rendre complet l'espace latent. Pour cela, les distributions calculées par l'encodeur sont encouragées à se superposer. C'est ce que l'on souhaite vérifier.

Pour avoir une visualisation 2D de l'espace, on commence par une décomposition en valeurs singulières des états latents d'entraînement. On écrit :

$$a^{\text{train}} = \Psi_a \Sigma_a V_a^T$$

Puis on tronque Ψ_a aux deux premières colonnes pour avoir $\Psi_{2\text{D},a} \in \mathbb{R}^{r\times 2}$. Les deux premières composantes principales de l'état $a_{[:,t]}$ sont :

$$a_{2\text{D}[:,t]} = \left[\Psi_{\mathbf{2D},a}\right]^T a_{[:,t]}$$

Pour les réductions variationnelles, la projection de l'écart type est obtenue de la même façon, selon :

$$a_{2\text{D},\sigma[:,t]} = \left[\Psi_{\mathbf{2D},a}\right]^T a_{\sigma[:,t]}$$

L'ellipse de confiance à 95% autour de l'état moyen est ensuite tracé selon :

$$\left(\frac{x - a_{2\text{D}[1,t]}}{a_{2\text{D},\sigma[1,t]}}\right)^2 + \left(\frac{y - a_{2\text{D}[2,t]}}{a_{2\text{D},\sigma[2,t]}}\right)^2 = 5.991$$

Le nombre 5.991 permet d'avoir une ellipse de confiance à 95%, comme détaillé dans la note de bas de page [15]. Bien évidemment, la visualisation 2D d'un espace de dimension r fait perdre de l'information. La décomposition en valeurs singulières nous assure toutefois que la visualisation est *optimale* en termes d'énergie. D'autres méthodes de visualisation, dites de *manifold learning* pourraient être utilisées (Tauro *et al.*, 2014).

2.4.4 | La comparaison des spectres

De l'énergie est perdue lors de l'auto-encodage des données. Pour quantifier cette perte, on compare les valeurs singulières de la matrice des clichés $u_b \in \mathbb{R}^{n\times m_b}$ aux valeurs singulières de la matrice des clichés auto-encodés $\hat{u}_b = d \circ e\,(u_b)$. Pour les réductions linéaires, on s'attend à ce que seules les r premiers niveaux d'énergie

15. Soit une ellipse 2D d'échelle s centrée en $(0,0)$ dont les axes sont de longueurs σ_x et σ_y. L'équation cartésienne est $\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 = s$. Ici, les données x et y sont issues de distributions normales indépendantes, hypothèse faite pour faciliter l'apprentissage des auto-encodeurs variationnels. On se pose la question : comment choisir s pour que l'ellipse représente un intervalle de confiance à $1-\alpha$? Le membre de gauche suit une loi χ_2 à deux degrés de libertés car on calcule la somme des carrés d'échantillons issues de gaussiennes indépendantes. Connaissant la densité de probabilité de cette loi, on peut calculer la vraisemblance de tirer aléatoirement s. On cherche le quantile s_u tel que $P(s < s_u) = 1-\alpha$. De cette manière, on dispose de la plage $[0, s_u]$ pour laquelle la somme des carrés a $1-\alpha$ de chance d'être tirée aléatoirement. Pour une confiance de 95%, on trouve $s_u = 5.991$, d'où la valeur.

soient restitués, l'approximation de rang faible étant construite sur les structures les plus énergétiques. Pour les réductions variationnelles, l'ajout de la pénalité empêche de présager du résultat.

2.5 | Conclusion du chapitre

Les auto-encodeurs permettent d'encoder puis de décoder un état de haute dimension. L'état caché, aussi appelé *latent*, est une représentation basse dimension de l'état complet. Pour déterminer cette représentation, différentes méthodes sont étudiées. **La première** est l'analyse en composantes principales. Elle consiste à diagonaliser la matrice de covariance des données pour trouver des directions non corrélées (orthogonales) rangées par importance énergétique. En se concentrant sur les r premières directions, on dispose de la meilleure représentation de rang r (au sens moindres carrés) des données. **La seconde méthode** est l'auto-encodeur neuronal. En utilisant des réseaux de neurones, la méthode de réduction peut être aussi flexible que souhaitée. L'auto-encodeur linéaire a un lien fort avec l'analyse en composantes principales à ceci près qu'aucune contrainte d'orthogonalité n'est imposée. En encodant les données en distributions plutôt qu'en points, on parle d'auto-encodeur variationnel. Ces auto-encodeurs, dans leurs versions linéaires ou non linéaires, font la promesse de régulariser l'espace latent pour assurer un décodage robuste de nouveaux états latents.

Les architectures des auto-encodeurs neurones ne sont pas optimisées afin de faciliter l'écriture de leur équations et garder un lien avec l'analyse en composantes principales. Les quatre méthodes de réduction retenues pour ce travail sont résumées dans le tableau 2.1.

Finalement, on dispose de structures latentes selon lesquelles le système de haute dimension évolue. On souhaite approximer l'état de ces structures à un instant donné à partir d'informations partielles au même instant. Cela permettra d'estimer l'état de haute dimension à partir d'une estimation de l'état de basse dimension. C'est l'objectif de la **reconstruction** qui est abordée au chapitre 3.

	POD	**LAE**	**LVAE**	**VAE**
Objectif	$\Phi_r \in \mathbb{R}^{n\times r}$	$W_e \in \mathbb{R}^{n\times r}$ et $b_h \in \mathbb{R}^r$ $W_d \in \mathbb{R}^{r\times n}$ et $b_o \in \mathbb{R}^n$	$W_1 \in \mathbb{R}^{n\times r}$ et $b_1 \in \mathbb{R}^r$ $W_2 \in \mathbb{R}^{r\times r}$ et $b_2 \in \mathbb{R}^r$ $W_3 \in \mathbb{R}^{r\times r}$ et $b_3 \in \mathbb{R}^r$ $W_4 \in \mathbb{R}^{r\times n}$ et $b_4 \in \mathbb{R}^n$	Idem LVAE
Encodage	$a_{[:,t]} = \Phi_r^T u_{[:,t]}$	$a_{[:,t]} = W_e u_{[:,t]} + b_h$	$\begin{cases} h_{[:,t]} = W_1 u_{[:,t]} + b_1 \\ a_{\mu[:,t]} = W_2 h_{[:,t]} + b_2 \\ a_{\sigma[:,t]} = \sigma\left[W_3 h_{[:,t]} + b_3\right] \end{cases}$	$\begin{cases} h_{[:,t]} = \tanh\left[W_1 u_{[:,t]} + b_1\right] \\ a_{\mu[:,t]} = W_2 h_{[:,t]} + b_2 \\ a_{\sigma[:,t]} = \sigma\left[W_3 h_{[:,t]} + b_3\right] \end{cases}$
Décodage	$\hat{u}_{[:,t]} = \Phi_r a_{[:,t]}$	$\hat{u}_{[:,t]} = W_d a_{[:,t]} + b_o$	$\begin{cases} a_{[:,t]} \sim \mathcal{N}\left(a_{\mu[:,t]}, a_{\sigma[:,t]}\right) \\ \hat{u}_{[:,t]} = W_4 a_{[:,t]} + b_4 \end{cases}$	$\begin{cases} a_{[:,t]} \sim \mathcal{N}\left(a_{\mu[:,t]}, a_{\sigma[:,t]}\right) \\ \hat{u}_{[:,t]} = \tanh\left[W_4 a_{[:,t]} + b_4\right] \end{cases}$
À noter	Modes orthogonaux $\Phi_r^T \Phi_r = I_{r\times r}$	Modes non orthogonaux $W_e = W_d^\dagger$	-	Réduction non linéaire
Modes	Φ_r	W_d	W_4	Extraits par SVD sur W_4
Méthode	SVD	Minimisation de l'erreur $\epsilon = \|\|u^{\text{train}} - \hat{u}^{\text{train}}\|\|_2^2$	Minimisation de l'erreur et régularisation $\epsilon = \|\|u^{\text{train}} - \hat{u}^{\text{train}}\|\|^2 + D_{KL}\left[\mathcal{N}(a_{\mu[:,t]}^{\text{train}}, a_{\sigma[:,t]}^{\text{train}}) \,\|\, \mathcal{N}(0, I_{r\times r})\right]$	

Tableau 2.1 *– Résumé des méthodes de réduction étudiées.*

3. Reconstruction de l'état latent à partir de mesures limitées

Dans ce chapitre, on s'intéresse à l'estimation de l'état latent du système à partir d'une mesure de l'état de haute dimension. Tout d'abord, on formalise le problème de reconstruction tel que présenté dans l'état de l'art. Puis, on présente l'algorithme non supervisé retenu pour la placement *systématique* des capteurs. Ensuite, on développe la méthode d'estimation directe, purement basée sur l'algèbre linéaire. Enfin, on présente la méthode d'estimation régressive qui couvre un large pan de l'apprentissage supervisé. L'objectif est surtout d'introduire les outils qui seront extensivement utilisés dans la dernière partie du manuscrit.

Sommaire

3.1 | Le problème de reconstruction

En reprenant les notations du chapitre précédent, l'état de haute dimension est noté $u_{[:,t]} \in \mathbb{R}^n$. En appliquant l'opérateur de mesure $\mathcal{H}$, on dispose de sa mesure $y_{[:,t]} = \mathcal{H}\left[u_{[:,t]}\right] \in \mathbb{R}^p$ avec p le nombre de capteurs. Le problème de reconstruction consiste à trouver l'opérateur $\mathcal{G}$ tel que $u_{[:,t]} = \mathcal{G}\left[y_{[:,t]}\right]$. Dans le cas où $\mathcal{H}$ est bien conditionné, trouver cette relation ne poserait pas de problèmes. Dans le cas où les mesures sont limitées [1], c'est une autre affaire : l'opérateur $\mathcal{H}$ est difficilement inversible et il faut réfléchir à d'autres stratégies. Dans l'état de l'art, on a présenté trois approches possibles. Celles-ci sont résumées dans le tableau 3.1.

Méthode	Idée	Description	Formulation
Reconstruction directe	Évaluer $\mathcal{G}\left[y_{[:,t]}\right]$	Résolution d'un problème d'optimisation $\mathcal{P}$	$\hat{u}^{(R)}_{[:,t]} = \min_u \mathcal{P}\left[y_{[:,t]}\right]$
Reconstruction régressive	Estimer $\mathcal{G}$	Apprendre l'opérateur $\mathcal{G}$ Estimation $\hat{\mathcal{G}}$	$\hat{u}^{(R)}_{[:,t]} = \hat{\mathcal{G}}\left[y_{[:,t]}\right]$
Assimilation de données	Mise à jour des prédictions avec $y_{[:,t]}$	Propagation (modèle dynamique f_d) et Analyse (gain de Kalman G_{t+1})	$\begin{cases} \hat{u}^{(P)}_{[:,t]} = f_d\left[\hat{u}^{(A)}_{[:,t-1]}\right] \\ \hat{u}^{(A)}_{[:,t]} = \hat{u}^{(P)}_{[:,t]} + G_t\left\{y_{[:,t]} - \mathcal{H}\left[\hat{u}^{(P)}_{[:,t]}\right]\right\} \end{cases}$

Tableau 3.1 – *Rappel des différentes stratégies de reconstruction. Pour la méthode directe, une estimation requiert de résoudre un problème d'optimisation. Pour la méthode régressive, $\mathcal{G}$ est estimé par $\hat{\mathcal{G}}$ en utilisant l'apprentissage supervisé. Pour l'assimilation de données, on a besoin d'un modèle dynamique f_d pour propager l'état (exposant P) et ensuite le corriger (exposant A). L'exposant R est réservé aux deux premières méthodes de reconstruction.*

L'état latent entre en jeu

Avec l'étape de réduction, le problème de reconstruction peut-être reformulé. On dispose du décodeur d et de l'encodeur e tels que :

$$u_{[:,t]} \approx \hat{u}_{[:,t]} = d \circ e\left(u_{[:,t]}\right) = d\left(a_{[:,t]}\right)$$

avec $a_{[:,t]} \in \mathbb{R}^{r \ll n}$ l'état latent du système. Plutôt que d'estimer l'état de haute dimension, on estime cet état réduit. En notant f_r l'application de reconstruction telle que $a_{[:,t]} = f_r\left[y_{[:,t]}\right]$, on peut écrire :

1. Dans le sens très ponctuelles. C'est généralement le cas pour des applications pratiques avec des raisons évidentes de coût ou de technologie. En mécanique des fluides, on imagine aussi que les mesures sont potentiellement intrusives, ce que l'on cherche à limiter.

$$u_{[:,t]} = \mathcal{G}\left[y_{[:,t]}\right] = d \circ f_r\left[y_{[:,t]}\right]$$

En supposant le décodeur d connu, l'estimation de $u_{[:,t]}$ à partir de sa mesure $y_{[:,t]}$ repose sur l'application $f_r : \mathbb{R}^p \to \mathbb{R}^r$ plutôt que l'application $\mathcal{G} : \mathbb{R}^p \to \mathbb{R}^n$. La figure 3.1 schématise cette stratégie.

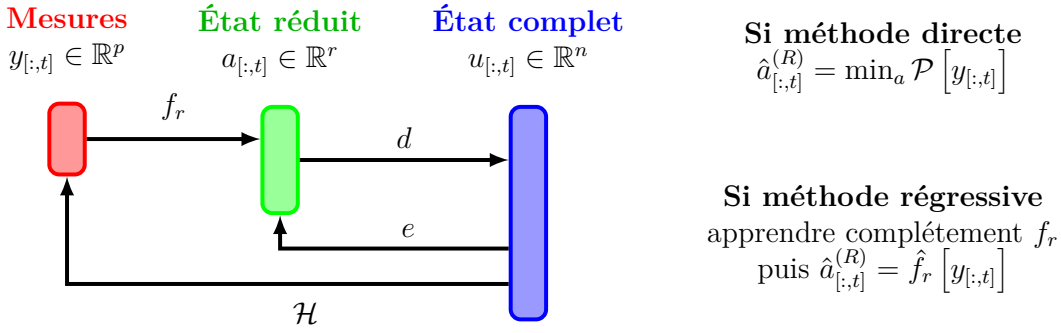


Figure *3.1* – *Schéma du problème de reconstruction : l'état latent est estimé par des mesures courantes via la fonction f_r avant d'être décodé.*

Avant de commencer

Dans la suite, on se concentre sur les méthodes de reconstruction directe et régressive. L'assimilation de données séquentielle (filtre de Kalman) nécessitant un modèle dynamique, cette méthode sera discutée dans le chapitre suivant. Pour indiquer une estimation par reconstruction, on utilisera la notation (R) de sorte qu'un état latent reconstruit est $\hat{a}^{(R)}_{[:,t]}$. Les mesures utilisées pour l'estimation sont des composantes connues de l'état complet. Ainsi, l'opérateur de mesure est une matrice $C \in \mathbb{R}^{p \times n}$ telle que :

$$y_{[:,t]} = C u_{[:,t]}$$

Si le capteur i mesure la composante j de l'état, la ligne $C_{[i,:]}$ est remplie de zéros sauf en colonne j où $C_{[i,j]} = 1$. On définit alors $y^{\text{train}} = Cu^{\text{train}}$ et $y^{\text{test}} = Cu^{\text{test}}$ les mesures d'apprentissage et de test. Dans le cadre de la mécanique des fluides, cela revient à dire qu'on reconstruit l'état latent du champ de vitesse fluctuante en ayant connaissance de quelques signaux de vitesse fluctuante [2]. Pour avoir une estimation du champ instantané, on rajoute le champ moyen calculé sur les données d'entraînement.

2. Par exemple, des fils chauds permettent de mesurer des signaux de vitesse instationnaire.

Dans la suite du chapitre, on propose une méthode systématique pour placer les capteurs et on donne les détails mathématiques des reconstructions directes et régressives.

3.2 | Choix non supervisé d'une grille de capteurs

Attention

Cette section est spécifique aux systèmes définis en temps et en **espace**. À un instant donné, l'état $u_{[:,t]}$ est une écriture discrète de l'état aux n points du maillage spatial.

La méthode est adaptée à un cas mécanique des fluides car $u_{[:,t]} \in \mathbb{R}^n$ est le champ de vitesse fluctuante à un instant donné avec n représentant le nombre de cellules (n_c) multipliée par le nombre de composantes de vitesse (n_v) à estimer.

Pour construire la matrice de mesures C, on utilise un algorithme de clustering amélioré adapté de Jayaraman *et al.* (2019). C'est une stratégie en trois temps :

1. Les $n = n_c \times n_v$ points de maillage sont regroupés en K_c groupes par un algorithme de partitionnement. L'objectif est de déterminer des *grappes* bien différenciées et les plus homogènes possibles.
2. On calcule la variance des données d'entraînement recouvrée par chaque partition. Les partitions sont ensuite classés par ordre décroissant d'énergie.
3. Les centres des p partitions les plus énergétiques sont définis comme capteurs. Si un centroïde ne coïncide pas avec un point de maillage, le capteur est placé au point de maillage le plus proche.

Fonction coût

Les centres des n_c cellules sont répétés n_v fois. Ainsi, l'état du système au point de maillage $x_{[i_n,:]} \in \mathbb{R}^{n_v}$ est $u_{[i_n,t]}$. Partitionner l'ensemble $\{x_{[i_n,:]}\}_{i_n \in [1..n]}$ en K_c grappes revient à déterminer les cellules de Voronoï $\{V_k\}_{k \in [1..K_c]}$ de centroïdes $\{\mu_{[k,:]} \in \mathbb{R}^{n_v}\}_{k \in [1..K_c]}$ solutions du problème d'optimisation (Likas *et al.*, 2003) :

$$V_1, ..., V_{K_c} = \underset{V_1,...,V_{K_c}}{\arg\min} \sum_{k=1}^{K_c} \sum_{x_{i_n} \in V_k} ||x_{[i_n,:]} - \mu_{[k,:]}||_2^2$$

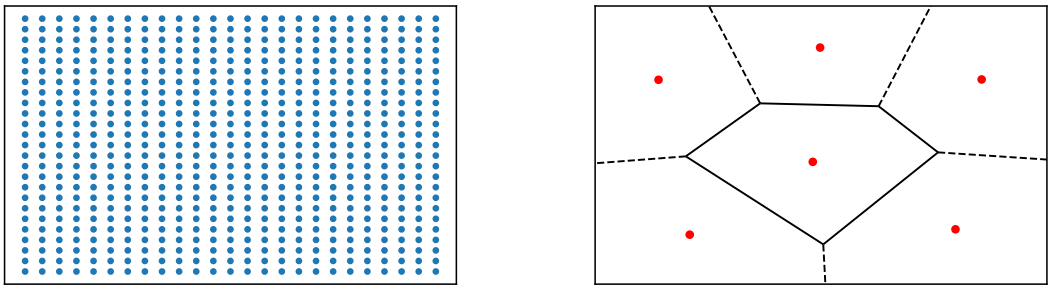
La fonction coût ainsi définie est une mesure de *l'inertie* totale, aussi appelée variabilité intra-classe [3]. Ici, K_c est un hyper-paramètre qui détermine le nombre de classes. En minimisant l'inertie, on maximise l'homogénéité des K_c partitions.

3. *within-cluster variance* en anglais

Avec $K_c = 1$, une seule partition est définie et l'inertie est la somme quadratique des distances au centroïde. Cette somme est minimisée en choisissant :

$$\mu_{[1,:]} = \frac{1}{n} \begin{bmatrix} \sum_{i_n=1}^{n} x_{[i_n,1]} \\ \vdots \\ \sum_{i_n=1}^{n} x_{[i_n,v]} \end{bmatrix}$$

Dans le cas où $K_c > 1$, chaque point de maillage est attribué au cluster dont la distance au centroïde associé est minimal. Cela permet de définir des règles de classification pour des données qui n'étaient pas étiquetées. La figure 3.2 illustre le partitionnement d'un maillage cartésien avec $K_c = 6$ et $n_v = 2$.



(a) Maillage **(b)** Partitions pour $K_c = 6$

Figure *3.2* – *Exemple de partitionnement en six grappes d'un maillage cartésien. Les points rouges représentent les centres des partitions de Voronoï.*

Remarque : dans Jayaraman *et al.* (2019), le partitionnement amélioré définit des points d'interpolation pour ensuite construire un processus gaussien de l'écoulement autour d'un cylindre.

Algorithme

Pour déterminer la position des centroïdes $\{\mu_{[k,:]} \in \mathbb{R}^{n_v}\}_{k\in[1..K_c]}$, on choisit d'utiliser l'algorithme des k-moyennes. C'est une procédure qui consiste à 1) initialiser les centres ; 2) attribuer les points de maillage aux partitions ; 3) recalculer les centres ; 4) reprendre à la deuxième étape[4]. La procédure de placement des capteurs est donc la suivante :

4. Il existe des approches plus élaborées dites *spectrales* et par *agglomération*. La première méthode est utilisée dans Hadjighasem *et al.* (2016) pour déterminer les structures Lagrangiennes d'un écoulement. La seconde méthode est utilisée dans Costa *et al.* (2019) pour segmenter l'espace de phase de Lorenz 63 et construire des modèles dynamiques locaux.

Algorithme de partitionnement amélioré

1. **Définir K_c centres initiaux :**

$$\{\mu^{(0)}_{[k,:]}\}_{k\in[1,..K_c]}$$

2. **Tant que les centres ne sont pas convergés (indice l) :**

   (a) **Déterminer les cellules de Voronoï :**

$$V_k^{(l)} = \left\{ x_{[i_n,:]} \in \mathbb{R}^{n_v} : \left|\left| x_{[i_n,:]} - \mu^{(l)}_{[k,:]} \right|\right|_2^2 \leq \left|\left| x_{[i_n,:]} - \mu^{(l)}_{[k^*,:]} \right|\right|_2^2 \quad \forall k^* \in [1..K_c] \right\}$$

   Cette étape permet de définir la région de l'espace où les points $x_{[i_n,:]}$ sont plus proches du centre k que les centres voisins k^*.

   (b) **Calculer les nouveaux centres :**

$$\mu^{(l+1)}_{[k,:]} = \frac{1}{|V_k^{(l)}|} \sum_{\substack{i_n \text{ tel que} \\ x_{[i_n,:]}\in V_k}} x_{[i_n,:]}$$

   Où $|V_k^{(l)}|$ désigne la cardinalité de la partition k à l'étape l.

3. **Dans chaque partition, calculer la variance recouvrée :**

$$\sigma^2_{V_k} = \frac{1}{m_{\text{train}}} \sum_{t=1}^{m_{\text{train}}} \sum_{\substack{i_n \text{ tel que} \\ x_{[i_n,:]}\in V_k}} \left[u^{\text{train}}_{[i_n,t]}\right]^2$$

   Pas besoin de retrancher la moyenne de l'état car les données pour l'apprentissage sont centrées (voir chapitre 2 ; en mécanique des fluides, on travaille avec le champ de vitesse fluctuante).

4. **Sélectionner les p capteurs tels que :**

$$\sigma^2_{V_1} \geq ... \geq \sigma^2_{V_p}$$

Si on dénote σ^2 la variance totale des données d'entraînement, on vérifie la relation :

$$\sigma^2 = \sum_{k=1}^{K_c} \sigma^2_{V_k}$$

En choisissant p capteurs parmi les K_c centres disponibles, une partie de la variabilité des données n'est pas mesurée. Toute la difficulté consiste à choisir l'hyper-paramètre

K_c pour avoir un nombre raisonnable de candidats [5] Dans la troisième partie du manuscrit, on fixera $K_c = 500$ sans chercher à l'optimiser [6].

Critiques du placement

La méthode proposée permet un placement *systématique* de capteurs et fait le compromis entre une approche naïve (grille aléatoire) et une approche coûteuse (placement QR, voir Manohar *et al.* (2018a) pour un exposé didactique). La méthode est appelée partitionnement *amélioré* car elle combine une approche non supervisée (k-moyennes) avec de la connaissance (calcul de variance). Trois inconvénients sont à soulever :

→ Les capteurs sont placés en fonction de la variance recouvrée des données d'entraînement. Cela n'empêche pas les multi-colinéarités qui pourraient nuire aux méthodes d'estimation régressive. Arnault *et al.* (2016) mesurent cet impact en estimation stochastique linéaire via le facteur d'inflation de la variance.

→ L'observabilité et la controllabilité des capteurs n'est pas attestée. Pour des applications pratiques, c'est ce vers quoi il faudrait tendre. Les approches de troncature équilibrée sont prometteuses mais sortent du cadre de ce manuscrit (Manohar *et al.*, 2018b).

→ Le placement n'exploite pas les structures spatiales déterminées lors de l'étape de réduction. Si un capteur est placé sur le zéro d'un mode, on est certain que le capteur ne donnera aucune information sur ce mode. Il est plus judicieux de placer les capteurs là où *chaque* mode fournit de l'information (Cohen *et al.*, 2003).

Maintenant que l'on dispose des mesures de l'état complet, on souhaite utiliser ces informations pour estimer l'état latent du système.

3.3 | Estimation directe de l'état latent

L'estimation directe consiste à écrire l'état latent comme une combinaison linéaire d'éléments de référence. Les coefficients de la combinaison changent à chaque estimation et sont déterminés en résolvant un problème d'optimisation construit sur le vecteur de mesure.

5. Si K_c est trop petit, le maillage est décrit par de grosses partitions très variables. Synthétiser la variabilité d'une telle partition par l'information au niveau du centroïde n'est pas pertinent, surtout si la source de variabilité en est éloignée. Si K_c est trop grand, on prend le risque de construire des petites partitions *trop* homogènes et on se retrouve à définir des capteurs très proches les uns des autres.

6. Une des méthodes possibles est le critère du coude. Le nombre de clusters optimal correspond au coude de la courbe d'inertie en fonction du nombre de clusters. Au delà de ce coude, l'ajout de partitions ne vaut plus le gain en inertie (Kodinariya et Makwana, 2013). On reparlera de ce critère dans le chapitre suivant. À noter que dans un contexte opérationnel (non numérique), ce nombre maximal de capteurs est déjà très élevé voir inatteignable.

Problème général

L'état complet $u_{[:,t]} \in \mathbb{R}^n$ est mesuré en utilisant la matrice $C \in \mathbb{R}^{p \times n}$ selon $y_{[:,t]} = Cu_{[:,t]} \in \mathbb{R}^p$. Avec l'état latent $a_{[:,t]} \in \mathbb{R}^r$ et le décodeur $d : \mathbb{R}^r \to \mathbb{R}^n$, l'état complet est approximé par $\hat{u}_{[:,t]}$. Sa mesure est donnée par :

$$\hat{y}_{[:,t]} = C\hat{u}_{[:,t]} = Cd\left(a_{[:,t]}\right)$$

L'objectif est de trouver l'état latent qui minimise l'erreur entre la mesure réelle $y_{[:,t]}$ et son approximation $\hat{y}_{[:,t]}$. Pour les auto-encodeurs linéaires, ce problème d'optimisation est un problème *d'algèbre linéaire*. Pour des décodeurs non linéaires, la recherche d'une solution est beaucoup plus complexe. Dans le suite, on ne se concentre que sur les réductions linéaires.

Cas des réductions linéaires

Pour l'analyse en composante principales et les auto-encodeurs linéaires, la forme générale du décodage est :

$$d\left(a_{[:,t]}\right) = Wa_{[:,t]} + b$$

où les colonnes de $W \in \mathbb{R}^{n \times r}$ sont les modes et $b \in \mathbb{R}^n$ contient les biais. En utilisant l'opérateur de mesure, l'état mesuré est approché par :

$$\hat{y}_{[:,t]} = CWa_{[:,t]} + Cb$$

Minimiser l'erreur entre $\hat{y}_{[:,t]}$ et $y_{[:,t]}$ revient à résoudre le système linéaire de p équations et r inconnues :

$$CWa_{[:,t]} + Cb = y_{[:,t]} \leftrightarrow \theta a_{[:,t]} = s_{[:,t]}$$

Dans cette écriture, $\theta = CW \in \mathbb{R}^{p \times r}$ contient les modes mesurés, $a_{[:,t]} \in \mathbb{R}^r$ est l'inconnue et $s_{[:,t]} = y_{[:,t]} - Cb \in \mathbb{R}^p$ est le second membre. Deux configurations sont possibles (Al Mamun *et al.*, 2018).

Dans le premier cas, le rang de la matrice augmentée $\left[\theta \mid s_{[:,t]}\right]$ est identique au rang de la matrice θ. Le vecteur $s_{[:,t]}$ est une combinaison linéaire des colonnes de θ et il existe *au moins* une solution au problème.

Dans le cas $\mathbf{rg}\left[\theta \mid s_{[:,t]}\right] = \mathbf{rg}\left[\theta\right]$

Il y a deux possibilités :

- → Si $\mathrm{rg}(\theta) = r$ alors il existe une solution *unique*, déterminée en inversant le système.
- → Si $\mathrm{rg}(\theta) < r$, il y a plus d'inconnues que de contraintes libres. Il existe une *infinité* de solutions mais on choisit celle de norme minimale. On

résout donc :

$$\hat{a}^{(R)}_{[:,t]} = \begin{cases} \min\limits_{a} ||a||_1 \\ \text{tel que } \theta a = s_{[:,t]} \end{cases}$$

L'utilisation de la norme l_1 promeut la parcimonie dans la solution. La robustesse de la méthode pour une application en mécanique des fluides est étudiée dans Callaham *et al.* (2019) sous le nom de *Sparse Reconstruction*.

Dans le second cas, le rang de la matrice augmentée $\left[\theta \mid s_{[:,t]}\right]$ est plus grand que le rang de θ. Les colonnes de θ et le second membre $s_{[:,t]}$ forment une famille libre et il n'y a *aucune* solution au problème. On cherche une solution approchée.

Dans le cas rg $\left[\theta \mid s_{[:,t]}\right] >$ rg $[\theta]$

On cherche une solution approchée pour que la somme quadratique des carrées soit minimale. Cette *pseudo-solution* s'écrit :

$$\hat{a}^{(R)}_{[:,t]} = \arg\min_{a} ||\theta a - s_{[:,t]}||_2^2$$

Pour obtenir la solution, on projette orthogonalement $s_{[:,t]}$ dans l'espace des colonnes Col(θ) et on cherche $\hat{a}_{[:,t]}$ permettant d'atterrir sur le projeté. On suit les étapes suivantes :

Étape 1 → On écrit $s_{[:,t]} = s = s_\perp + s_\parallel$. Le vecteur $s_\perp$ appartient à l'espace Col(θ) tandis que $s_\parallel$ appartient à l'espace orthogonal Ker(θ^T).

Étape 2 → La condition d'appartenance à l'orthogonal (donc au noyau) s'écrit $\theta^T s_\parallel = 0_{\mathbb{R}^p}$, ce que l'on traduit par $\theta^T s = \theta^T s_\perp$

Étape 3 → Le vecteur $s_\perp$ vit dans Col(θ) donc rg $[\theta \mid s_\perp]$ = rg $[\theta]$. L'étude de cas de la première configuration assure l'existence d'une solution a_R telle que $s_\perp = \theta a_R$.

Étape 4 → Le système de l'étape 2 se réécrit $\theta^T s = \theta^T \theta a_R$. Il s'agit d'un nouveau système linéaire dont la solution *unique* est :

$$\hat{a}^{(R)}_{[:,t]} = a_R = \left[\theta^T\theta\right]^{-1} \theta s = \theta^\dagger s$$

La pseudo-solution est donc obtenue avec l'inverse de Moore Penrose de θ.

Sous-déterminé et sur-déterminé

La matrice $\theta \in \mathbb{R}^{p \times r}$ représente une application linéaire de $\mathbb{R}^r$ dans $\mathbb{R}^p$. Le rang d'une application étant invariant par transposition, on a la condition :

$$\boxed{\text{rg}(\theta) \leq \min(p, r)}$$

Dans le cas où il y a plus de capteurs que de modes ($r < p$), la condition sur le rang s'écrit $\text{rg}(\theta) \leq r$. Si la matrice est de rang plein ($\text{rg}(\theta) = r$), ce sont $p - r$ contraintes qui ne seront pas satisfaites. Le système est dit *sur-déterminé* et on se retrouve dans la deuxième classe de problèmes.

Dans le cas où il y a moins de capteurs que de modes ($p < r$), la condition sur le rang s'écrit $\text{rg}(\theta) \leq p$. Si la matrice est de rang plein ($\text{rg}(\theta) = p$), il manque $r - p$ contraintes. Le système est dit *sous-déterminé* et on se retrouve dans la première classe de problèmes, deuxième configuration.

Conclusion

Dans la méthode directe, l'état latent est estimé comme une combinaison linéaire d'éléments de référence. Les coefficients de la combinaison sont déterminés par le vecteur de mesure, en résolvant un système linéaire dans le cas d'un décodeur linéaire et d'une matrice de mesures. Une estimation requérant une optimisation, la méthode ne semble pas adaptée pour du temps réel bien que l'on puisse potentiellement changer la position de capteurs à chaque nouvelle estimation. Il semble plus judicieux d'apprendre complètement la relation de passage entre l'espace de mesures et l'espace latent, ce qui constitue l'objectif de l'approche régressive.

3.4 | Estimation de l'état latent par régression

Dans l'approche régressive, on utilise les techniques de l'apprentissage supervisé pour apprendre la fonction f_r. Tout l'enjeu consiste à choisir un *bon* modèle qui fasse le compromis entre le biais et la variance. Dans la suite, on détaille l'intérêt de ce compromis puis quelques formes de modèles. On présente également la procédure de validation croisée pour la sélection des hyper-paramètres.

3.4.1 | Compromis biais-variance

Dans la section 3.1, on a écrit la relation $a_{[:,t]} = f_r\left[y_{[:,t]}\right]$. Dans les faits, l'état latent peut ne pas être déterministe et il convient d'ajouter un terme *d'erreur irréductible*. On écrit alors :

$$\boxed{a_{[:,t]} = f_r\left[y_{[:,t]}\right] + \epsilon \text{ avec } \epsilon \sim \mathcal{N}(0, \Sigma^2)}$$

où Σ^2 est la matrice de covariance de l'erreur, supposée diagonale et donc définie par r composantes. L'objectif est d'apprendre une approximation de f_r qui se généralise à d'autres points que $(y^{\text{train}}, a^{\text{train}})$. La procédure d'apprentissage est en deux temps :

Étape 1 → on choisit une forme de modèle pour f_r. Le modèle peut expliquer les r composantes de l'état latent indépendamment (méthode mono-tâche) ou simultanément (méthode multi-tâches) à partir d'une combinaison des mesures (méthode paramétrique) ou d'une prise de décision (méthode non paramétrique).

Étape 2 → le modèle choisi est optimisé en minimisant un coût évalué sur les données d'entraînement. Si on désigne par $\mathcal{D}$ l'ensemble d'apprentissage et $\hat{f}_r$ le modèle appris, une reconstruction s'écrit :

$$\hat{a}^{(R)}_{[:,t]} = \hat{f}_r \left[y_{[:,t]} \; ; \; \mathcal{D} \right]$$

Pour que $\hat{f}_r$ se généralise à de nouvelles données, il faut assurer un compromis entre le *biais* du modèle et sa *variance*. Ces termes apparaissent en décomposant l'erreur quadratique moyenne pour un nouveau point. Les formules [7] sont indiquées dans l'encadré ci-dessous.

Décomposition de l'erreur d'estimation

En utilisant une nouvelle mesure $y_{[:,t]}$, l'erreur quadratique moyenne pour la reconstruction de $a_{[i_r,t]}$ s'écrit :

$$\mathcal{E} = \mathbb{E}_{\mathcal{D}} \left[\left(a_{[i_r,t]} - \left\{ \hat{f}_r \left[y_{[:,t]} \; ; \; \mathcal{D} \right] \right\}_{i_r} \right)^2 \right]$$

L'espérance est indexée par $\mathcal{D}$ car le calcul doit considérer différents choix pour l'ensemble d'entraînement [a]. L'indice i_r renvoie à la composante i_r de l'estimation. L'erreur peut être décomposée selon :

$$\mathcal{E} = \text{Biais}_{\mathcal{D}} \left[\left\{ \hat{f}_r \left[y_{[:,t]} \; ; \; \mathcal{D} \right] \right\}_{i_r} \right]^2 + \text{Var}_{\mathcal{D}} \left[\left\{ \hat{f}_r \left[y_{[:,t]} \; ; \; \mathcal{D} \right] \right\}_{i_r} \right] + \Sigma^2_{i_r}$$

Chacun de ces termes a une interprétation :

→ Le **biais** de la méthode d'apprentissage indique si le modèle choisi est un bon candidat pour approximer la fonction réelle. Plus la méthode est flexible (i.e. hyper-paramétrable) et plus son biais peut être diminué [b]. Le biais se calcule selon :

7. Volontairement écrites avec les notations complètes. Compacter comme dans certains sources peut provoquer des contre-sens. Ici, on insiste sur le fait que l'erreur d'estimation d'un nouveau point est calculée pour **tous les choix** possibles de l'ensemble d'entraînement.

$$\mathbb{E}_{\mathcal{D}}\left\{\left(\hat{f}_r\left[y_{[:,t]}\ ;\ \mathcal{D}\right]\right)_{i_r} - \left(f_r\left[y_{[:,t]}\ ;\ \mathcal{D}\right]\right)_{i_r}\right\}$$

→ La **variance** de la méthode d'apprentissage indique comment varie l'erreur en changeant l'ensemble d'apprentissage. Une variance élevée signifie que le modèle est trop spécifique aux données utilisées [c]. On la calcule selon :

$$\mathbb{E}_{\mathcal{D}}\left\{\left[\left(\hat{f}_r\left[y_{[:,t]}\ ;\ \mathcal{D}\right]\right)_{i_r} - \mathbb{E}_{\mathcal{D}}\left\{\left(f_r\left[y_{[:,t]}\ ;\ \mathcal{D}\right]\right)_{i_r}\right\}\right]^2\right\}$$

→ L'**erreur irréductible** est liée au bruit de l'état latent, qu'aucun modèle ne peut capturer.

a. C'est pourquoi la décomposition a surtout un intérêt théorique et que l'erreur $\mathcal{E}$ est très peu calculée en pratique

b. Si la fonction réelle est non linéaire, un modèle linéaire est nécessairement biaisé. Un réseau de neurones est plus pertinent et on peut diminuer le biais en augmentant la complexité du réseau

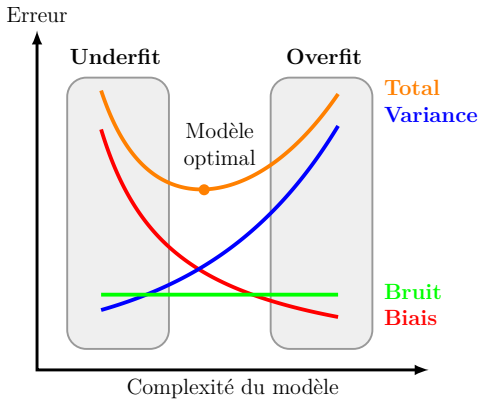
c. Ce que l'on souhaite éviter à tout prix si l'objectif est de déployer le modèle et l'utiliser sur de nouvelles données.

Le compromis entre le biais et la variance est illustré de façon *prototype* sur la figure 3.3. **En complexifiant** le modèle, on diminue son biais mais on augmente sa variance. On risque le *sur-apprentissage* c'est-à-dire une bonne performance sur les données d'entraînement mais une mauvaise performance sur des données de test. **En simplifiant** le modèle, on augmente son biais mais on diminue sa variance. On risque le *sous-apprentissage* c'est-à-dire d'apprendre un modèle généraliste mais trop peu précis. **Le modèle idéal** équilibre le biais et la variance, permettant des performances similaires et raisonnables sur les données d'entraînement et de test. Pour une forme de modèle, ce compromis est atteint en *régularisant* l'apprentissage par diverses méthodes (Belkin *et al.*, 2018).

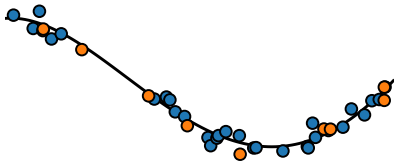
Dans la suite, on présente des formes de modèles pour l'apprentissage supervisé et quelques méthodes de régularisation. L'objectif est de fournir une carte d'identité pour les différentes fonctions de reconstruction qui seront utilisées dans le manuscrit.

3.4.2 | Estimation paramétrique

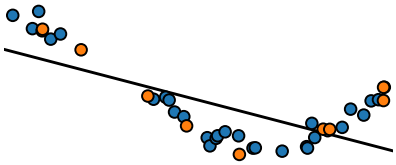
On considère différentes formes paramétriques pour la fonction $\hat{f}_r$. Celles-ci sont la régression linéaire, la régression par vecteurs supports et la régression par réseau de neurones. Les paramètres optimaux sont déterminés en minimisant l'erreur entre les données étiquetées $(y^{\text{train}}, a^{\text{train}})$ et les données reconstruites $(y^{\text{train}}, \hat{f}_r\left[y^{\text{train}}\right])$. Dans cette sous-section, on détaille les trois formes de modèle.



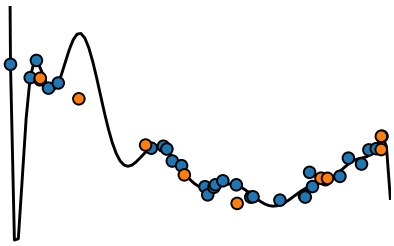
(a) Évolution prototype des courbes d'erreurs



(b) Modèle qui fait un compromis entre le biais et la variance.



(c) Modèle de biais important mais avec une variance faible.



(d) Modèle de biais faible mais avec une variance importante.

Figure *3.3* – *Illustration du compromis entre le biais et la variance. Pour les figures b,c et d, les points bleus représentent les données d'apprentissage et les points oranges sont les points de test. Le modèle en c) est robuste mais trop peu précis. Le modèle en d) est flexible mais peu performant sur les nouvelles données.*

Régression linéaire

La régression linéaire est la méthode la plus simple à implémenter. On la propose en version mono-tâche et en version multi-tâche.

→ Version mono-tâche

Chaque composante de l'état latent est modélisée indépendamment. Pour la composante i_r, le modèle explique la sortie $a_{[i_r,t]}$ par une combinaison linéaire des composantes de la mesure. La formule de reconstruction est donc :

$$\forall i_r \in [1..r] \rightarrow \hat{a}^{(R)}_{[i_r,t]} = \beta^T_{i_r} y_{[:,t]}$$

où le vecteur $\beta_{i_r} \in \mathbb{R}^{p\times 1}$ est spécifique à la reconstruction de la composante i_r de l'état latent. La fonction coût est la somme quadratique des erreurs d'entraînement que l'on pénalise avec une norme l_1. On écrit :

$$\forall i_r \in [1..r] \rightarrow \mathcal{L}(\beta_{i_r}) = \frac{1}{2m_{\text{train}}} ||R_{i_r}(\beta_{i_r})||^2_F + \alpha_{i_r} ||\beta_{i_r}||_1$$

où l'expression du résidu, de la norme de Frobenius et de la norme l_1 sont :

$$\begin{cases} R_{i_r}(\beta_{i_r}) = \left\{a^{\text{train}}_{[i_r,:]}\right\}^T - \left\{y^{\text{train}}\right\}^T \beta_{i_r} \\ ||R_{i_r}(\beta_{i_r})||_F = \left(\sum_{i,j} [R_{i_r}(\beta_{i_r})]^2_{[i,j]}\right)^{\frac{1}{2}} \\ ||\beta_{i_r}||_1 = \sum_{i_p=1}^{p} |\, [\beta_{i_r}]_{i_p} \,| \end{cases}$$

La norme l_1 exprime un besoin de parcimonie dans la solution. C'est un moyen de régularisation (dit du LASSO [8]) qui permet de sélectionner les variables. Le degré de sélection est contrôlé par l'hyper-paramètre α_{i_r}. Plus ce scalaire est élevé, plus le nombre de variables explicatives (c'est-à-dire de capteurs) neutralisées est important. À noter que l'on aurait pu pénaliser le coût quadratique par une norme l_2 (régression forte pente [9]). On aurait alors encouragé les solutions de faible variance. La figure 3.4 illustre la différence entre les deux régularisations mais dans le manuscrit, on se concentre uniquement sur la régression LASSO.

L'ajout de la norme l_1 empêche d'appliquer une descente de gradient classique car la norme n'est pas différentiable au vecteur nul. Il faut effectuer une descente de gradient *implicite* à l'aide de l'opérateur proximal. On renvoit le lecteur à la référence Zheng *et al.* (2018) pour plus de détails [10].

8. *Least Absolute Shrinkage and Selection Operator*, inventé en 1996 (Tibshirani, 1996). On parle de sélection car la norme l_1 a des bords aigus qui vont donc promouvoir la parcimonie dans une solution moindres carrés. Pour l'anecdote : la norme l_1 est appelée distance de Manhattan car dans cette ville, un taxi se déplace toujours sur des rues perpendiculaires.

9. *Ridge regression* en anglais (Hoerl et Kennard, 1970). On parle aussi de régularisation de Tikhonov.

10. La descente de gradient s'écrit $\beta_{i_r} \leftarrow \text{proj}_{|.|}\left(\beta_{i_r} - \eta \nabla \text{MSE}\right)$. L'argument correspond à une descente moindres carrés classique et $\text{proj}_{|.|}$ permet de prendre en compte la norme l_1. Les calculs montrent que cet opérateur est un *soft-thresholding*, ce qui est cohérent avec l'action de sélection de la norme l_1.

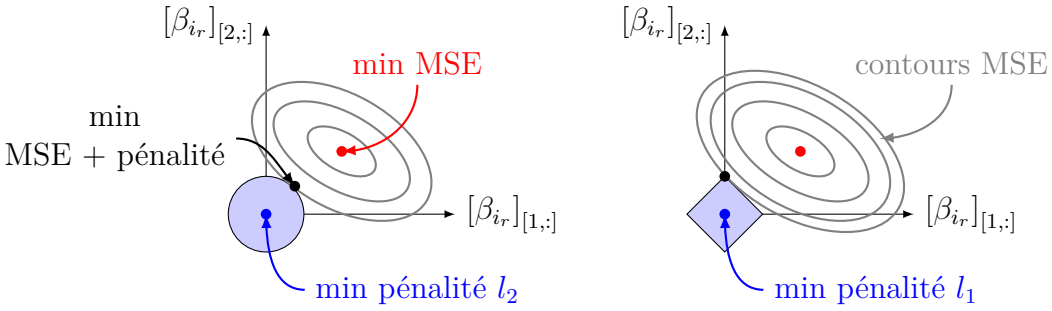


Figure *3.4* – *Principe de la régularisation de forte pente (norme l_2) ou lasso (norme l_1). Les moindres carrés donnent la solution rouge. Les normes l_1 ou l_2 sont minimales au point bleu. L'erreur quadratique pénalisée conduit à la solution* ***noire****. Les bords aigus de la norme l_1 favorisent une intersection des contours MSE et de la norme sur les axes et donc la neutralisation d'autres variables. La neutralisation est d'autant plus importante que la boule unité est de rayon importante, ce qui est contrôlé par l'hyper-paramètre α_{i_r}*

→ Version multi-tâche

Dans sa version multi-tâche, toutes les composantes de $a_{[:,t]}$ sont expliquées simultanément par une combinaison linéaire des composantes de $y_{[:,t]}$. La formule de reconstruction est donc :

$$\hat{a}^{(R)}_{[:,t]} = \beta^T y_{[:,t]}$$

avec $\beta \in \mathbb{R}^{p \times r}$. Pendant l'apprentissage, le coût est évalué selon :

$$\mathcal{L}(\beta) = \frac{1}{2m_{\text{train}}} ||R(\beta)||_F^2 + \alpha ||\beta||_{21}$$

où le résidu est calculé pour toutes les composantes de l'état latent et le paramètre de régularisation α est **unique** pour tous les modes. Les expressions détaillées sont :

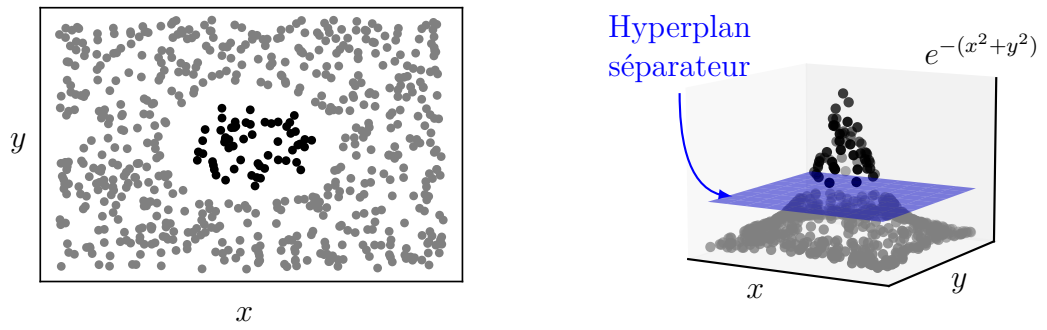
$$\begin{cases} R(\beta) = \left\{a^{\text{train}}\right\}^T - \left\{y^{\text{train}}\right\}^T \beta \\ ||R(\beta)||_F = \left(\sum_{i,j} [R(\beta)]^2_{[i,j]} \right)^{\frac{1}{2}} \\ ||\beta||_{21} = \sum_{i_p=1}^{p} \left(\sum_{i_r=1}^{r} \beta_{[i_p,i_r]} \right)^{\frac{1}{2}} \end{cases}$$

La norme l_{21} fut introduite par Li *et al.* (2018b). L'avantage de la formulation multi-tâche est qu'elle sélectionne les mêmes mesures pour expliquer toutes les composantes de l'état latent. Pour des reconstructions pratiques comme évoquées dans l'état de l'art, c'est une approche privilégiée.

Remarque : en mécanique des fluides, la formulation mono-tâche sans terme de pénalisation constitue la méthode *d'estimation stochastique linéaire*. D'un point de vue historique, la méthode a été pensée comme un développement au premier ordre d'une moyenne conditionnelle (voir chapitre 1 et Adrian (1975)).

Machines à vecteurs supports

Les machines à vecteurs supports (SVM) furent initialement introduits pour des problèmes de classification binaire. L'objectif était de trouver la meilleure séparatrice *linéaire* entre deux classes. Ce cadre d'application très restrictif a été étendu grâce à l'astuce du noyau (Vapnik, 1999) : si les données ne sont pas linéairement séparables dans l'espace latent, il suffit de passer dans un espace de dimension supérieure où une séparation linéaire des données est pertinente (voir la figure 3.5). Ce principe de base est la clé des régressions par vecteur supports (SVR), que l'on explicite dans un cas linéaire et non linéaire.



(a) Dans l'espace initial, les données ne sont pas séparables linéairement.

(b) En ajoutant une nouvelle dimension, on peut définir une règle de décision linéaire.

Figure *3.5* – *Illustration de l'astuce du noyau pour les machines à vecteur supports. Pour décider de la classe d'un nouveau point, la décision est prise dans l'espace de haute dimension puis on repasse dans l'espace initial.*

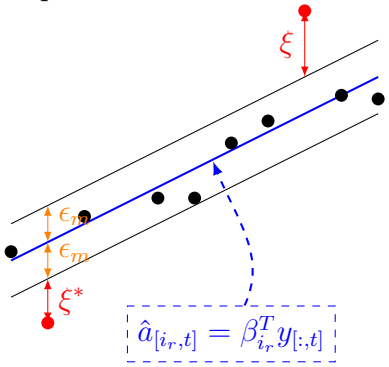
→ Formulation linéaire

Chaque composante de l'état latent est expliqué *indépendamment* (mono-tâche) par une combinaison linéaire des mesures. Ainsi, la formule de reconstruction est :

$$\forall i_r \in [1..r] \rightarrow \hat{a}^{(R)}_{[i_r,t]} = \beta_{i_r}^T y_{[:,t]}$$

Dans le cas de la régression linéaire classique, les paramètres $\beta_{i_r} \in \mathbb{R}^{p\times 1}$ étaient déterminés en minimisant la somme des résidus quadratiques. Ici, l'objectif est de déterminer l'hyperplan qui assure une déviation d'au plus ϵ_m par rapport aux étiquettes réelles. La formulation *primale*[11] du problème est donc la suivante :

$$\begin{cases} \min\limits_{\beta_{i_r},\xi,\xi^*} \dfrac{1}{2}\beta_{i_r}^T\beta_{i_r} + C_B \sum\limits_{t=1}^{m_{\text{train}}} (\xi_t + \xi_t^*) \\ a^{\text{train}}_{[i_r,t]} - \beta_{i_r}^T y^{\text{train}}_{[:,t]} \leq \epsilon_m + \xi_t \ \forall t \\ \beta_{i_r}^T y^{\text{train}}_{[:,t]} - a^{\text{train}}_{[i_r,t]} \leq \epsilon_m + \xi_t^* \ \forall t \\ \xi_t, \xi_t^* \geq 0 \ \forall t \end{cases}$$



Les variables $\xi \in \mathbb{R}^{m_{\text{train}}}$ et $\xi^* \in \mathbb{R}^{m_{\text{train}}}$ pénalisent les observations qui sont hors du tube ϵ_m. Cette pénalisation est contrôlée par l'hyper-paramètre C_B appelée contrainte de la boîte[12]. En introduisant les multiplicateurs de Lagrange $\alpha \in \mathbb{R}^{m_{\text{train}}}$ (associé à ξ) et $\alpha^* \in \mathbb{R}^{m_{\text{train}}}$ (associé à ξ^*), on obtient la formulation *duale*[13] du problème :

$$\begin{cases} \min\limits_{\alpha,\alpha^*} \mathcal{L}(\alpha,\alpha^*) \\ \text{tel que } \sum\limits_{t=1}^{m_{\text{train}}} (\alpha_t - \alpha_t^*) = 0 \text{ avec } 0 \leq (\alpha_t, \alpha_t^*) \leq C \end{cases}$$

où la fonction coût est donnée par :

$$\mathcal{L}(\alpha,\alpha^*) = \frac{1}{2}(\alpha-\alpha^*)^T Q_{\text{SVR}}(\alpha-\alpha^*) + \epsilon_m e^T(\alpha+\alpha^*) - a^{\text{train}}_{[i_r,:]}(\alpha-\alpha^*)$$

Dans cette formule, $e \in \mathbb{R}^{m_{\text{train}}}$ est un vecteur ne contenant que des uns et $Q_{\text{SVR}} \in \mathbb{R}^{m_{\text{train}}\times m_{\text{train}}}$ est définie par $Q_{\text{SVR}} = [y^{\text{train}}]^T y^{\text{train}}$. C'est une matrice définie positive qui calcule les covariances entre les mesures d'entraînement dans l'espace des mesures.

11. Première forme du problème, où les contraintes sont explicites.
12. Ou *box constraint* en anglais. On comprend ce terme en regardant la formulation duale où les multiplicateurs sont contraints dans une boîte de taille C_B.
13. Autre formulation du problème où les contraintes sont prises implicitement. La fonction duale de Lagrange a l'avantage d'être concave, indépendamment de la convexité du problème primal. Le maximum de la fonction de Lagrange donne une borne inférieure à la solution du problème primal.

Dans le cadre d'une dualité forte [14], les conditions de Kuhn et Tucker (1951) qui sont nécessaires à l'optimalité de la solution s'écrivent :

$$\forall t \in [1..m_{\text{train}}] \begin{cases} \alpha_t \left(\epsilon_m + \xi_t - a^{\text{train}}_{[i_r,t]} + \left[\hat{a}^{(R)}_{[i_r,t]} \right]^{\text{train}} \right) = 0 \\ \alpha^*_t \left(\epsilon_m + \xi^*_t + a^{\text{train}}_{[i_r,t]} - \left[\hat{a}^{(R)}_{[i_r,t]} \right]^{\text{train}} \right) = 0 \\ \xi_t (C_B - \alpha_t) = 0 \\ \xi^*_t (C_B - \alpha^*_t) = 0 \end{cases}$$

Ces conditions indiquent que les observations strictement à l'intérieur du tube ont un multiplicateur nul tandis que les observations à l'extérieur du tube sont pénalisées. Si α_t ou α^*_t ne sont pas nuls, on dit que l'observation associée est un *vecteur support*. Lorsque tous les multiplicateurs de Lagrange ont été calculés (par descente de gradient), on peut calculer le vecteur de paramètres selon :

$$\forall i_r \in [1..r] \rightarrow \beta_{i_r} = \sum_{j=1}^{m_{\text{train}}} \left(\alpha^*_j - \alpha_j \right) y^{\text{train}}_{[:,j]}$$

De sorte que la reconstruction d'un nouveau point s'écrit :

$$\boxed{\forall i_r \in [1..r] \rightarrow \hat{a}^{(R)}_{[i_r,t]} = \sum_{j=1}^{m_{\text{train}}} \left(\alpha^*_j - \alpha_j \right) \left[y^{\text{train}}_{[:,j]} \right]^T y_{[:,t]}}$$

→ Formulation non linéaire

Dans le cas où une régression linéaire n'est pas pertinente dans l'espace initial, elle peut être pertinente dans un espace de dimension plus élevé. Pour ne pas calculer *explicitement* les transformations, on utilise une fonction noyau N qui permet de calculer les relations entre les mesures dans le nouvel espace [15]. La reconstruction de la composante i_r de l'état s'écrit alors :

$$\boxed{\forall i_r \in [1..r] \rightarrow \hat{a}^{(R)}_{[i_r,t]} = \sum_{j=1}^{m_{\text{train}}} \left(\alpha^*_j - \alpha_j \right) N \left(y^{\text{train}}_{[:,j]}, y_{[:,t]} \right)}$$

Pour déterminer les multiplicateurs de Lagrange, il suffit de modifier la matrice de covariance dans la fonction coût, en considérant :

$$Q_{\text{SVR},[t_1,t_2]} = N \left(y^{\text{train}}_{[:,t_1]}, y^{\text{train}}_{[:,t_2]} \right)$$

Dans le cas d'un noyau polynomial de degré q, la régression non linéaire dans l'espace de dimension p correspond à une régression linéaire dans l'espace de

14. La solution du problème primal est égal à la solution du problème dual. C'est bien le cas ici car le problème primal est convexe et que les contraintes sont affines (condition suffisante de Slater).

15. C'est le théorème de Mercer (1909). Une fonction noyau continue, symétrique et semi-définie positive peut s'exprimer comme un produit scalaire dans un espace de grande dimension.

dimension $q \times p$. La covariance se calcule selon :

$$Q_{\text{SVR},[t_1,t_2]} = \left(1 + \left\{y^{\text{train}}_{[:,t_1]}\right\}^T y^{\text{train}}_{[:,t_2]}\right)^q$$

Dans le cas d'un noyau gaussien d'échelle γ, la régression linéaire est effectué dans un espace de dimension infini [16] et la covariance se calcule selon :

$$Q_{\text{SVR},[t_1,t_2]} = \exp\left(-\gamma||y^{\text{train}}_{[:,t_1]} - y^{\text{train}}_{[:,t_2]}||^2\right)$$

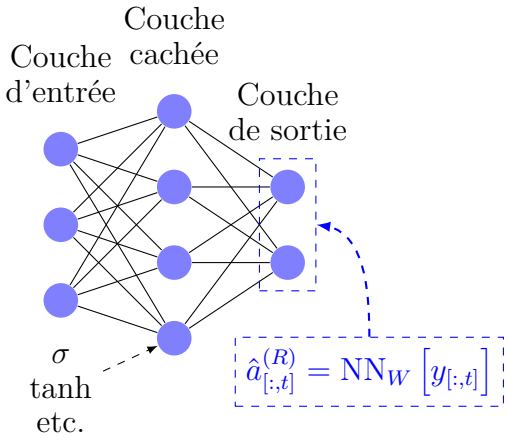
Remarque : le lecteur pourra trouver la dérivation complète des équations dans le tutoriel de Smola et Schölkopf (2004).

Réseau de neurones

Dans le chapitre précédent, on a introduit les réseaux de neurones et présenté le théorème d'universalité. Le pouvoir expressif des réseaux de neurones permet de réaliser des régressions non linéaires aussi complexes que souhaitées en jouant sur le nombre de couches, les fonctions d'activations et le nombre de neurones. Pour le problème de reconstruction, la fonction coût utilisée est la somme quadratique des résidus :

$$\mathcal{L}(W) = ||a^{\text{train}} - \text{NN}_W(y^{\text{train}})||^2_F$$

où W désigne l'ensemble des paramètres (poids et biais) du réseau et NN_W est l'opérateur de propagation avant



Lorsque l'ensemble de données est très riche, un réseau de neurones bien architecturé est un excellent interpolateur (Ott *et al.*, 2021). Pour éviter le sur-apprentissage, on peut utiliser la technique du *dropout* (Srivastava *et al.*, 2014) illustrée sur la figure 3.6. En se fixant un taux d'oubli, certains neurones sont désactivés *aléatoirement* durant l'apprentissage, ce qui améliore la généralisation [17].

Remarque 1 : la formulation du réseau de neurones est implicitement multi-tâche, avec p neurones d'entrées et r neurones de sortie.

Remarque 2 : dans ce manuscrit, seules des architectures superficielles seront considérées. Concernant les hyper-paramètres, nous n'optimiserons pas la taille du

16. Pour s'en convaincre, il suffit d'écrire la décomposition de Taylor de l'exponentielle.
17. Dis très simplement, le réseau apprend à utiliser tous ses neurones.

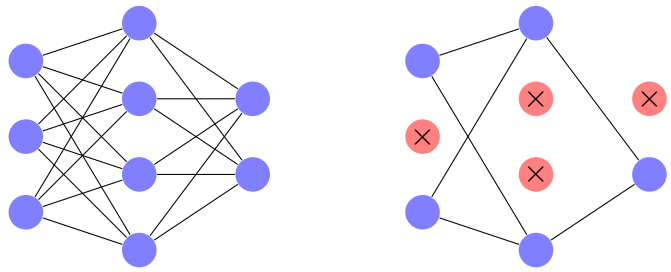


Figure *3.6* – *Illustration de la régularisation par oubli pour un réseau de neurones. Pendant l'apprentissage, une portion des neurones est aléatoirement neutralisée à chaque nouvelle descente de gradient.*

batch ni la vitesse d'apprentissage, qui seront les paramètres par défaut de Keras.

Remarque 3 : Erichson *et al.* (2020) utilise des réseaux de neurones superficiels pour reconstruire un écoulement. Sur l'écoulement laminaire autour d'un cylindre, les auteurs utilisent des mesures de pression pariétale pour reconstruire le champ de vorticité. Sur un écoulement turbulent homogène isotrope, le réseau effectue une tâche de *super-résolution* i.e. le champ de vitesse sur un maillage fin est estimée à partir du champ de vitesse sur un maillage très lâche. Les performances pour de l'extrapolation en temps (clichés de test *après* clichés d'entraînement) sont mauvaises.

Remarque 4 : dans Ott *et al.* (2021), nous avons utilisé des réseaux de neurones pour reconstituer un écoulement de jet à partir de mesures fils chauds. Pour une entrée (x, y, z, t), on dispose en sortie de la vitesse moyenne (de phase) et la vitesse RMS (de phase).

Synthèse

Dans cette sous-section, on a présenté trois formes de modèles paramétriques pour la reconstruction de l'état latent. L'objectif est d'apprendre les paramètres d'un modèle pour combiner linéairement ou non linéairement les composantes de la mesure et expliquer l'état latent.

3.4.3 | Estimation par prise de décision

L'apprentissage supervisé ne se limite pas aux formes paramétriques. Une autre possibilité est de reconstruire l'état latent à partir de *décisions* sur le vecteur mesure. Tout l'enjeu consiste à définir les bonnes règles de décision et à bien les utiliser. Dans la suite, on explicite ce qu'est un arbre de décision et comment on peut combiner des arbres dans le contexte de reconstruction. On précise que l'approche retenue est mono-tâche.

Arbre de décision

Dans le contexte de reconstruction, un arbre de décision régressif (Breiman *et al.*, 1984) estime $a_{[:,t]}$ à partir de règles *simples* appliquées sur les composantes de mesure $y_{[:,t]}$. Un tel arbre est défini par :

→ Des feuilles *terminales*. Une feuille terminale correspond à une estimation $\hat{a}^{(R)}_{[:,t]}$ possible. Les feuilles étant en nombre fini, les estimations possibles forment un ensemble de cardinal fini (à la différence d'une estimation paramétrique).

→ Des feuilles de *décision*. Au niveau d'une de ces feuilles, un choix binaire sur les composantes de $y_{[:,t]}$ permet d'aboutir sur une nouvelle feuille de décision ou une feuille terminale.

À titre d'exemple prototype, la figure 3.7 illustre un arbre de décision régressif avec une seule décision et deux feuilles terminales.

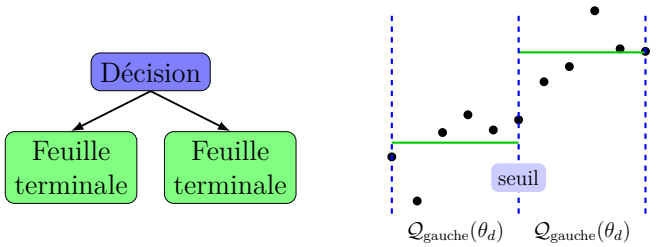


Figure 3.7 – *Arbre de décision prototype avec une seule décision. L'espace d'entrée est découpée en une partie gauche et une partie droite. Les feuilles terminales correspondent à la moyenne des données dans chaque région. Les notations sont précisées dans l'algorithme.*

On veut construire un arbre de décision ayant une profondeur d'au plus $d_{\max}$. L'espace d'entrée (donc de mesure) est découpé récursivement tant que la profondeur maximale n'est pas atteinte ou qu'il n'y ait plus d'échantillons disponibles [18]. Les seuils de décision sont optimisés en minimisant une fonction *d'impureté* qui évalue l'inhomogénéité des découpages. Plus l'impureté est faible, plus la répartition des points d'entraînement dans les régions gauche et droite est homogène. L'algorithme est le suivant :

Apprentissage d'un arbre de décision pour $\hat{a}^{(R)}_{[i_r,t]}$

→ **Au nœud k de l'arbre :**

— **Représenter les m_k données disponibles par l'ensemble $\mathcal{Q}_k$.**

18. Pour créer une feuille terminale, il faut au moins un échantillon d'apprentissage.

C'est un sous ensemble des données d'entraînement $(y^{\text{train}}, a^{\text{train}}_{[i_r,:]})$.

— **Construire la fonction d'impureté $\mathcal{I}$.**
Elle est définie par :

$$\mathcal{I}(\mathcal{Q}_k, \theta_d) = \frac{m_{\text{gauche}}}{m_k}\mathcal{X}\left(\mathcal{Q}_{\text{gauche}}(\theta_d)\right) + \frac{m_{\text{droite}}}{m_k}\mathcal{X}\left(\mathcal{Q}_{\text{droite}}(\theta_d)\right)$$

La variable θ_d contient un **seuil de décision** et le **capteur** sur lequel l'appliquer [a]. Suivant la décision, les données sont séparées en $\mathcal{Q}_{\text{gauche}(\theta_d)}$ et en $\mathcal{Q}_{\text{droite}(\theta_d)}$. La somme des entiers m_{gauche} et m_{droite} donne m_k.

La fonction $\mathcal{X}$ est l'erreur quadratique moyenne dans la région. Son expression, par exemple dans la région gauche, est :

$$\mathcal{X}\left(\mathcal{Q}_{\text{gauche}}(\theta_d)\right) = \sum_{a \in \mathcal{Q}_{\text{gauche}}(\theta_d)} (a - \bar{a}_{\text{gauche}})^2$$

où $\bar{a}_{\text{gauche}}$ est la moyenne des étiquettes dans la région. C'est la valeur que l'on choisirait pour $\hat{a}^{(R)}_{[:,t]}$ si $\mathcal{Q}_{\text{gauche}}(\theta_d)$ ne pouvait plus être découpé (feuille terminale)

— **Sélectionner les paramètres qui minimisent l'impureté.**
Il faut résoudre un problème d'optimisation avec une variable discrète (capteur ?) et une variable continue (seuil ?). On cherche :

$$\theta_d^* = \arg\min_{\theta_d} \left(\mathcal{Q}_k, \theta_d\right)$$

→ **Récursion sur les ensembles $\mathcal{Q}_{\text{gauche}}(\theta_d^*)$ et $\mathcal{Q}_{\text{droite}}(\theta_d^*)$.**

1. Les ensembles qui contiennent strictement moins d'échantillons qu'un nombre minimal [b] définissent des feuilles terminales. La valeur est donnée par la moyenne des étiquettes dans la région.
2. Les autres ensembles sont utilisés pour construire des décisions au nœud $k+1$ de l'arbre.
3. L'algorithme s'arrête s'il n'y a que des feuilles terminales. Si l'on atteint la profondeur maximale, tous les échantillons restants sont utilisés pour construire une feuille terminale.

a. Ce sont des paramètres. Le lecteur pourrait être perdu avec ce mot, sachant que les arbres de décision sont dit non paramétriques. Quand on parle de reconstruction paramétrique, on signifie que la sortie est expliquée par une combinaison *paramétrique* des entrées. Pour l'arbre de décision, la sortie est expliquée par *décisions* successives de mesures non paramétrées. Mais bien sûr, les seuils des décisions sont des paramètres.

b. On peut choisir 2, mais ça n'a pas grand intérêt.

Remarque 1 : l'algorithme de construction est *glouton* puisqu'un partage optimal est recherché à chaque nœud. Mais il n'y a aucune garantie que l'arbre final est *globalement* optimal [19].

Remarque 2 : un arbre de décision peut bien évidemment sur-apprendre. Pour éviter cela, on *l'élague* (*pruning* en anglais) en retirant certaines feuilles terminales et en remplaçant des décisions par des feuilles terminales.

Remarque 3 : un arbre de décision est souvent qualifié *d'instable* car un petit changement du vecteur d'entrée peut résulter en une succession de décisions bien différentes.

Remarque 4 : un arbre de décision est une boîte blanche car on peut remonter l'arbre pour comprendre une décision. Cela contraste avec le réseau de neurones qui est une boîte noire. Toutefois, on préfère souvent les réseaux de neurones car ils sont beaucoup plus rapides à évaluer.

Combinaison d'arbres

Un arbre de décision est rarement utilisé seul. Pour améliorer la décision, il est plus sage de combiner les avis de plusieurs arbres. Deux stratégies existent :

→ **Les forêts aléatoires**. On effectue un *boostrap* des données d'apprentissage c'est-à-dire que l'on crée plusieurs sous ensembles avec replacement possible. Sur chacun des sous ensembles, un arbre de décision est appris. Pour une nouvelle entrée $y_{[:,t]}$, chaque arbre de décision est évalué. La sortie qui reçoit le plus de votes est la sortie finale. On parle *d'ensachage* ou de *bagging* car les arbres de décision sont agrégés [20]. Les forêts aléatoires réduisent le sur-apprentissage en diminuant la variance de l'estimateur (Biau et Scornet, 2016). Pour une application en mécanique des fluides, on peut citer les travaux de Fukami *et al.* (2020) qui estiment les coefficients de forces et le champ de vorticité d'un cylindre à bas Reynolds.

→ **Adaboost**. Dans une forêt aléatoire, chaque arbre est créé indépendamment et n'a pas de profondeur prédéfinie. Dans adaboost, chaque arbre est composé d'un nœud et de deux feuilles (*stump* en anglais, pour dire souche) et se construit en prenant en compte les erreurs de l'arbre précédent. Chaque souche a une importance différente dans le vote final. Le terme adaboost

19. Considérons le problème de rendu de monnaie. L'objectif est de payer une somme en utilisant le moins de pièces possibles. L'algorithme glouton nous dit de prendre la pièce de valeur *maximale* possible à chaque étape. Si la somme à payer est de 6 et que le système de pièces est $(1, 3, 4)$, l'algorithme glouton nous dit de faire $4 + 1 + 1$ et donc d'utiliser trois pièces. Pourtant, il aurait été optimal de faire $3 + 3$. L'arbre de décision est construit sur un algorithme glouton car il minimise l'impureté à chaque étape, sans une vision globale. À noter : dans le système des pièces européens, on montre que l'algorithme glouton est toujours une solution optimale.

20. *Bagging* est la contraction de ***b****oostraping* et de ***agg****regat****ing***.

vient de ***ada**ptative **boost**ing* et renvoie au fait que des étiquettes mal reconstruites par un apprenant faible (une souche) sont *boostés* lors de la construction du prochain apprenant [21] (Freund et Schapire, 1997).

Pour le problème de reconstruction, on ne considérera pas ces deux méthodes d'ensemble. On se concentre sur une extension plus récente : les arbres de décision boostés par gradient.

Arbres de décisions boostés par gradient

Attention

Pour éviter les noms à rallonge, on parlera simplement d'arbres boostés même si ce nom est restrictif. On utilisera aussi l'acronyme GB pour *Gradient Boosting*

Comme pour adaboost, l'objectif est de construire *séquentiellement* des arbres en prenant en compte les erreurs des arbres précédents. Tous les arbres ont la même profondeur mais ne sont pas nécessairement des souches. Chaque arbre permet de corriger une reconstruction initiale représentée par une simple feuille terminale (Chen et Guestrin, 2016). La figure 3.8 donne un schéma prototype pour les arbres GB.

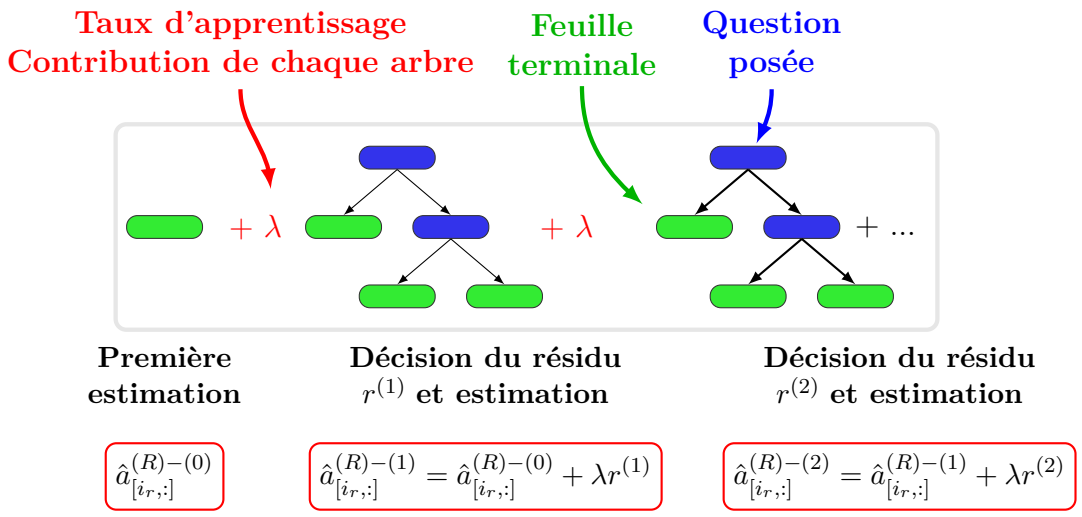


Figure *3.8* – *Principe des arbres de décisions boostés (par le gradient). Des arbres de décision permettent de corriger l'estimation fournie par une seule feuille. Les arbres sont construits séquentiellement et mis à l'échelle selon le même facteur.*

21. Pour la petite anecdote, les auteurs d'adaboost ont reçu le prestigieux prix Gödel en 2003 pour leur découverte.

Chaque arbre prend la décision d'un résidu pour améliorer l'estimation courante. Les résidus pour apprendre l'arbre sont calculés via le gradient d'une fonction coût, d'où le nom de la méthode. Si on note la fonction coût *différentiable* utilisée $\mathcal{L}$ et $(y^{\text{train}}, a^{\text{train}}_{[i_r,:]})$ l'ensemble d'apprentissage, l'algorithme est le suivant :

Algorithme du boosting par gradient

1. **Initialiser la fonction de reconstruction :**

$$\hat{a}^{(R)-(0)}_{[:,t]} = F_0(y_{[:,t]}) = \arg\min_{\gamma} \sum_{j=1}^{m_{\text{train}}} \mathcal{L}\left[a^{\text{train}}_{[i_r,j]}, \gamma\right]$$

   La solution de ce problème d'optimisation définit la feuille de base.

2. **Pour chaque arbre de décision (indice m) :**

   (a) **Calculer le gradient de l'erreur pour chaque échantillon d'apprentissage :**

$$\forall t \in [1, m_{\text{train}}] \rightarrow r_{t,m} = -\left\{\frac{\partial \mathcal{L}\left[a^{\text{train}}_{[i_r,t]}, F(y)\right]}{\partial F(y)}\right\}_{F(y)=F_{m-1}\left(y^{\text{train}}_{[:,t]}\right)}$$

   (b) **Apprendre l'arbre de décision pour décider du résidu $r_{t,m}$ à partir de $y_{[:,t]}$.**
   Il suffit d'appliquer l'algorithme récursif précédemment explicité.

   (c) **Créer les régions terminales $R_{j,m}$ pour $j = [1, J_m]$.**
   Ce sont les régions de l'espace de mesure qui ne seront plus découpées car le nombre minimal d'échantillons d'apprentissage est atteint ou la profondeur maximal de l'arbre est atteint.

   (d) **Synthétiser l'information dans les régions terminales** :

$$\forall j \in [1, J_m] \rightarrow \gamma_{j,m} = \arg\min_{\gamma} \sum_{y^{\text{train}}_{[:,t]} \in R_{j,m}} \mathcal{L}\left[a^{\text{train}}_{[i_r,t]}, F_{m-1}(y^{\text{train}}_{[:,t]}) + \gamma\right]$$

   La valeur γ est un scalaire qu'il faut ajouter à la reconstruction actuelle F_{m-1} pour l'améliorer.

(e) **Mettre à jour la fonction de reconstruction** :

$$\hat{a}_{[:,t]}^{(R)-(m)} = F_m(y_{[:,t]}) = F_{m-1}(y_{[:,t]}) + \lambda \sum_{j=1}^{J_m} \gamma_{j,m} \mathbb{I}(y_{[:,t]} \in R_{j,m})$$

Ici, $\mathbb{I}$ désigne la fonction indicatrice et renvoie 1 si $y_{[:,t]}$ est dans la région $R_{j,m}$ et 0 sinon.

Cet algorithme fait peur dans le cadre général. Pour ce qui nous concerne, on utilisera comme coût l'erreur quadratique moyenne. On a donc :

$$\mathcal{L}\left[a_{[i_r,t]}, F(y_{[:,t]})\right] = \frac{1}{2}\left(a_{[i_r,t]} - F\left[y_{[:,t]}\right]\right)^2$$

Dans ce cas, F_0 est la valeur moyenne de $a_{[i_r,:]}^{\text{train}}$, ce qui est une estimation certes naïve mais somme toute logique pour une nouvelle mesure $y_{[:,t]}$. Lors de la construction de l'arbre m, les résidus sont calculés selon :

$$\forall t \in [1..m_{\text{train}}] \rightarrow r_{t,m} = F_{m-1}\left[y_{[:,t]}^{\text{train}}\right] - a_{[i_r,t]}^{\text{train}}$$

Les valeurs terminales dans les régions terminales correspondent au résidu moyen. Une fois les M arbres appris, la fonction d'estimation est donnée par F_M. La valeur λ est un hyper-paramètre appelé vitesse d'apprentissage. Des preuves empiriques suggèrent qu'une petite valeur de λ (beaucoup de petits pas mais dans la bonne direction) diminue la variance du modèle. C'est donc un paramètre de régularisation.

Synthèse

Les arbres de décision boostés permettent de combiner des arbres experts pour prendre une décision d'ensemble. C'est une approche non paramétrique qui peut être utilisée dans le cadre de la reconstruction. L'objectif est d'apprendre les règles de décisions pour décider de l'état latent à partir du vecteur de mesure.

3.4.4 | La validation croisée des modèles pour gagner en robustesse

Les méthodes paramétriques (régression linéaire, par vecteurs supports ou par réseau de neurones) et non paramétrique (arbres boostés) présentées sont construites avec des *hyper-paramètres*. Il s'agit de degrés de libertés qui contrôlent la flexibilité de la méthode. Il est essentiel pour l'utilisateur de bien *régler* cette liberté afin d'assurer un compromis entre le biais et la variance du modèle qui sera appris. Classiquement, on choisit **des** combinaisons de candidats, on apprend autant de modèles que de combinaisons puis on sélectionne **la** combinaison qui mène à l'erreur de généralisation la plus faible. Mais il y a un problème : l'erreur de généralisation

est l'erreur sur de nouvelles données, non utilisées pendant l'apprentissage. Cela ne peut pas être des données de test car par définition, ces données interviennent uniquement dans le *déploiement* du modèle et certainement pas dans sa *conception*. Il faut donc introduire un nouvel ensemble, que l'on appelle ensemble de *validation*.

La validation par blocs

Pour estimer l'erreur de généralisation sans utiliser les données de test, l'approche *naïve* consiste à diviser le set d'entraînement en un nouveau set d'entraînement et un set de validation. Le découpage classique 80/20 et sans mélange est illustré sur le diagramme de la figure 3.9. En pratique, estimer l'erreur de généralisation avec *un seul bloc* n'est pas idéal. Si la structure temporelle des données n'importe pas [22], on peut définir plusieurs blocs de validation et estimer l'erreur de généralisation comme l'erreur de validation moyenne. On parle de validation croisée par blocs (Fushiki, 2011) dont l'algorithme est le suivant :

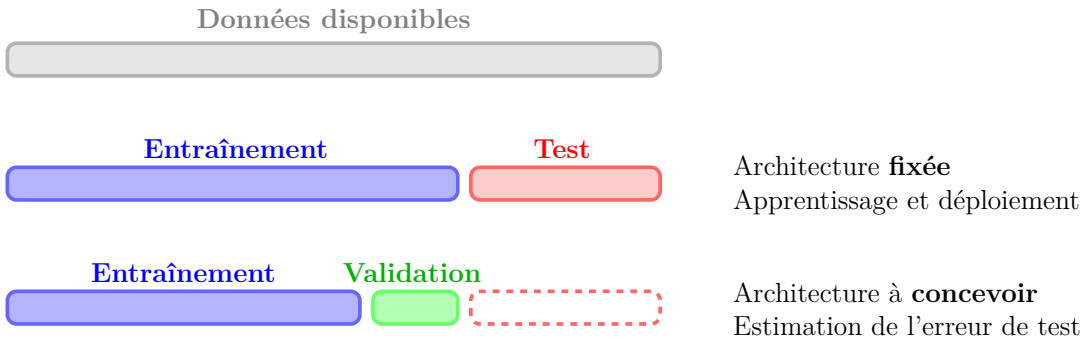


Figure *3.9* – *Stratégie naïve de conception d'un modèle. Pour évaluer la qualité d'un hyper-paramètre, l'erreur de généralisation est estimée sur un ensemble de validation. Si l'erreur sied, l'architecture du modèle est fixée, le modèle est appris sur tout l'ensemble d'entraînement puis il est déployé. On peut alors calculer l'erreur de test.*

Validation croisée par blocs

Pré-requis : les données disponibles ont été divisées en données d'entraînement et données de test. Une forme de modèle a été choisie et on souhaite régler ses hyper-paramètres. Pour une combinaison d'hyper-paramètres, on calcule l'erreur de validation selon :

22. Ce qui est le cas dans le problème de RECONSTRUCTION, vu qu'on cherche à apprendre la relation entre une mesure à un instant donné et l'état latent au même instant. On peut donc mélanger l'ensemble des données, ça n'aura que peu d'influence. En revanche, on ne pourra pas le faire dans le problème de prédiction.

1. **Diviser les données d'entraînement en K_v blocs de taille égale**.

2. **Définir K_v partitions différentes entraînement - validation**.

   Pour $K = 3$ (défaut), on dispose de trois partitions différentes. L'ensemble de validation contient 33% des données.

3. **Pour chaque partition disponible (indice k)** :

   (a) Apprendre le k-ième modèle.
   On utilise les données d'entraînement i.e. la partie bleue.

   (b) Évaluer le k-ième modèle.
   On utilise les données de validation i.e. la partie verte. On peut avoir une vision *erreur* et dans ce cas, on utilise l'erreur quadratique moyenne. On peut aussi avoir une vision *score* et dans ce cas, on utilise le coefficient de détermination.

4. **Évaluer la qualité de généralisation**. Il suffit de moyenner les erreurs (ou les scores) de validation. Ce résultat reflète la qualité de généralisation du modèle construit avec la combinaison d'hyper-paramètres choisie.

Conclusion : lors de la conception, on évalue la qualité de généralisation pour différentes combinaisons d'hyper-paramètres. L'erreur minimale (ou le score maximal) est obtenue pour les hyper-paramètres optimaux. C'est ceux qu'on utilisera pour apprendre le modèle final avec **toutes** les données d'entraînement.

On dispose donc d'un moyen de valider les hyper-paramètres. Encore faut-il savoir quelles combinaisons candidates considérer initialement.

Le choix des hyper-paramètres candidats

Selon la méthode de reconstruction, le choix des hyper-paramètres candidats n'est pas effectuée de la même manière. Pour les méthodes de reconstruction, on considère trois approches.

→ **Recherche par grille**. Les candidats sont choisis dans une grille définie par l'utilisateur. C'est la méthode utilisée pour la *régression linéaire*, où le seul hyper-paramètre à valider est la contribution LASSO α.

→ **Recherche aléatoire**. Les candidats sont échantillonnés aléatoirement dans l'espace des hyper-paramètres. C'est la méthode utilisée pour la régression par vecteurs supports et pour les arbres boostés.

- Pour la *régression par vecteurs supports*, il faut optimiser la taille de la boîte (C_B, échantillonné d'une distribution exponentielle [23] d'échelle 15) et le noyau (polynomial de degré 2 à 10 ou gaussien d'échelle échantillonné d'une distribution exponentielle d'échelle 0.01). La taille de la marge ϵ_m est égale à 0.1 (valeur par défaut).
- Pour les *arbres boostés*, il faut optimiser la vitesse d'apprentissage (λ, échantillonné d'une distribution uniforme $[0, 1]$), le nombre d'estimateurs (entier aléatoire entre 10 et 50), le nombre minimal de données dans une feuille (entier aléatoire entre 5 et 15) et la profondeur maximal (entier entre $0.1p$ et $0.8p$).

Au total, 25 combinaisons aléatoires sont testées. Cela permet de trouver la région optimale des hyper-paramètres ($\pm 5\%$) avec une probabilité de 70% (voir la figure 3.10 et Bergstra et Bengio (2012))

→ **Optimisation génétique**. Une population initiale de modèles (construits avec des hyper-paramètres aléatoires) est évoluée sur plusieurs générations. À la génération g, les modèles ayant l'erreur de généralisation la plus faible survivent et deviennent des *parents*. Les autres modèles ne survivent pas. La population de la génération $g + 1$ est complétée de modèles *enfants*, construits avec les hyper-paramètres de deux modèles parents. Des *mutations* peuvent aussi avoir lieu. Une optimisation génétique [24] est utilisée pour la reconstruction par *réseau de neurones*. On considère sept générations de quinze réseaux de neurones. Il faut optimiser le nombre de couches cachées (1, 2 ou 3 pour rester superficiel), le nombre de neurones dans chaque couche ($r, 2r, p, p/2$ ou $r + p$, des choix purement empiriques), les fonctions d'activation (relu, tanh, sigmoïde ou linéaire), le taux d'oubli ($0.0, 0.1, 0.2$ ou 0.3) et la vitesse d'apprentissage ($10^{-4}, 10^{-3}, 10^{-2}$ ou 0.1).

Quelques remarques

On a présenté un moyen de valider les hyper-paramètres. Quatre remarques importantes doivent être faites.

Remarque 1 : la validation croisée présentée est *non-imbriquée*. Le découpage initial des données est arbitraire donc la performance du modèle (estimée sur les données de test) sera biaisée. Il faudrait plusieurs découpages entraînement - test pour avoir une estimation non biaisée. On parlerait de validation *imbriquée* mais

23. Densité de probabilité définie par $d(t) = 0$ si $t < 0$ et $d(t) = s \exp(-st)$ sinon. Le réel s est appelée *échelle*. Il n'y a pas de raison particulière pour échantillonner les hyper-paramètres selon cette loi ; on souhaite simplement des hyper-paramètres positifs pas trop grand.

24. Précisons les paramètres par défaut. Lorsqu'une population est apprise, 40% des meilleures combinaisons d'hyper-paramètres sont retenues et 60% sont détruites (étape de *sélection*). Il y a toutefois 10% de chance qu'un mauvais modèle survive. Pour compléter la population, les modèles enfants sont construits sur la base de deux survivants, sélectionnés aléatoirement. Pour chaque degré de liberté, l'enfant hérite aléatoirement de l'attribut du père ou de la mère (étape de *reproduction*. Les enfants peuvent subir dans 20% des cas une modification aléatoire de ses hyper-paramètres (étape de *mutation*).

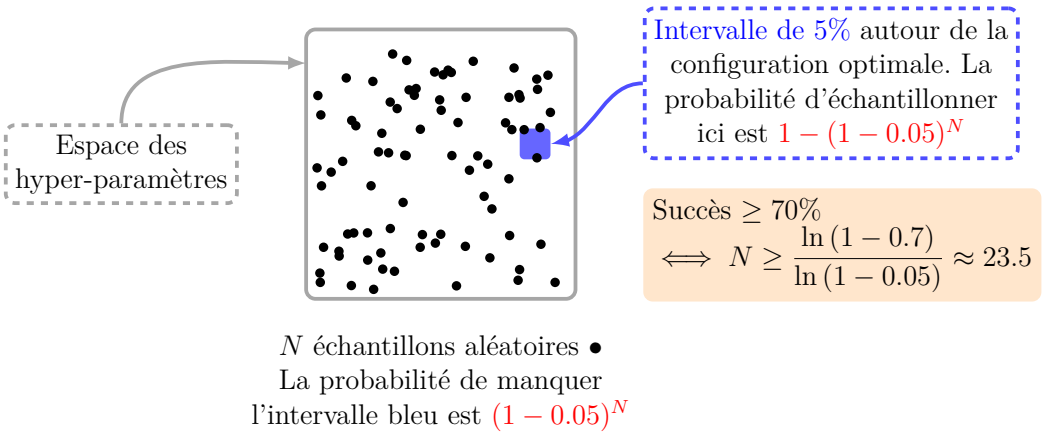


Figure *3.10* – *Recherche aléatoire dans une grille. N combinaisons d'hyper-paramètres sont tirés aléatoirement. Avec $N = 25$, la probabilité d'échantillonner dans la région optimale (à plus ou moins 5%) est supérieure à 70%.*

c'est un procédé très long et coûteux (Varoquaux *et al.*, 2017).

Remarque 2 : l'apprentissage n'est pas effectué sur les données brutes. Pour faciliter la convergence et accélérer l'apprentissage, les modèles sont appris sur les données normalisées (variance unitaire, moyenne nulle). La variance et la moyenne à utiliser pour la reconstruction de nouvelles données sont celles des données d'entraînement.

Remarque 3 : la méthode quasi-systématiquement utilisée dans la littérature pour le choix des hyper-paramètres est l'optimisation bayésienne (Brochu *et al.*, 2010). Il s'agit d'approximer par un processus gaussien le coût de validation en fonction des hyper-paramètres. À l'étape k, l'hyper-paramètre le plus incertain est exploré (évaluation du coût) et le processus (moyenne et écart type) est mis à jour. La méthode a fait ces preuves pour les modèles à beaucoup de degrés de libertés mais le procédé est très long. Par exemple pour un réseau de neurones, une évaluation de la fonction coût requiert trois apprentissages (approche trois blocs). De plus, les modèles peuvent être instables comme observés par (Lui et Wolf, 2019) pour un problème de décrochage dynamique.

Remarque 4 : notre approche de l'optimisation génétique est *très* basique, avec des candidats pris dans des ensembles discrets. Il faudrait utiliser des algorithmes évolutionnaires plus élaborés, par exemple avec adaptation selon la covariance (Igel *et al.*, 2007).

Une fois les hyper-paramètres optimisés et le modèle final appris, on peut le déployer. Il est toutefois important de pouvoir estimer les incertitudes de reconstruction.

3.4.5 | Quantifier l'incertitude de reconstruction

Attention

Ce que l'on présente dans cette partie est à titre informatif. Cela sera très rapidement traité dans la partie résultats du manuscrit.

Dans le cadre d'une régression linéaire moindres carrés, on sait estimer l'incertitude sur les paramètres et l'incertitude sur la reconstruction d'un nouveau point. Ce travail est purement statistique, basé sur la construction de tests. Avec des modèles plus exotiques (pénalisation LASSO ou autre forme de modèle), il n'existe pas de méthode universelle. Ici, on propose une méthode applicable à *n'importe quel* modèle régressif et une méthode spécifiquement applicable aux réseaux de neurones.

Méthode générale

Cette méthode est inspirée de Qiu *et al.* (2019). Pour chaque composante de l'état latent (méthode mono-tâche), un *processus gaussien* permet d'estimer le résidu de reconstruction à partir de la reconstruction, comme illustrée sur la figure 3.11.

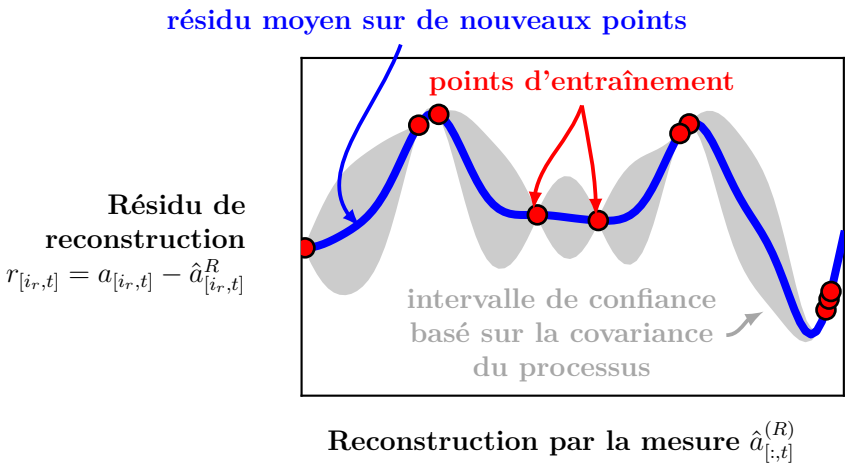


Figure *3.11* – *Pour chaque composante de l'état latent, un processus gaussien permet d'estimer le résidu de reconstruction. Les points rouges sont des couples d'entraînement utilisés pour calculer la moyenne (courbe bleu) et la variance (intervalles gris) a posteriori du processus.*

Par définition, un processus gaussien est un processus stochastique telle que

toute collection des étiquettes suit une loi normale multivariée [25]. Si on modélise le résidu $r_{[i_r,t]} = a_{[i_r,t]} - \hat{a}^{(R)}_{[i_r,t]}$ par un processus gaussien de moyenne nulle, on peut donc écrire pour N points quelconques :

$$\begin{bmatrix} r_{[i_r,t_1]} \\ \vdots \\ r_{[i_r,t_N]} \end{bmatrix} \sim \mathcal{N}\left(0_{\mathbb{R}^N} \; ; \; k\left[\hat{a}^{(R)}_{[:,t_i]}, \hat{a}^{(R)}_{[:,t_j]}\right]_{(i,j)\in[1..N]^2}\right)$$

La fonction k est appelée *noyau* et mesure les similarités entre les reconstructions aux temps t_i et t_j. Pour le choix classique du noyau gaussien de paramètres σ et γ, l'expression est :

$$\boxed{k_{[t_i,t_j]} = k\left[\hat{a}^{(R)}_{[:,t_i]}, \hat{a}^{(R)}_{[:,t_j]}\right] = \sigma^2 \exp\left[-\gamma \left|\left|\hat{a}^{(R)}_{[:,t_i]} - \hat{a}^{(R)}_{[:,t_j]}\right|\right|_2^2\right]}$$

Le premier objectif est de déterminer les paramètres du noyau. Pour cela, on observe les données $(\hat{a}^{(R),\text{train}} \in \mathbb{R}^{r\times m_{\text{train}}}, r^{\text{train}}_{[i_r,:]} \in \mathbb{R}^{m_{\text{train}}})$ et on construit la fonction de *vraisemblance* des paramètres sachant les données :

$$\boxed{\mathcal{L}(k^{\text{train}} \mid r^{\text{train}}_{[i_r,:]}) = \frac{1}{(2\pi)^{\frac{m_{\text{train}}}{2}} |k^{\text{train}}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\left\{r^{\text{train}}_{[i_r,:]}\right\}^T \left\{k^{\text{train}}\right\}^{-1} \left\{r^{\text{train}}_{[i_r,:]}\right\}\right]}$$

Ici, $k^{\text{train}} \in \mathbb{R}^{m_{\text{train}}\times m_{\text{train}}}$ est la variable (variance paramétrée calculée sur $\hat{a}^{(R),\text{train}}$) et $|.|$ est l'opération du déterminant. Les paramètres optimaux sont ceux qui maximisent cette vraisemblance et on peut les calculer par descente de gradient. Le second objectif consiste à calculer la distribution *a posteriori* d'un nouveau point. Par application de la règle de Bayes, on démontre [26] que pour un nouveau point $\hat{a}^{(R)}_{[:,t]}$:

25. On considère deux variables aléatoires X et Y telle que $Y = f(X)$. On suppose que f est un processus gaussien. Dans ce cas, pour x_i donné, l'étiquette y_i est échantillonné selon une loi normale. Pour une collection quelconque $x_1, ...x_n$, la collection des étiquettes $y_1, ..., y_N$ suit une loi gaussienne multivariée de dimension N. Le processus est défini par une fonction moyenne et une fonction variance, que l'on cherche à apprendre à partir de couples d'entraînement (x_i, y_i) donnés.

26. Donnons les idées principales de la démonstration. Par hypothèse du processus gaussien, la collection de N points (r^*, r) suit une gaussienne multivariée. En supposant la moyenne nulle, on peut donc écrire $\begin{bmatrix} r^* \\ r \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} ; \begin{bmatrix} \Sigma_{[r^*,r^*]} & \Sigma_{[r^*,r]} \\ \Sigma_{[r,r^*]} & \Sigma_{[r,r]} \end{bmatrix}\right)$. La distribution *a priori* (jointe) des étiquettes est $p(r^*, r^*) = \frac{1}{(2\pi)^{n/2}|\Sigma|^2} \exp\left(-\frac{1}{2}\begin{bmatrix} r^* \\ r \end{bmatrix}^T \begin{bmatrix} \Sigma_{[r^*,r^*]} & \Sigma_{[r^*,r]} \\ \Sigma_{[r,r^*]} & \Sigma_{[r,r]} \end{bmatrix}^{-1} \begin{bmatrix} r^* \\ r \end{bmatrix}\right)$. L'inverse de la matrice de covariance est appelée matrice de précision, que l'on définit par les blocs Γ. Lorsque les points r sont observés, on peut calculer la distribution conditionnelle sur les points r^* sachant $r = r^{\text{train}} = r_0$. Sans se préoccuper du terme marginal et du terme constant selon $\Gamma_{r,r}$, on peut écrire $p(r^* \mid r_0) \propto \exp\left(-\frac{1}{2}[r^*]^T\Gamma_{[r^*,r^*]}r^* - r^T\Gamma_{[r^*,r]}r_0\right)$. La suite repose sur des astuces calculatoires. D'abord, on écrit $\Gamma_{[r^*,r]} = \Gamma_{[r,r]}\Gamma^{-1}_{[r,r]}\Gamma_{[r^*,r]}$. Puis, on factorise l'exponentielle par

$$\begin{cases} \bar{r}_{[i_r,t]} = k^{t-\text{train}} \left[k^{\text{train}}\right]^{-1} r^{\text{train}}_{[i_r,:]} \\ \text{cov}\left(r_{[i_r,t]}\right) = k^{t-t} - k^{t-\text{train}} \left[k^{\text{train}}\right]^{-1} k^{\text{train}-t} \end{cases}$$

avec les dimensions $k^{t-\text{train}} \in \mathbb{R}^{1\times m_{\text{train}}}$ et $k^{t-t} \in \mathbb{R}^{1\times 1}$. La première covariance indique les similitudes entre le nouveau point et toutes les observations d'entraînement, de sorte que des points proches (covariance maximale) auront des résidus proches.

Finalement, on dispose d'une méthode générale pour estimer les incertitudes de reconstruction pour n'importe quelle méthode régressive. Plusieurs remarques importantes doivent être faites.

Remarque 1 : les processus gaussiens utilisés sont multi-caractéristiques mais mono-tâche. Il existe des méthodes pour étendre aux cas multi-tâches (et donc expliquer les r résidus simultanément) mais cela impose de définir des matrices de *co-régionalisation* pour modéliser la dépendance des résidus.

Remarque 2 : en géo-statistiques, on utilise le terme de *krigeage* pour parler d'une modélisation par processus gaussien. Dans les approches multi-tâches, on parle de *co-kirgeage*.

Remarque 3 : dans l'article de Qiu *et al.* (2019), le noyau est construit sur les similitudes en entrée (les reconstructions) *et* les similitudes en sortie (le résidu sur la composante i_r). Ils parlent de noyau in/out.

Remarque 4 : Les processus gaussien ont une complexité en $\mathcal{O}([m^{\text{train}}]^3)$ à cause de l'inversion de k^{train}. On utilisera seulement et aléatoirement 20% des données d'entraînement pour accélérer le calcul.

Remarque 5 : la moyenne du résidu permet de mettre à jour la reconstruction selon $\hat{a}^{(R),\text{màj}}_{[:,t]} = \hat{a}^{(R)}_{[i_r,t]} + \bar{r}_{[i_r,t]}$. L'intervalle de confiance à 95% autour de la reconstruction améliorée est $\hat{a}^{(R),\text{màj}}_{[:,t]} \pm 1.96\sqrt{\text{cov}\left(r_{[i_r,t]}\right)}$.

$\exp\left(\frac{1}{2} r_0 \Sigma_{[r,r^*]} \Gamma^{-1}_{[r^*,r^*]} \Gamma_{[r^*,r^*]} \Gamma^{-1}_{[r^*,r^*]} \Gamma_{[r^*,r]} r_0\right)$. Ce facteur ne faisant pas apparaître r^*, c'est juste un terme de proportionnalité. L'exponentielle restante traduit que $r^*|r = r_0$ suit une loi normale de moyenne $\Gamma^{-1}_{[r^*,r^*]}\Gamma_{[r^*,r]} r_0$ et de variance $\Gamma_{[r^*,r^*]}$. Avec de l'algèbre linéaire, on peut facilement expliciter les blocs Γ en fonction des blocs Σ. La moyenne se réécrit alors $\bar{r^*} = \Sigma_{[r^*,r]}\Sigma^{-1}_{[r,r]} r_0$ et la covariance $\Sigma_{[r^*,r^*]} - \Sigma_{[r^*,r]}\Sigma^{-1}_{[r,r]}\Sigma_{[r,r^*]}$. Cette seconde matrice est appelée *complément de Schur* de $\Sigma_{[r,r]}$ dans la matrice de covariance complète.

Intervalles par dropout

Lorsque l'on apprend un réseau de neurones, on évite le sur-apprentissage en ajoutant des couches de *dropout*. De cette manière, une fraction des neurones est aléatoirement neutralisée à chaque descente de gradient. Lorsque le modèle est déployé, les couches d'oubli sont classiquement désactivées. C'est bien dommage car en laissant le dropout actif pour la reconstruction d'un nouvel état latent, on peut construire un ensemble de reconstructions. Les premiers et derniers des 40-quantiles définissent (pour chaque composante de l'état latent) un intervalle de confiance à 95% autour de la reconstruction moyenne [27]. Ce principe est illustré sur la figure 3.12.

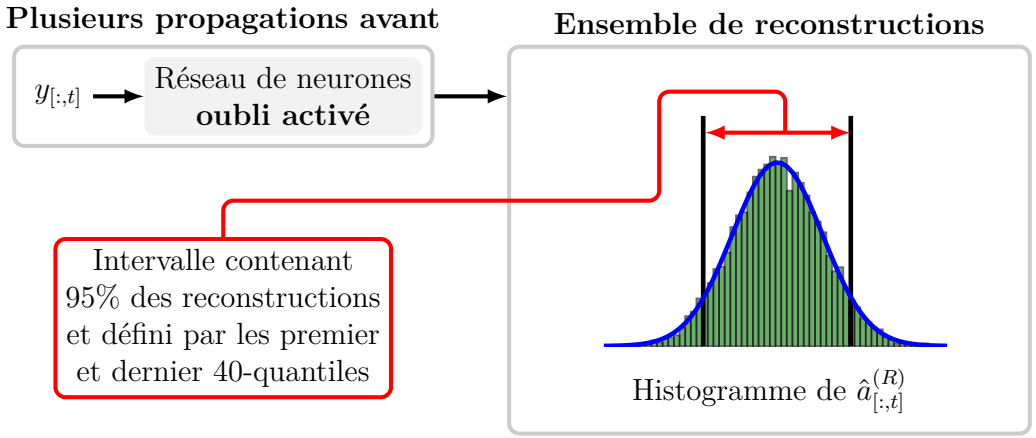


Figure *3.12* – *Estimation de l'incertitude de reconstruction à partir d'un ensemble de reconstructions.*

La difficulté de la méthode consiste à utiliser le *bon* taux d'oubli. Un taux important (proche de 100%) mènerait à des reconstructions très diverses et donc un large intervalle de confiance. Un taux faible (proche de 0%) mènerait à des reconstructions très similaires et donc de petits intervalles de confiance. Pour optimiser le taux, on suit l'argument heuristique suivant : sur les données d'entraînement, l'intervalle de confiance à ci% doit contenir ci% des valeurs de a^{train}. L'algorithme d'optimisation du taux est donc :

Optimisation du taux d'oubli

1. **Définir des taux d'oubli**.
   Il s'agit d'une grille de valeurs entre 0 et 0.6. Au delà de 60% d'oubli, les

27. Toutes les reconstructions sont triées. Un q quantile est chacune des $q-1$ valeurs qui divisent les reconstructions en q parts égales, de sorte que chaque partie représente $1/q$ de l'échantillon. Pour construire un intervalle de confiance à 95%, on calcule les 40 quantiles, de sorte que chaque partie représente 2.5% des données. Entre le premier et le dernier quantiles, on dispose de 95% des données.

performances du modèle seraient trop mauvaises (sous-apprentissage).

2. **Définir des niveaux de confiance**.
   Il s'agit d'une grille de valeurs entre 0% et 100%.
3. **Pour chaque taux d'oubli** :
   (a) **Pour chaque niveau de confiance ci%** :
      i. **Pour chaque mesure d'entraînement** $y^{\text{train}}_{[:,t]}$:
         A. **Faire un ensemble de reconstructions**.
            On effectue 250 reconstructions. Propager un vecteur de mesure dans un réseau de neurones étant rapide, l'obtention de l'ensemble a un faible coût calculatoire.
         B. **Déterminer les q-quantiles**.
            Chaque région doit contenir $(50 - 0.5\text{ci})\%$ des données.
         C. **Construire l'intervalle de confiance à ci%**.
            On utilise les premier et dernier q-quantiles.
      ii. **Évaluer les intervalles de confiance** :
         On dispose de m_{train} intervalles de confiance, un pour chaque reconstruction. On compte le nombre de clichés réels a^{train} effectivement à l'intérieur de l'intervalle de confiance associé.
   (b) **Évaluer la qualité du taux d'oubli** :
      On trace le pourcentage de clichés d'entraînement à l'intérieur des intervalles des confiance, en fonction du niveau de confiance.
4. **Sélectionner le taux d'oubli** :
   On choisit celui dont la courbe obtenue en 3.b) est la plus proche d'une droite passant par (0,0) et (1,1).

Cette méthode est potentiellement applicable à d'autres formes de modèles que le réseau de neurones. Il faudrait simplement étendre le concept de *dropout*.

Finalement, on dispose de plusieurs méthodes pour estimer un état latent à partir de sa mesure. Il ne reste plus qu'à quantifier la qualité des reconstructions.

3.5 | Qualité des reconstructions

La qualité des reconstructions est quantifiée à la fois dans l'espace latent et dans l'espace physique. On désigne par $a^b \in \mathbb{R}^{r \times m_b}$ un batch de données et $\hat{a}^{(R)-b}$ sa reconstruction.

Dans l'espace latent

Dans l'espace latent, on utilise le coefficient de détermination. Il s'agit d'un *score* défini pour chaque composante de l'état latent. Le vecteur $R^2 \in \mathbb{R}^r$ est tel que :

$$\forall i_r \in [1..r],\ R^2_{i_r} = 1 - \frac{\sum_{t=1}^{m_b} \left[a^b_{[i_r,t]} - \hat{a}^{(R)-b}_{[i_r,t]}\right]^2}{\sum_{t=1}^{m_b} \left[a^b_{[i_r,t]} - \bar{a}^b_{i_r}\right]^2}$$

Ici, $\bar{a}^b \in \mathbb{R}^r$ désigne la moyenne empirique des clichés de l'état latent. Lorsque $R^2_{i_r}$ est proche de l'unité, la reconstruction de la contribution du mode i_r est *bonne* en moyenne : la variabilité du batch reconstruit est semblable à la variabilité attendue. Le score moyen informe de la qualité globale (i.e. pour tous les modes) de reconstruction dans l'espace latent.

Dans l'espace physique

D'un point de vue pratique, c'est l'erreur d'estimation de l'état complet qui est intéressante. On utilise l'erreur quadratique moyenne normalisée telle qu'introduite dans le chapitre de réduction. Ainsi, on définit $\mathrm{NMSE} \in \mathbb{R}^n$ tel que :

$$\forall i_n \in [1..n], \mathrm{NMSE}_{i_n} = \frac{\sum_{t=1}^{m_b} \left(u^b_{[i_n,t]} - d\left[\hat{a}^{(R)-b}_{[i_r,t]}\right]\right)^2}{\sum_{t=1}^{m_b} \left(u^b_{[i_n,t]} - \bar{u}^b_{i_n}\right)^2}$$

Comme pour le coefficient de détermination, l'erreur globale correspond à une moyenne sur toutes les composantes de l'état. Cette erreur est comprise entre 0 et 1, la borne inférieure étant celle souhaitée.

3.6 | Conclusion du chapitre

Le problème de reconstruction consiste à estimer l'état de haute dimension du système à partir de sa mesure. En supposant le décodeur connu, l'estimation peut porter uniquement sur l'état latent.

La première question concerne la placement des capteurs. On suppose que ce sont des signaux connus de l'état complet. Pour optimiser la position de ces signaux, on utilise un algorithme de partitionnement amélioré en deux étapes. D'abord on détermine les cellules de Voronoï du maillage puis on définit les centres des partitions les plus énergétiques comme capteurs. Cette méthode semi-supervisée a l'avantage d'être systématique mais elle reste critiquable. En particulier, le nombre de partitions n'est pas optimisé et on n'évite pas les multi-colinéarités. De plus, aucune information dynamique n'est prise en compte. Des approches plus élaborées faisant intervenir les méthodes de troncature équilibrée (Manohar *et al.*, 2018b) pourraient être envisagées.

La seconde question concerne la méthode d'estimation de l'état latent. Deux solutions ont été proposées. **La première** est l'estimation directe où l'état latent

est reconstruit par une combinaison linéaire des modes de décodage. Une telle estimation requiert la résolution d'un problème d'optimisation, construit sur des considérations d'algèbre linéaire. Conceptuellement, cela revient à estimer l'état latent *pour* la mesure donnée, sans chercher à apprendre l'opération de passage complète. Les coefficients de la combinaison linéaire étant différents pour chaque nouvelle estimation, ce n'est pas une méthode adaptée pour des reconstructions rapides. **La seconde** méthode est l'estimation régressive où les outils de l'apprentissage supervisé sont extensivement utilisés. L'objectif est d'apprendre *complétement* le passage entre l'espace de mesure et l'espace latent. De cette manière, une nouvelle estimation ne nécessite que l'évaluation du modèle. Afin d'assurer un compromis biais-variance, différentes formes de modèles hyper-paramétrables sont utilisées. Le tableau 3.2 donne la carte d'identité de chacune de ces méthodes.

Finalement, on dispose d'un moyen pour estimer l'état latent courant à partir de la mesure de l'état complet au même instant. Or dans une application pratique, les mesures peuvent ne pas être disponibles à chaque instant. Pour une estimation en temps réel, il est donc nécessaire de *prédire* l'état entre deux reconstructions. Apprendre un modèle dynamique pour l'état latent, c'est l'objectif de la **prédiction** que l'on traite au chapitre 4.

Méthode	Régression linéaire	Vecteurs supports	Réseau de neurones	Arbres boostés
Description	Les mesures expliquent linéairement les composantes de l'état latent	Régression linéaire dans un espace de dimension plus élevé	Neurones interconnectés avec des activations non linéaires	Arbres de décisions entraînés séquentiellement sur la base des erreurs des arbres précédents
Multi-tâches ?	Oui et non	Non	Oui	Non
Paramètres	β	α	poids et biais	Seuils de décision
Régularisation	LASSO	Contrainte de la boîte	Dropout	Vitesse d'apprentissage
Hyper-paramètres	Coefficient Lasso	Noyau Taille de la boîte	Taux d'oubli Nombre de couches cachées Fonctions d'activations Taux d'apprentissage	Nombre d'arbres Profondeur maximale d'un arbre Vitesse d'apprentissage
Optimisation des hyper-paramètres	Recherche dans une grille	Recherche aléatoire	Algorithme génétique	Recherche aléatoire
Avantages	Méthode simple Multi-tâche possible	Astuce du noyau	Flexibilité	Partition de l'espace de mesures
Désavantages	Trop simple ?	r régressions	Boîte noire très (trop) hyper-paramétrable	r regressions Instabilité

Tableau 3.2 *– Carte d'identité des méthodes d'estimation régressives.*

4. Prédiction de l'état latent par l'opérateur de Koopman

Dans ce chapitre, on s'intéresse à la prédiction du futur de l'état latent du système. Tout d'abord, on introduit les problématiques générales liées aux systèmes dynamiques et on formalise le problème de prédiction. Puis, on présente les aspects théoriques de l'opérateur de Koopman, un outil permettant de décrire un système dynamique fini non linéaire sous la forme d'un système dynamique linéaire de dimension infinie. Ensuite, on propose différents algorithmes pour obtenir une approximation finie de l'opérateur de Koopman. De cette manière, on disposera d'un modèle linéaire fini (i.e. une matrice) interprétable via une décomposition spectrale pour prédire le futur état latent du système. Enfin, on présente la stratégie d'assimilation de données pour mettre à jour les prédictions du modèle dynamique à partir de reconstructions ponctuelles.

Sommaire

4.1 | Formalisation de la prédiction

Attention

La science des systèmes dynamiques est extrêmement riche et complexe, avec des raisonnements mathématiques qui sortent du cadre de ce manuscrit. Ce chapitre donne quelques éléments de compréhension pour introduire ce qui nous intéresse : l'opérateur de Koopman.

Par ailleurs, tout le travail est effectué sur l'état latent du système, pas l'état de haute dimension. Comme pour le problème de reconstruction, il suffit d'appliquer le décodeur pour repasser dans l'espace de haute dimension.

Formulation générale

L'état latent du système est noté $\boldsymbol{a} \in \mathcal{M}$ avec $\mathcal{M}$ une variété inclue dans $\mathbb{R}^r$. La dynamique de l'état latent s'écrit de manière *très* générale :

$$\frac{d}{dt}\boldsymbol{a}(t) = \boldsymbol{f}_d(\boldsymbol{a}(t),\ t;\ \boldsymbol{\mu})$$

avec $\boldsymbol{f}_d$ un champ de vecteurs qui dépend de l'état courant, du temps et d'un ensemble de paramètres $\boldsymbol{\mu}$. Un exemple est donné en note de bas de page [1]. Une représentation si générique cache de nombreuses problématiques résumées ci-dessous :

→ Les équations du système peuvent être inconnues [2]. En mêlant les techniques d'apprentissage automatique avec des approches parcimonieuses, on sait aujourd'hui identifier des modèles interprétables et généralisables mais les applications restent assez académiques (Brunton *et al.*, 2016b).

→ Le système peut être non linéaire et la solution peut évoluer sur une variété topologiquement complexe. Ne pas pouvoir écrire la solution comme une combinaison de fonctions de référence est un frein dans certaines applications,

1. Pour le système de Lorenz 63, l'état est $\boldsymbol{a} = (x, y, z)$ et les paramètres sont $\boldsymbol{\mu} = (\sigma, \rho, \beta)$. Le modèle dynamique est défini par

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

C'est un modèle très simplifié de l'atmosphère, basé sur le phénomène de convection de Rayleigh Bénard. La dynamique est *bifurcative* car elle change en fonction des paramètres. Pour $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, une particule évolue sur un attracteur dont la forme ressemble aux ailes d'un papillon. Cette forme particulière est une *variété*, obtenue en continuant dans le domaine non linéaire les vecteurs propres de la dynamique linéarisée aux points d'équilibres.

2. C'est par exemple le cas du cerveau ou du climat pour lesquels on ne sait pas appliquer la deuxième loi de Newton. De manière générale, on manque de lois physiques pour les systèmes modernes.

notamment pour du contrôle (Kaiser *et al.*, 2018). Bien que le théorème d'Hartman-Grobman [3] établit quand et où une dynamique non linéaire peut être approximée par une dynamique linéaire, l'analyse *globale* d'un système non linéaire est souvent qualitative. L'analyse spectrale de Koopman est une piste prometteuse pour comprendre les systèmes non linéaires avec une vision linéaire (Bagheri, 2013). C'est ce que l'on étudie principalement dans ce chapitre.

→ Le système peut être chaotique [4] et intermittent [5]. Des opérateurs simples ne permettent pas de représenter de tels phénomènes. Des coordonnées retardées (Brunton *et al.*, 2017) ou des décompositions multi-résolutions sont prometteuses (Kutz *et al.*, 2016b).

→ La dynamique peut être multi-échelles. Le placement optimal de capteurs pour ces systèmes est un sujet actif de recherche, le travail se faisant très souvent de manière heuristique ou par force brute (toutes les possibilités sont testées une par une). Certaines techniques de l'acquisition comprimée (état de l'art, section 1.4.1) sont prometteuses (Manohar *et al.*, 2018a).

Flot de la dynamique

Supposons que le système soit *autonome* [6]. On s'intéresse à l'évolution d'une condition initiale $\boldsymbol{a}(t_0)$ sur un horizon de temps Δt. Pour obtenir cette trajectoire, il faut intégrer la condition initiale en utilisant le *flot* de la dynamique. Le flot est défini formellement par $\boldsymbol{F}_{\Delta t}$ tel que :

$$\boldsymbol{F}_{\Delta t}\left[\boldsymbol{a}(t_0)\right] = \boldsymbol{a}(t_0) + \int_{t_0}^{t_0+\Delta t} \boldsymbol{f}_d\left[\boldsymbol{a}(\tau)\right] d\tau$$

Très souvent, l'intégration de la fonctionnelle $\boldsymbol{f}_d$ (lorsqu'elle est connue) est faite numériquement.

Vision discrète

Le flot de la dynamique permet de définir le système dynamique *discret* :

$$\boldsymbol{a}_{k+1} = \boldsymbol{F}_{\Delta t}\left[\boldsymbol{a}_k\right]$$

où l'indice k indique la solution à l'instant $k\Delta t$. En choisissant un pas de temps Δt

3. Autour d'un point d'équilibre hyperbolique, le système non linéaire se comporte comme le système linéarisé. Un point hyperbolique est un point d'équilibre dont les parties réelles des valeurs propres du jacobien ne sont pas nulles.

4. La configuration *papillon* de Lorenz 63 est un exemple de système chaotique. En effet, deux conditions initiales très proches peuvent avoir des trajectoires bien différentes. D'où le fameux *effet papillon*.

5. Comme en turbulence. Par exemple, la fumée d'une cigarette présente des zones laminaires et turbulentes intermittentes. Un écoulement de vent est aussi intermittent, ce qui se traduit par les rafales.

6. Le système s'écrit $\dot{\boldsymbol{a}}(t) = \boldsymbol{f}_d[\boldsymbol{a}(t)]$ i.e. dépendance uniquement à l'état courant.

tel que le pas de temps sans dimension Δt^* vaut l'unité [7], on construit le système :

$$\boxed{a_{[:,t^*+1]} = \boldsymbol{F}_{\Delta t^*}\left[a_{[:,t^*]}\right]}$$

Pour ne pas alourdir les notations, les astérisques ne seront plus indiquées. Les clichés $a_{[:,t]} \in \mathbb{R}^r$ introduits dans les chapitres précédents correspondent à la réalisation de ce système discret.

Problème de prédiction

Le problème de prédiction consiste à approximer le flot $\boldsymbol{F}_{\Delta t}$ à partir des clichés a^{train}. Deux approches sont possibles. **La première approche** se base sur les techniques de prédiction de séries temporelles. Les outils privilégiés sont les modèles auto-régressifs ou les modèles boîte noire du type *réseau de neurones récurrents*. Pour cette seconde méthode, des fonctions d'activations élaborées utilisent l'histoire de l'état pour prédire le futur de l'état. Un travail complet s'appuyant sur cette technique est proposé dans l'annexe G avec le système chaotique de Lorenz 63. **La seconde approche** est basée sur l'opérateur de Koopman dont on cherche une approximation. On dispose alors d'un modèle *linéaire fini* pour avancer dans le temps l'état du système. La suite du chapitre se consacre à cette approche.

4.2 | L'opérateur de Koopman : aspects théoriques

En 1931, Bernard Koopman a démontré l'existence d'un opérateur linéaire de dimension *infinie* permettant d'avancer avec le flot de la dynamique *toutes* les mesures de l'état du système associé (Koopman, 1931). Pour le système discret qui nous intéresse, cet opérateur se note $\mathcal{K}_{\Delta t}$ et se définit par :

$$\boxed{\forall g : \mathbb{R}^r \to \mathbb{R},\ \mathcal{K}_{\Delta t} g = g \circ \boldsymbol{F}_{\Delta t}}$$

où g est une fonction d'observation prise dans un espace de Hilbert, par exemple celui de toutes les fonctions de carré intégrable. Pour une telle fonction, la dynamique s'écrit :

$$g\left(a_{[:,t+1]}\right) = \mathcal{K}_{\Delta t} g\left(a_{[:,t]}\right)$$

Il y a un échange de difficultés, illustré avec le schéma 4.1. La dynamique initiale était certes non linéaire mais elle était de dimension finie. Dans l'espace de tous les observables, la dynamique est maintenant linéaire mais de dimension infinie, ce qui

7. La définition dépend du système. Dans le cas d'un écoulement, on considère $\Delta t^* = \dfrac{U \Delta t}{L}$ où Δt est le pas de temps entre deux clichés, U est la vitesse caractéristique (par exemple la vitesse *free-stream*) et L est la longueur caractéristique (par exemple la dimension d'un obstacle). Δt^* est un temps sans dimension qui caractérise l'instationnarité de l'écoulement. Pour construire les clichés, on choisit Δt tel que $\Delta t^* = 1$.

facilite l'interprétation (utilisation des fonctions propres) mais pas la représentativité. Tout l'enjeu consiste à définir une approximation finie de l'opérateur pour pouvoir le représenter par une matrice.

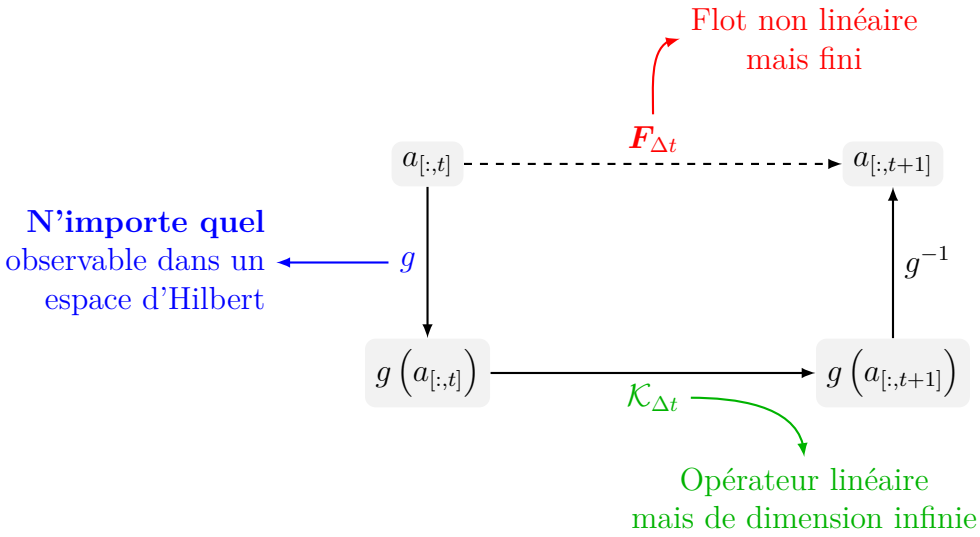


Figure *4.1* – *Vision initiale (trait pointillé) et vision de Koopman (trait plein) d'un système dynamique.*

Remarque : d'un point de vue continu, si la variété $\mathcal{M}$ est *lisse* et que $\boldsymbol{f}_d$ est différentiable en tout point, on peut faire tendre Δt vers 0. On définit le *générateur infinitésimal* de la famille des opérateurs de Koopman et il existe le système continu $\dot{g} = \mathcal{K}g$ (Klus *et al.*, 2020).

4.2.1 | Décomposition spectrale

L'opérateur $\mathcal{K}_{\Delta t}$ étant linéaire, on peut effectuer une décomposition spectrale. La fonction propre φ_j associée à la valeur propre λ_j satisfait :

$$\boxed{\mathcal{K}_{\Delta t}\varphi_j\left[a_{[:,t]}\right] = \lambda_j\varphi_j\left[a_{[:,t]}\right]}$$

En partant d'une condition initiale, on peut donc écrire :

$$\forall k \in \mathbb{N},\ \varphi_j\left[a_{[:,t+k]}\right] = \lambda_j^k\varphi_j\left[a_{[:,t]}\right]$$

D'un point de vue continu, la fonction propre est associée à la valeur propre ω_j telle que :

$$\frac{d}{dt}\varphi_j\left[a_{[:,t]}\right] = \omega_j\varphi_j\left[a_{[:,t]}\right]$$

Ce qui permet d'avoir la trajectoire :

$$\forall k \in \mathbb{N},\ \varphi_j \left[a_{[:,t+k]}\right] = \varphi \left[a_{[:,t]}\right] \exp(\omega_j \times k \times 1)$$

où on indique explicitement $\times 1$ pour rappeler la présence du pas temps, choisi égal à l'unité vaut l'unité. On peut donc relier la valeur propre *discrète* à la valeur propre *réelle* selon :

$$\boxed{\omega_j = \frac{\ln \lambda_j}{1}}$$

Les fonctions propres de l'opérateur de Koopman définissent une base privilégiée pour décrire l'évolution de n'importe quel observable g. On peut écrire :

$$\forall g : \mathbb{R}^r \to \mathbb{R},\ \forall k \in \mathbb{N},\ g\left(a_{[:,t]}\right) = \sum_{j=0}^{\infty} \lambda_j^k \varphi_j \left[a_{[:,t]}\right] (\varphi_j | g)$$

où la dernière quantité représente le produit scalaire canonique [8] dans l'espace d'Hilbert des observables. Cette décomposition, introduite soixante dix ans après l'opérateur [9], permet de caractériser *globalement* le système dynamique (Mezić et Banaszuk, 2004; Bagheri, 2013).

Remarque : en appliquant une dérivation à la chaîne, on observe que les fonctions propres vérifient l'équation aux dérivées partielles $\nabla \varphi_j(\boldsymbol{a}).\boldsymbol{f}_d(\boldsymbol{a}) = \lambda_j \varphi_j(\boldsymbol{a})$.

4.2.2 | Espace d'observables invariant

Si on souhaite capturer l'évolution de *toutes* les observations possibles de l'état latent (dans un espace d'Hilbert), il faut considérer *toutes* les fonctions propres de l'opérateur de Koopman. D'un point de vue pratique, cela pose naturellement des problèmes de représentativité et de calcul numérique. L'idée consiste alors à approximer l'évolution de *quelques* observables seulement. Ceux-ci vivent dans un espace de dimension fini décrit par un sous ensemble de l fonctions $g_1, ..., g_l$. Deux cas de figure sont possibles :

Cas 1 → Le sous espace $\mathcal{S} = \text{Vect}(g_1, ..., g_l)$ est invariant par l'opérateur de Koopman. Ainsi, $\forall g \in \mathcal{S},\ \mathcal{K}_{\Delta t} g \in \mathcal{S}$. La restriction de $\mathcal{K}_{\Delta t}$ à l'espace $\mathcal{S}$ est une représentation *finie* de l'opérateur de Koopman mais uniquement valide sur les fonctions dans $\mathcal{S}$. Un sous ensemble de fonctions propres est invariant par construction, mais leur détermination n'est pas triviale.

Cas 2 → Le sous espace $\mathcal{S}$ n'est pas invariant par l'opérateur de Koopman. Dans ce cas, le meilleur modèle linéaire (au sens moindres carrés) pour décrire linéairement la dynamique des observables est une approximation finie de l'opérateur de Koopman. Les vecteurs propres de la matrice permettent

8. Par exemple pour les fonctions de carrés intégrables, le produit scalaire canonique est $(g_1|g_2) = \int_{\mathcal{M}} g_1(\boldsymbol{a}) g_2(\boldsymbol{a}) d\boldsymbol{a}$

9. L'opérateur n'a suscité que peu d'intérêts après son introduction en 1931. Les applications pratiques voient le jour avec la science des données, où les méthodes régressives permettent enfin d'identifier des fonctions propres (Lusch *et al.*, 2018).

d'identifier des fonctions propres mais celles-ci sont *fallacieuses* (i.e. erronées) et ne vérifient pas l'équation aux dérivées partielles des fonctions propres de Koopman.

Dans la sous-section suivante, on donne trois exemples pour illustrer ces deux cas. Ensuite, on présentera différentes méthodes d'approximation de l'opérateur de Koopman.

Remarque : le premier cas est *idéal* d'un point de vue théorique mais pas d'un point de vue pratique. En effet, c'est surtout la dynamique dans l'espace *latent* qui est intéressante pour avoir des prédictions interprétables. Si on connaît un sous espace d'observables invariants mais que le passage vers l'espace latent est difficile, ce n'est pas idéal.

4.2.3 | Trois exemples formateurs

Les aspects théoriques de l'opérateur de Koopman sont illustrés sur des systèmes dynamiques simples. Le traitement est ici analytique et sert à motiver l'utilisation d'approches orientées données pour l'approximation finie de $\mathcal{K}_{\Delta t}$.

Le premier exemple illustre la configuration souhaitée : on connaît un sous espace invariant par Koopman. On dira que l'approximation est *fermée*.

Exemple 1 - ce que l'on souhaite

On considère le système dynamique non linéaire suivant :

$$\frac{d}{dt}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \mu a_1 \\ \lambda(a_2 - a_1^2) \end{bmatrix}$$

Pour $\lambda < \mu < 0$, une condition initiale finit par atterrir sur la variabilité parabolique [a] définie par $a_2 = a_1^2$. Avec les cinq fonctions d'observations $(g_1, g_2, g_3, g_4, g_5) = (a_1, a_2, a_1^2, a_1 a_2, a_1^3)$, le système est augmenté [b] selon :

$$\frac{d}{dt}\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix} = \underbrace{\begin{bmatrix} \mu & 0 & 0 & 0 & 0 \\ 0 & \lambda & -\lambda & 0 & 0 \\ 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & \mu+\lambda & -\lambda \\ 0 & 0 & 0 & 0 & 3\mu \end{bmatrix}}_{\mathrm{K}} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

La matrice K est la restriction de l'opérateur de Koopman au sous espace généré par les cinq observables considérés. On peut faire la prédiction dans ce sous espace et repasser sans problèmes dans l'espace initial car a_1 et a_2 font partis des observables. **C'est un cas idéal**.

a. Le système dynamique a pour seul point fixe $(0, 0)$. Le jacobien du système en $(0, 0)$ est donné par $J(0,0) = \begin{bmatrix} \mu & 0 \\ 0 & \lambda \end{bmatrix}$. Un vecteur propre associé à la valeur propre μ est $(1, 0)$

et un vecteur propre associé à la valeur propre λ est $(0, 1)$. Si la valeur propre λ est plus rapide que la valeur propre μ (i.e. $\lambda < \mu < 0$ donc $e^{\lambda}t$ tend plus vite vers 0 que $e^{\mu}t$), l'état adhère rapidement (via a_2) à la variabilité parabolique $a_2 = a_1^2$ puis évolue lentement (via a_1) vers le point fixe.

b. Par exemple avec l'observable g_5, on peut écrire $\dot{g_5} = \dot{a_1^3} = 3a_1^2\dot{a_1} = 3a_1^2 \times \mu a_1 = 3\mu a_1^3 = 3\mu g_5$. C'est une simple application de la dérivation en chaîne.

Le second exemple illustre ce qui arrive lorsque l'on considère *trop* d'observables et que le système augmenté est toujours non linéaire. On dira que l'approximation est *ouverte*.

Exemple 2 - ce qu'il peut arriver

On reprend le même système dynamique que précédemment mais avec les neuf observables $(g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9) = (a_1, a_2, a_1^2, a_1a_2, a_2^2, a_1^3, a_1^2a_2, a_1a_2^2, a_2^3)$. Dans ce cas, le système dynamique augmenté s'écrit :

$$\frac{d}{dt}\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \\ g_9 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & -\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2\mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu+\lambda & -\lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\lambda & 0 & -2\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 3\mu & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda+2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\lambda+\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3\lambda \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \\ g_9 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\lambda a_1^4 \\ -2\lambda a_1^3 a_2 \\ -3\lambda a_1^2 a_2^2 \end{bmatrix}$$

Le sous espace généré par les neufs observables n'est pas invariant par l'opérateur de Koopman. Le système dynamique augmenté est toujours non linéaire. Dans une approche orientée données (formalisée dans la section suivante), on ferait une régression linéaire entre les données à t et les données à $t + 1$. Bien évidemment, le modèle appris serait erroné et correspondrait à la matrice tri-bande aliasée par le terme bleu. Il faudrait identifier les observables responsables de l'ouverture, par exemple avec des approches parcimonieuses (voir Kaiser *et al.* (2017) qui traite cet exemple en détail).

Le troisième exemple montre que des observables polynomiaux ne sont pas toujours le meilleur choix d'observables.

Exemple 3 - Un exemple non trivial

On considère le système dynamique $\dot{x} = x^2$. Avec les observations $g_1 = x$ et $g_2 = x^2$, le système augmenté s'écrit :

$$\frac{d}{dt}\begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} g_1 \\ g_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 2x^3 \end{bmatrix}$$

On tente donc d'ajouter l'observation $g_3 = x^3$ mais le résultat est identique : on continue d'ouvrir le système avec un degré supérieur. **Il n'y a donc aucune série de Taylor qui puisse être une fonction propre de l'opérateur**.

Si maintenant on considère l'observation $g = \exp(-1/x)$, le système se réécrit $\dot{g} = g$. La dynamique est représentée linéairement mais l'intuition pour trouver l'observable n'est pas triviale. De plus, le passage vers l'espace initial n'est connu que par un traitement analytique.

Remarque : on peut montrer que toutes les fonctions propres de l'opérateur pour ce système dynamiques sont des séries de Laurent. Pour la valeur propre $\lambda \in \mathbb{C}$, les fonctions propres associées sont proportionnelles à :

$$\varphi(x) = 1 - \lambda x^{-1} + \frac{\lambda^2}{2}x^{-2} - \frac{\lambda^3}{3}x^{-3} + ... = \exp(-\lambda/x)$$

Ces trois exemples permettent de mieux définir le travail d'approximation finie de l'opérateur de Koopman. **Soit** il faut déterminer *automatiquement* les bonnes observations de l'état en assurant un retour dans l'espace latent. **Soit** il faut partir d'une bibliothèque d'observables et éliminer ceux susceptibles d'ouvrir l'approximation. Dans tous les cas, une fois les observations fixées, l'opérateur de Koopman est approximé par une régression linéaire entre les observations à t et les observations à $t+1$. Dans la suite, on formalise ce travail et on étudie différents choix d'observables.

Remarque : pour des systèmes dynamiques non linéaires qui possèdent plusieurs points fixes, il est impossible d'avoir une approximation finie de l'opérateur de Koopman incluant les composantes de l'état comme observables (Brunton *et al.*, 2016a).

4.3 | Approximation finie de l'opérateur

Dans cette partie, on formalise l'approximation finie de l'opérateur de Koopman puis on propose une vision orientée données. On propose alors différents algorithmes pour aboutir à différentes approximations de l'opérateur.

Cadre général

On considère l fonctions d'observations $g_1, ..., g_l$ et on note $\mathcal{S} = \text{Vect}(g_1, ..., g_l)$ le sous espace généré par ces fonctions. On peut donc écrire :

$$\forall i \in [1..l],\ g_i\left(a_{[:,t]}\right) = \sum_{j=1}^{l} \delta_i^j g_j\left(a_{[:,t]}\right) = \sum_{j=1}^{l} g_j\left(a_{[:,t]}\right)[\boldsymbol{e}_j]_i$$

où δ_i^j est le symbole de Kronecker et $[\boldsymbol{e}_j]$ est le j-ième vecteur de la base canonique de $\mathbb{R}^p$. Cette écriture indique que la fonction d'observation $g_i : \mathbb{R}^r \to \mathbb{R}$ est identifiable au vecteur $\boldsymbol{e}_i$. Le sous espace $\mathcal{S}$ n'étant pas nécessairement invariant, l'avancement par le flot introduit un résidu r_i tel que :

$$\boxed{\mathcal{K}_{\Delta t} g_i\left(a_{[:,t]}\right) = \sum_{j=1}^{l} \alpha_{[i,j]} g_j\left(a_{[:,t]}\right) + r_i\left(a_{[:,t]}\right)}$$

avec $r_i \notin \mathcal{S}$. En considérant les l fonctions d'observation, on peut écrire la relation matricielle :

$$\boxed{G^{(+1)}(a_{[:,t]}) = KG(a_{[:,t]}) + R(a_{[:,t]})}$$

avec $K \in \mathbb{R}^{l \times l}$ la matrice telle que $K_{[i,j]} = \alpha_{[i,j]}$, $G(a_{[:,t]}) \in \mathbb{R}^l$ les observations courantes, $G^{(+1)}(a_{[:,t]}) = G(a_{[:,t+1]})$ les observations un pas de temps dans le futur et $R(a_{[:,t]}) \in \mathbb{R}^l$ les résidus. La matrice K est une *approximation finie* de l'opérateur de Koopman. Les vecteurs propres **gauche** [10] permettent d'identifier l fonctions propres de l'opérateur de Koopman :

$$\boxed{\textbf{Décomposition spectrale} \rightarrow \xi K = \Lambda \xi \implies \Phi(a_{[:,t]}) = \xi G(a_{[:,t]})}$$

où $\Lambda \in \mathbb{C}^{l \times l}$ contient les valeurs propres de K et $\xi \in \mathbb{C}^{l \times l}$ contient les vecteurs propres unitaires associés (rangés par ligne). La matrice $\Phi(a_{[:,t]}) \in \mathbb{R}^l$ est l'évaluation des *fonctions propres* identifiées pour l'état $a_{[:,t]}$. Deux cas de figure sont possibles.

Cas 1 → Le résidu R vaut $0_{\mathbb{R}^l}$ quel que soit l'état $a_{[:,t]}$. Dans ce cas, la matrice K est la restriction de $\mathcal{K}_{\Delta t}$ au sous espace $\mathcal{S}$. Les fonctions propres identifiées sont des fonctions propres *exactes* de l'opérateur.

Cas 2 → Le résidu n'est pas nul. Dans ce cas, les fonctions propres identifiées sont *fallacieuses* i.e. erronées [11] et ne vérifient pas l'équation aux dérivées partielles des fonctions propres. Autrement dit, les fonctions identifiées n'ont pas la même évolution que celle prédite par les valeurs propres associées et il existe un horizon de prédiction h tel que $\Phi(a_{[:,t+h]}) \neq \Lambda^h \Phi(a_{[:,t]})$

10. Attention à ne pas se tromper ! Certains articles utilisent directement les vecteurs propres droits mais c'est parce que les clichés qui serviront à apprendre K sont rangés par *lignes* ; c'est notamment le cas de Williams *et al.* (2015) qui introduisent la décomposition modale dynamique étendue en mécanique des fluides. Dans ce manuscrit, les clichés seront toujours rangés par colonne.

11. On rencontre le mot *spurious* en anglais.

Apprentissage de K

On suppose que les l fonctions d'observations sont fixées. Dans ce cas, on peut estimer la matrice K par une *régression linéaire*. L'algorithme est le suivant :

Approximation orientée données de $\mathcal{K}_{\Delta t}$

1. **Considérer une trajectoire de l'état et sa version décalée.**

   On dispose de m_{train} clichés organisés dans la matrice a^{train}. La tranche $[1 \to m_{\text{train}} - 1]$ donne la matrice A tandis que la tranche $[2 \to m_{\text{train}}]$ donne la matrice $A^{(+1)}$. Ces deux matrices contiennent chacune $m_{\text{train}} - 1$ clichés de r composantes. Les clichés sont rangés par *colonnes*.

2. **Observer les clichés contenus dans A et $A^{(+1)}$.**

   On crée les matrices $G = G[A]$ et $G^{(+1)} = G[A^{(+1)}]$. Chaque matrice contient $m_{\text{train}} - 1$ clichés de l composantes.

3. **Trouver K tel que $G^{(+1)} \approx KG$.**

   La solution moindres carrés utilise l'inverse de Moore Penrose et s'écrit :

$$K = G^{(+1)}G^{\dagger} = G^{(+1)}G^{T}\left[GG^{T}\right]^{-1}$$

   En utilisant des méthodes parcimonieuses du type LASSO, on limite la corruption de la matrice K par le résidu.

   Remarque : si on ne fait pas attention au sur-apprentissage, le modèle linéaire peut parfaitement décrire la dynamique des observations d'entraînement. Cela ne signifie pas que les fonctions d'observations forment un sous espace invariant. Il faut tester le modèle sur une nouvelle trajectoire pour statuer de la qualité des observables.

Prédiction récursive

La matrice K peut être utilisée pour prédire **les observations** plusieurs pas de temps dans le futur. Il suffit d'écrire la décomposition spectrale $K = \xi^{-1}\Lambda\xi$ puis d'itérer sur l'horizon h. On parle de prédiction *récursive* telle que :

$$\textbf{Prédiction observables} \to \widehat{G}^{(+h)}\left(a_{[:,t]}\right) = K^{h}G\left(a_{[:,t]}\right) = \xi^{-1}\Lambda^{h}\xi G\left(a_{[:,t]}\right)$$

Ainsi, $\widehat{G}^{(+h)}\left(a_{[:,t]}\right)$ est une approximation des observations au temps $t+h$. Dans le cas où $\mathcal{S}$ est invariant, les fonctions propres identifiées sont *exactes* et quelque soit l'horizon, on a $\xi\widehat{G}^{(+h)}\left(a_{[:,t]}\right) = \xi G\left(a_{[:,t+h]}\right)$. Dans le cas contraire, les fonctions identifiées sont *fallacieuses* car la dynamique réelle des observables est non linéaire. Cette remarque est illustrée avec la figure 4.2 où l'on reprend le deuxième exemple de la section 4.2.3.

Pour une prédiction de l'état, il faut estimer les poids de passage entre l'espace des observables et l'espace latent. En reprenant les notations de l'algorithme, on cherche la matrice $B \in \mathbb{R}^{r\times l}$ telle que $A = BG$. Avec une régression linéaire classique, on obtient

$$\boxed{B \approx B_{[\mathrm{MC}]} = AG^{\dagger}}$$

où l'acronyme MC signifie *moindres carrés*. La prédiction récursive dans l'espace d'état s'écrit alors :

$$\boxed{\textbf{Prédiction état} \rightarrow \hat{a}^{(P)}_{[:,t+h]} = B_{[\mathrm{MC}]}\widehat{G}^{(+h)}\left(a_{[:,t]}\right) = B_{[\mathrm{MC}]}\xi^{-1}\Lambda^{h}\xi G\left(a_{[:,t]}\right)}$$

où la notation (P) fait référence au terme *prédiction* comme la notation (R) faisait référence à la *reconstruction* dans le chapitre précédent. La terminologie est importante :

- $\rightarrow$ Λ est une matrice diagonale qui contient les valeurs propres de la matrice K. Dans le cas général, ce sont des complexes qui viennent en paires conjuguées car K est à coefficient réels [12].
- $\rightarrow$ ξ^{-1} est une matrice (l,l) dont les colonnes sont appelées *modes dynamiques*. Ce sont les directions principales du mouvement (linéaire) dans l'espace des observables [13].
- $\rightarrow$ $B_{[\mathrm{MC}]}\xi^{-1}$ est une matrice (r,l) dont les colonnes sont appelées *modes de Koopman*. Ce sont les directions principales du mouvement dans l'espace

12. Si $\boldsymbol{v}$ est un vecteur propre associé à la valeur propre λ alors $K\boldsymbol{v} = \lambda v$. Le passage au conjugué donne $\bar{K}\bar{\boldsymbol{v}} = \bar{\lambda}\bar{\boldsymbol{v}}$ mais K est à coefficients réels donc $\bar{K} = K$. Ainsi, $\bar{\boldsymbol{v}}$ est un vecteur propre associé à la valeur propre $\bar{\lambda}$.

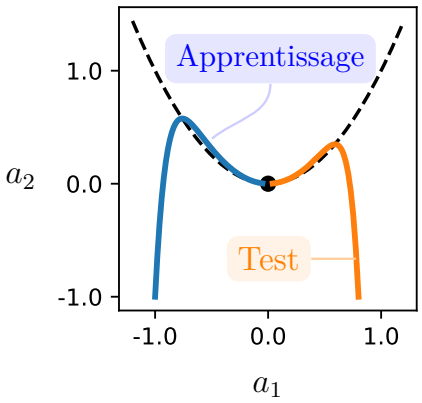
13. Le vecteur des observations évolue selon des directions principales (associées aux valeurs propres réelles) et des rotations (fixées par les valeurs propres complexes). Dans le cas général, si $\boldsymbol{v} = \boldsymbol{v}_r + i\boldsymbol{v}_i$ est un vecteur propre associé à la valeur propre $\lambda = \lambda_r + i\lambda_i$, on peut écrire $K\boldsymbol{v} = \lambda\boldsymbol{v}$ et identifier les parties réelles et imaginaires. On parvient à :

$$\begin{cases} K\boldsymbol{v}_r = \lambda_r\boldsymbol{v}_r - \lambda_i\boldsymbol{v}_i \\ K\boldsymbol{v}_i = \lambda_i\boldsymbol{v}_r + \lambda_r\boldsymbol{v}_i \end{cases}$$

En notant $\mathcal{V}$ la matrice de deux colonnes $(\boldsymbol{v}_r, \boldsymbol{v}_i)$, la vision réelle est $K\mathcal{V} = \mathcal{V}\mathcal{E}$ où $\mathcal{E}$ est la matrice de rotation $\begin{bmatrix} \lambda_r & \lambda_i \\ -\lambda_i & \lambda_r \end{bmatrix}$. Avec une représentation polaire de la valeur propre, on fait apparaître l'angle de rotation et la vitesse de rotation autour du vecteur nul. Si le module $|\lambda|$ est plus grand que l'unité, le mouvement est amplifié. S'il est plus petit que l'unité, le mouvement est amorti.

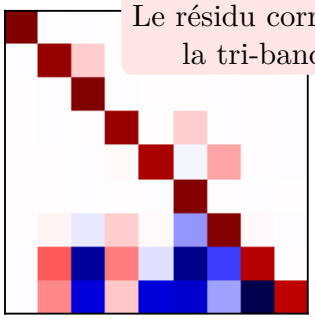
Étape 1
Intégration numérique de deux conditions initiales pour le système dynamique

$$\frac{d}{dt}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -0.1a_1 \\ -(a_2 - a_1^2) \end{bmatrix}$$



Étape 2
Observation de la trajectoire d'entraînement et régression $G^{(+1)} = KG$

Neuf observations
$a_1, a_2, a_1^2, a_1a_2, a_2^2$
$a_1^3, a_1^2a_2, a_1a_2^2, a_2^3$



Étape 3
Décomposition spectrale et comparaison de $\xi G(a^{\text{test}}_{[:,h]})$ avec $\xi \widehat{G}^{(+h)}(a^{\text{test}}_{[:,0]})$

Neuf fonctions propres identifiées

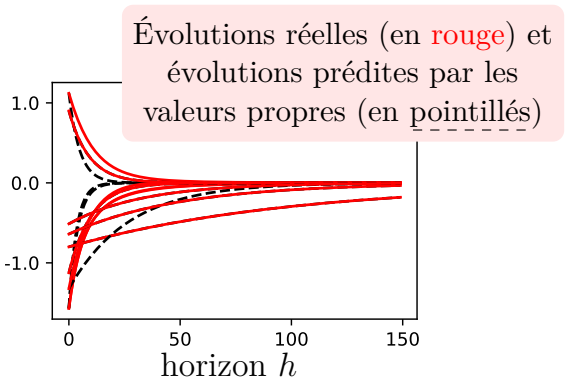


Figure 4.2 – *Approximation de l'opérateur de Koopman par approche orientée données. Les fonctions propres identifiées sont pour la plupart fallacieuses, synonyme que certaines observations ouvrent le système augmenté. Pour une étude détaillée sur ce système, le lecteur intéressé est encouragé à lire les articles de Kaiser et al. (2017) et Brunton et al. (2016a).*

latent.

Le cœur du problème

L'objectif est de trouver la *meilleure* approximation finie de l'opérateur de Koopman. On peut agir :

— Lors du choix des observables. Soit on considére des éléments de référence canoniques (série de Taylor, de Laurent, de Fourier, etc.) soit on cherche à identifier des fonctions propres. Les approches basées sur les réseaux de neurones ou SINDy implicite sont prometteuses pour déterminer des observables qui ont une dynamique linéaire (Brunton et Kutz, 2019).

— Lors de la régression pour déterminer K. On peut limiter la corruption de mauvais observables en promouvant la parcimonie dans la solution (*e.g.* LASSO ou LARS [14]).

Ces deux approches ne s'excluent pas l'une de l'autre. Dans les sous-sections qui suivent, on présente les différentes stratégies qui seront implémentées. Pour faciliter la lecture, on résume dans le tableau 4.1 les éléments clés qui distinguent chaque sous-section.

Section	Description	Innovation
DMD - 4.3.1	Les observations sont les composantes de l'état	Utilisation de modèles directs
HODMD - 4.3.2	Les observations sont les composantes de l'état retardé	Optimisation du retard Erreur de validation
EDMD - 4.3.3	Les observations sont les monômes au plus quadratiques	Erreur de validation
EDMD RBF - 4.3.3	Les observations sont des fonctions radiales	Optimisation des partitions Erreur de validation
EDMD DL - 4.3.4	Les observations sont les sorties d'un réseau de neurones	Procédure ASHA

Tableau 4.1 – *Description succincte des approximations de l'opérateur de Koopman qui sont développées dans les sections qui suivent.*

4.3.1 | Décomposition modale dynamique

L'algorithme de décomposition modale dynamique (DMD) a été introduit en mécanique des fluides par (Schmid, 2011). Les observations considérées sont directement les composantes de l'état. L'approximation finie de l'opérateur est donc **le meilleur modèle dynamique linéaire** pour décrire l'évolution des données. Il s'obtient en résolvant :

14. *Least Angle RegreSsion*. La tendance linéaire est apprise en étudiant les corrélations entre chaque prédicteur (ici les observables à t) et les sorties (ici les observables à $t+1$).

$$K = \arg\min_{K} \left|\left| \begin{bmatrix} a^{\text{train}}_{[1,1]} & \dots & a^{\text{train}}_{[1,m_{\text{train}}-1]} \\ \vdots & \vdots & \vdots \\ a^{\text{train}}_{[r,1]} & \dots & a^{\text{train}}_{[r,m_{\text{train}}-1]} \end{bmatrix} - K \begin{bmatrix} a^{\text{train}}_{[1,2]} & \dots & a^{\text{train}}_{[1,m_{\text{train}}]} \\ \vdots & \vdots & \vdots \\ a^{\text{train}}_{[r,2]} & \dots & a^{\text{train}}_{[r,m_{\text{train}}]} \end{bmatrix} \right|\right|^2_F$$

où $||.||_F$ est la norme de Frobenius. On fait le choix de ne pas ajouter de pénalité (comme dans la présentation classique de la méthode) donc la solution s'obtient avec une pseudo-inverse. La matrice des poids de Koopman est l'identité. Pour le lecteur habitué à la DMD dans le cadre de la mécanique des fluides, on renvoie à la note de bas de page [15].

Correction par modèles directs

Dans le cas où la dynamique de l'état latent n'est pas linéaire, le sous espace généré par les observables DMD n'est pas invariant. Les fonctions propres identifiées sont fallacieuses et les erreurs s'accumulent lors d'une prédiction récursive. Pour corriger la prédiction à un horizon donné, on peut utiliser des modèles *directs*.

On note $m^{(h)}$ l'opérateur qui permet de faire une prédiction récursive h pas de temps dans le futur à partir d'une condition initiale. Ainsi, on peut écrire :

$$m^{(h)}\left(a_{[:,t]}\right) = \hat{a}^{(P)}_{[:,t+h]} = K^h a_{[:,t]}$$

Le modèle direct m^h permet d'estimer le résidu de prédiction à l'horizon h, selon :

$$m^h\left(a_{[:,t]}\right) \approx a_{[:,t+h]} - m^{(h)}\left(a_{[:,t]}\right)$$

On peut donc mettre à jour la prédiction récursive à l'horizon h :

$$\hat{a}^{(P),\text{UP}}_{[:,t+h]} = m^{(h)}\left(a_{[:,t]}\right) + m^h\left(a_{[:,t]}\right)$$

où le sigle UP fait référence au mot *updated*. Le modèle direct m^h est établi par l'algorithme ci-dessous :

15. La décomposition modale dynamique introduite dans l'article de Schmid (2011) a surtout vocation à post traiter en mécanique des fluides. En partant du cliché initial de **haute dimension** $u_1 = u^{\text{train}}_{[:,1]}$, on construit le sous espace de *Krylov* Vect $\left(u_1, Ku_1, ..., K^{m-1}u_1\right)$ où $m = m_{\text{train}}$. Les itérations d'*Arnoldi* (Gram-Schmidt modifié) déterminent une base orthogonale dont les vecteurs sont approximativement les m vecteurs propres dominants de K. Ces vecteurs propres définissent les directions principales d'évolution du meilleur modèle dynamique linéaire *un pas de temps*. Aujourd'hui, on utilise plutôt l'algorithme DMD qui 1) réduit les clichés par décomposition orthogonale propre ; 2) effectue une régression linéaire ; 3) effectue la décomposition spectrale sur l'approximation de faible rang de la dynamique (Kutz *et al.*, 2016a). Dans cette thèse, on reprend l'algorithme DMD mais la réduction est déjà faite et elle peut être LAE, LVAE ou VAE.

Correction par modèle direct

1. **Choisir l'horizon** h
2. **Apprendre le modèle direct** m^h :
   (a) **Créer l'ensemble d'apprentissage**

   On dispose de $m_{\text{train}} - h$ données étiquetées. La donnée t est telle que :

$$\begin{cases} \textbf{Entrée} \rightarrow a^{\text{train}}_{[:,t]} \\ \textbf{Sortie} \rightarrow a^{\text{train}}_{[:,t+h]} - m^{(h)}\left(a^{\text{train}}_{[:,t+h]}\right) \end{cases}$$

   (b) **Faire une régression**

   On utilise les outils de l'apprentissage supervisé. L'objectif est d'expliquer la sortie par l'entrée. On rappelle les éléments clés : choix d'une forme de modèle pour m^h, validation croisée des hyper-paramètres, apprentissage sur tout le set de données et déploiement.

Remarque : pour l'application qui nous intéresse, on apprendra seize modèles directs, un pour chaque horizon de prédiction de $h = 1$ à $h = 16$. Les modèles seront des arbres de décision boostés. On comparera des modèles validés et non validés [16].

4.3.2 | Décomposition modale dynamique d'ordre élevé

La décomposition modale dynamique d'ordre élevé (HODMD) utilise l'état courant et les états passés comme observables (Le Clainche et Vega, 2017). Une décomposition d'ordre 0 est une DMD classique. Pour une décomposition d'ordre $d > 1$, le problème à résoudre est

$$K = \arg\min_K ||\tilde{a}^{[\text{train},(+1)]} - K\tilde{a}^{\text{train}}||_F^2 + \alpha||K||_{21}$$

Les matrices $\tilde{a}^{[\text{train},(+1)]}$ et $\tilde{a}^{\text{train}}$ possèdent $[(d+1) \times r]$ lignes et $[m_{\text{train}} - d - 1]$ colonnes. La notation *tilda* est reprise de l'article original et traduit :

16. Si l'on doit apprendre plusieurs modèles directs mono-tâche, une longue validation croisée peut ne pas valoir le gain en robustesse.

$$\tilde{a}_{[:,t]} = \begin{bmatrix} a_{[:,t-d]} \\ a_{[:,t-(d-1)]} \\ \vdots \\ a_{[:,t-1]} \\ a_{[:,t]} \end{bmatrix} \quad \text{et} \quad \tilde{a}^{(+1)}_{[:,t]} = \begin{bmatrix} a_{[:,t-(d-1)]} \\ a_{[:,t-(d-2)]} \\ \vdots \\ a_{[:,t]} \\ a_{[:,t+1]} \end{bmatrix}$$

Étant donné la structure de ces vecteurs, l'approximation finie de l'opérateur de Koopman *devrait* (s'il n'y a pas de problèmes numériques) prendre la forme suivante :

$$K = \begin{bmatrix} 0_{r\times r} & I_{r\times r} & 0_{r\times r} & \dots & 0_{r\times r} \\ 0_{r\times r} & 0_{r\times r} & I_{r\times r} & \dots & 0_{r\times r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{r\times r} & 0_{r\times r} & 0_{r\times r} & \dots & I_{r\times r} \\ K_1 & K_2 & K_3 & \dots & K_{d+1} \end{bmatrix}$$

où les sous matrices K_j sont des blocs $[r \times r]$ qui permettent d'expliquer l'état futur $a_{[:,t+1]}$ par les composantes des états $a_{[:,t-(d-1)]}$ à $a_{[:,t]}$. Le terme de régularisation $||K||_{21}$ contrôle la parcimonie [17]. Il reste deux points à aborder : 1) comment choisir le nombre de retards d et 2) comment valider l'hyper-paramètre α ?

Nombre de retards

Dans le chapitre sur les méthodes de reconstruction, les hyper-paramètres contrôlaient la structure du modèle **sans changer le nombre d'entrées ni de sorties**. Les différentes architectures pouvaient donc être comparées, via le score de validation et notamment le coefficient de détermination. Pour le modèle HODMD, le nombre de retards fixe le nombre de variables explicatives et donc les modèles ne sont pas directement comparables [18]. Dans la théorie de l'information, le critère d'information *d'Akaike* permet de pénaliser les modèles en fonction du nombre de paramètres (donc implicitement de retards) utilisés. Il se calcule par :

$$\text{AIC} = 2T - 2\ln(L)$$

où T est le nombre total de paramètres [19] et L est le maximum de vraisemblance du modèle [20]. Le critère d'information *Bayésien* est une variante qui pénalise aussi le modèle par la taille de l'échantillon. Il se calcule par :

$$\text{BIC} = T\ln(m_{\text{train}}) - 2\ln(L)$$

17. Pour une prédiction robuste, il vaut mieux ne pas considérer toutes les $[(d+1)\times r]$ variables explicatives.

18. C'est d'ailleurs l'un des plus grands défauts du coefficient de détermination : sa valeur augmente nécessairement avec le nombre de prédicteurs. Il existe une version ajustée qui pénalise l'ajout de variables explicatives, mais on préfère utiliser des critères d'informations (plus classique).

19. Il faut déterminer $d+1$ sous blocs K_j de taille $r \times r$ donc $(d+1)\times r^2$ paramètres au total.

20. Bien formalisé dans une approche bayésienne de la régression linéaire, où chaque étiquette est échantillonné d'une loi normale. Voir Bishop et Tipping (2003).

où $\ln(m_{\text{train}})$ remplace le 2 du critère AIC. Le nombre optimal de retards est celui qui maximise un critère d'information.

Régularisation LASSO

Pour gagner en robustesse, on sélectionne les variables explicatives en utilisant LASSO. L'hyper-paramètre α est choisi par validation croisée de type *hold-out*[21] avec une erreur de validation modifiée :

$$E(\alpha) = \frac{1}{m_{\text{val}}} \sum_{t=1}^{m_{\text{val}}} ||\tilde{a}^{\text{val}}_{[:,t+1]} - K(\alpha)\tilde{a}^{\text{val}}_{[:,t]}||_2^2 + \frac{1}{m_{\text{val}}} \sum_{h=1}^{m_{\text{val}}} ||\tilde{a}^{\text{val}}_{[:,h+1]} - [K(\alpha)]^h \tilde{a}^{\text{val}}_{[:,1]}||_2^2$$

Le premier terme est l'erreur quadratique moyenne classique. Le second évalue l'erreur de prédiction récursive moyenne de toute la trajectoire de validation. On indique explicitement $K(\alpha)$ car l'approximation de l'opérateur de Koopman dépend naturellement du niveau de parcimonie imposé.

Quelques remarques

L'algorithme HODMD est une extension assez naturelle de l'algorithme DMD. Quelques remarques de contexte doivent être faites.

Remarque 1 : pour de nombreux systèmes dynamiques, la complexité *spectrale*[22] est plus grande que la complexité *spatiale*[23]. Une décomposition modale classique n'est pas consistante pour ces systèmes car le spectre recouvrable est limité par la dimension de l'état. Avec une décomposition modale d'ordre plus élevée, on peut jouer sur le nombre de retards pour augmenter le spectre recouvrable et donc trouver autant de modes que la complexité spectrale. La HODMD peut donc être appliquée à des systèmes dynamiques très généraux, comme démontré et illustré dans l'article ayant introduit la méthode (Le Clainche et Vega, 2017).

Remarque 2 : la version purement statistique de l'algorithme HODMD est VAR (pour *Vector Auto Regression*). La bibliothèque *statsmodel* propose ce modèle avec d'autres outils : étude de la stationnarité (test de *Dickey-Fuller*), de la cointégration (test de *Johansen*) et de la causalité (test de *Granger*). Ces études détaillées sont surtout effectuées dans des problèmes économétriques et sortent du cadre de ce manuscrit[24] (Toda et Phillips, 1994).

21. Celle que l'on avait appelée *naïve* dans le chapitre précédent. Comme on travaille avec des séries temporelles, ça n'aurait pas de sens de faire une validation par trois blocs et de placer un ensemble de test *avant* un ensemble d'entraînement.

22. Nombre de modes dynamiques requis pour écrire l'état comme une superposition de ces modes. Pour le système de Lorenz chaotique, la complexité spectrale est infinie.

23. Dimension du sous espace généré par les modes dynamiques. Pour le système de Lorenz chaotique, la complexité spatiale est 3.

24. On a quand même vérifié que toutes les séries temporelles étudiées dans les chapitres 5 et 6 étaient stationnaires.

Remarque 3 : les critères d'informations sont bien détaillées à la page 151 de Brunton et Kutz (2019). En maximisant un critère, on détermine le nombre de retards optimal pour expliquer *toutes* les composantes de l'état. Si on cherche un modèle auto-régressif pour *une seule* composante, on étudie plutôt l'auto-corrélation et l'auto-corrélation partielle. L'auto-corrélation est une fonction de d, définie pour la composante i_r par $\mathbb{E}_t\left[a_{[i_r,t]} \times a_{[i_r,t+d]}\right]$ et estimée sur les données d'entraînement. Si l'auto-corrélation est significative [25] jusqu'à un retard $d > 1$, il existe un terme auto-régressif. Le nombre de retards est diagnostiqué avec l'auto-corrélation partielle qui calcule l'auto-corrélation du signal en ayant retiré les dépendances linéaires des variables $a_{[i_r,t+1]}$ à $a_{[i_r,t+d-1]}$.

Remarque 4 : pour l'algorithme HODMD, la matrice des poids de Koopman est une matrice rectangulaire $r \times [(d+1) \times r]$ nulle sauf pour les r dernières colonnes qui forment l'identité.

4.3.3 | Décomposition modale dynamique étendue

La décomposition modale dynamique étendue a été introduite en 2015 (Williams *et al.*, 2015). C'est le cadre le plus général pour approximer l'opérateur de Koopman, les observables étant des fonctions non linéaires **choisies par l'utilisateur**. Le problème à résoudre s'écrit génériquement :

$$K = \arg\min_K ||G^{[\text{train},(+1)]} - KG^{\text{train}}||_F^2 + \alpha||K||_{21}$$

La validation croisée du paramètre α utilise la même erreur que la décomposition d'ordre élevé. Ici, on s'intéresse à deux choix de fonctions candidats : des monômes au plus quadratique ou des fonctions de base radiales.

Observations quadratiques (EDMD)

Les observables sont les monômes au plus quadratiques que l'on peut construire avec les composantes de l'état. Le nombre de combinaisons possibles [26] est :

$$\binom{r+2}{2} - 1 = \frac{(r+2)!}{2r!} - 1 = \frac{(r+1)(r+2)}{2} - 1$$

Le vecteur des observations à l'instant t s'écrit typiquement :

25. On effectue le *test Z*. Les limites de significativité haute et basse pour un risque α sont données par $B = \pm z_{1-\alpha/2} SE(r_d)$ où r_d est l'auto-corrélation estimée au retard d. Si r_d est en dehors des limites, la corrélation est statistiquement significative i.e. on peut rejeter l'hypothèse nulle "la corrélation de la population au retard d est nulle". Le terme $z_{1-\alpha/2}$ est le quantile d'ordre $1-\alpha/2$ de la distribution normale centrée réduite et SE est l'erreur standard, calculée par la formule de Bartlett.

26. La théorie s'appuie sur les polynômes symétriques homogènes complets. Si on a r variables et que l'on veut les degrés 1 à n, le nombre de monômes possibles est n parmi $r+n$ auquel on retranche 1 (pour ne pas avoir le monôme 1).

$$G_{[:,t]} = \begin{bmatrix} a_{[1,t]} \\ a_{[1,t]}a_{[2,t]} \\ a_{[1,t]}a_{[3,t]} \\ \vdots \\ a_{[r,t]}a_{[r-1,t]} \\ a^2_{[r,t]} \end{bmatrix} \qquad \text{et} \qquad G^{(+1)}_{[:,t]} = G_{[:,t+1]}$$

où il faut faire attention à ne pas considérer de monômes doublons.

Observations plaque minces (EDMD RBF)

On commence par partitionner les états d'entraînement a^{train} avec l'algorithme des k-moyennes. Les l centroïdes déterminés sont utilisés pour construire les fonctions d'observations :

$$\boxed{\forall i_l \in [1..l],\ G_{[i_l,t]} = ||a_{[:,t]} - \mu_{i_l}||_2^2 \log\left[a_{[:,t]} - \mu_{i_l}\right]}$$

où $\mu_{i_l} \in \mathbb{R}^r$ est le centroïde de la partition l et $||.||_2$ est la norme l_2. Le nombre de partitions est choisi dans le coude de la courbe d'inertie en fonction du nombre de partitions [27]. On vérifie *a posteriori* que les partitions sont homogènes en évaluant le coefficient de silhouette (Letí *et al.*, 2004) pour chaque échantillon. Il est défini par :

$$s\left(a^{\text{train}}_{[:,t]}\right) = \frac{d_{in}(a^{\text{train}}_{[:,t]}) - d_{out}(a^{\text{train}}_{[:,t]})}{\max(d_{in}(a^{\text{train}}_{[:,t]}), d_{out}(a^{\text{train}}_{[:,t]}))}$$

où d_{in} est la distance moyenne du point à son groupe et d_{out} est la distance moyenne au groupe voisin le plus proche [28]. Un bon partitionnement [29] se traduit par un coefficient de silhouette moyen (sur tous les échantillons) proche de $+1$.

Remarque 1 : l'acronyme RBF signifie *Radial Basis Function*. Les fonctions d'observations sont en fait des fonctions de base qui peuvent être utilisées pour faire de l'interpolation dite *de plaque mince*. C'est pourquoi certains articles parlent d'observations TPS pour *thin plate splines*.

Remarque 2 : pour les observations quadratiques, la matrice des poids de Koopman est triviale. Pour les observations radiales, une pseudo-inverse pour estimer le retour vers l'espace d'état n'est sans doute pas suffisant.

27. Critère heuristique : au delà du coude, ajouter des partitions ne vaut plus le gain en inertie.
28. Les formules complètes n'ont pas grand intérêt et ne sont pas indiquées.
29. Sur un axe vertical, on définit l intervalles. Dans l'intervalle i_l, on trace autant de barres horizontales que d'échantillons dans la partition i_l. La taille d'une barre est fixée par le coefficient de silhouette de l'échantillon. À la fin, on dispose de l silhouettes. Un partitionnement est bon si les silhouettes ont une épaisseur semblable et si chaque silhouette est homogène (barres de longueur semblables).

4.3.4 | Apprentissage du dictionnaire d'observables

Toutes les méthodes précédentes sont basées sur des observations fixées par l'utilisateur (avec une sélection LASSO possible). Dans cette section, les observations sont les sorties **à optimiser** d'un réseau de neurones. Le problème d'optimisation général s'écrit (Li *et al.*, 2017) :

$$(K, W) = \underset{(K,W)}{\arg\min} ||G^{[\text{train},(+1)]}(W) - KG^{\text{train}}(W)||_F^2 + \alpha||K||_F^2$$

où W désigne les paramètres (poids et biais) du réseau de neurones, $G(W)$ est la couche de sortie calculée après propagation de $a_{[:,t]}$ et K est l'approximation finie de l'opérateur de Koopman. La variance de K est pénalisée grâce au terme $||K||_F^2$. Pour être plus concis, on désigne le coût par $J(W, K)$. Deux questions se posent : 1) comment résout-on le problème pour des hyper-paramètres fixés et 2) comment optimiser les hyper-paramètres du réseau ?

Hyper-paramètres fixés

On suppose que l'architecture du réseau est fixée. L'objectif est d'apprendre les paramètres W tels que les observables appris fournissent une bonne approximation K de l'opérateur de Koopman. Pour résoudre le problème d'optimisation, on utilise une méthode *alternée* résumée dans l'encadrée ci-dessous.

Apprentissage des observables

Requis : les hyper-paramètres du réseau sont fixés. Le réseau possède r neurones d'entrées et l neurones de sortie, un neurone étant une fonction d'observation.

Tant que le coût J n'est pas convergé (itération k) :

— **Créer les observables $G = [G_{\text{fixe}}|G_{\text{NN}}(W^{(k)})]$**

La partie gauche contient deux observables fixes [a], que l'on choisit comme étant des fonctions radiales. La partie droite contient les observables du réseau de neurones à l'itération k.

— **Fixer $W^{(k)}$, apprendre K**

C'est la régression linéaire classique :

$$K^* = \underset{K}{\arg\min}\, J(W^{(k)}, K)$$

La pénalité α est optimisée par validation croisée de type *hold-out*.

— **Fixer K, apprendre $W^{(k+1)}$**

C'est l'apprentissage classique d'un réseau de neurones. Il s'agit d'un processus itératif :

1. **Initialisation des paramètres pour le dictionnaire $k+1$:**

$$W^{[(k+1),(0)]} = W^{(k)}$$

2. **Descente de gradient sur plusieurs epochs (itération e) :**

$$W^{[(k+1),(e+1)]} \leftarrow W^{[(k+1),(e)]} - \eta \nabla \bar{J}_{[K^*]} \left(W^{[(k+1),(e)]} \right)$$

Cette notation est *synthétique*. Comme pour l'apprentissage des auto-encodeurs, la descente s'effectue par batchs de données. Lorsque le coût évalué sur les batchs de validation (non utilisés pour la descente) évolue peu ou raugmente, la descente est arrêtée.

3. **Conclure au dernier epoch d'entraînement, noté E :**

$$W^{(k+1)} = W^{[(k+1),(E)]}$$

La fonction coût $\bar{J}$ est une version modifiée de J composée de trois termes. Pour le batch b composé de m_b données et pour les paramètres W, elle s'écrit :

$$\begin{aligned} \bar{J}_{[K^*]}(W) = &\sum_{t=1}^{m_b} ||G^b_{[:,t+1]} - K^* G^b_{[:,t]}||_2^2 \\ &+ \sum_{h=1}^{m_b} ||G^b_{[:,h+1]} - [K^*]^h G^b_{[:,1]}||_2^2 \\ &+ \sum_{t=1}^{m_b} ||a^b_{[:,t]} - a^b \left[G^b\right]^\dagger G^b_{[:,t]}||_2^2 \end{aligned}$$

Les paramètres W définissent les observations $G^b \in \mathbb{R}^{l \times m_b}$, obtenues par propagation avant du batch $a^b \in \mathbb{R}^{r \times m_b}$. Le premier terme est l'erreur de prédiction *un pas de temps*, déjà présent dans le coût J. Le second terme est l'erreur de prédiction *récursive* sur tout le batch. Le troisième terme impose d'apprendre des observables *facilement* inversibles par moindres carrés.

À la fin des itérations, on dispose d'un réseau de neurones optimisé pour construire des observables qui évoluent linéairement. On peut alors approximer l'opérateur de Koopman.

a. Ils sont ajoutés pour ne pas apprendre la solution triviale $J(\theta, K) = 0$ obtenue avec des observations nulles.

La procédure d'apprentissage est plutôt longue car une itération est constituée d'une régression forte pente (avec validation croisée de α) et l'apprentissage d'un réseau de neurones. Si on veut optimiser les hyper-paramètres du réseau, une recherche par grille n'est donc pas envisageable.

Optimisation des hyper-paramètres

Pour optimiser les hyper-paramètres du réseau de neurones, on utilise la version asynchrone d'un algorithme dit SHA pour *Sucessive Halving Algorithm* (Jamieson et Talwalkar, 2016). Il s'agit d'une recherche aléatoire par grille avec arrêt prématuré [30]. Supposons que l'on veuille tester 64 combinaisons aléatoires d'hyper-paramètres. Plutôt que de dérouler *complètement* l'encadré bleu sur toutes les combinaisons et choisir celle au coût de validation J minimal, on teste les combinaisons sur peu d'itérations et on élimine les moins prometteuses. Avec un taux d'élimination de 4, on se retrouve avec le schéma du tableau 4.2. Ainsi, les 64 combinaisons d'hyper-paramètres sont testées sur *une seule* itération (encadré bleu avec $k = 1$). Le quart des combinaisons les plus prometteuses est promu au palier suivant et entraîné sur trois itérations supplémentaires (encadré bleu jusque $k = 4$). Le quart des meilleures combinaisons est encore promu et ainsi de suite.

Palier	Nombre de combinaisons	Nombre itérations
0	64	1
1	16	4
2	4	16
3	1	64

Tableau 4.2 – *Brassage des hyper-paramètres testés dans une approche de halving synchrone. Ici, on part de 64 combinaisons aléatoires d'hyper-paramètres et on promeut le quart des combinaisons les plus prometteuses sur la première itération (i.e. un apprentissage de K et une optimisation des paramètres du réseau d'observables). On continu jusqu'à n'avoir qu'une seule combinaison d'hyper-paramètres.*

Dans cette version *synchrone* de l'algorithme, toutes les combinaisons d'un palier (aussi appelé *rung*) sont entraînées avant de faire les promotions et de passer au

30. Même principe que le *early stopping* dans l'apprentissage d'un réseau de neurones : on arrête l'entraînement lorsque le coût de validation raugmente ou évolue peu sur plusieurs epochs. Pour l'optimisation des hyper-paramètres, on sélectionnera donc aléatoirement des candidats puis on arrête prématurément les combinaisons qui augmentent l'erreur de validation.

palier suivant. Dans la version *asynchrone* que l'on utilise, les configurations sont promues dès que possible. Cela permet une optimisation massive (et parallèle) des hyper-paramètres. L'algorithme complet nécessitant un bon niveau d'abstraction, il n'est pas indiqué ici. Le lecteur intéressé peut se référer à l'article original (Li *et al.*, 2018a) et tester l'algorithme avec la bibliothèque python *sherpa*.

Quelques remarques

L'algorithme EDMD DL est assez complexe à implémenter. On donne ici quelques remarques de contexte.

Remarque 1 : la méthode EDMD DL est inspirée de l'article de Li *et al.* (2017). Toutefois, les hyper-paramètres n'étaient pas optimisés et les applications étaient purement académiques.

Remarque 2 : on choisit une régularisation de Tikhonov plutôt qu'une régularisation LASSO pour accélérer le calcul. Une pénalité l_2 pose moins de problèmes d'un point de vue calcul car la norme est différentiable partout.

Remarque 3 : les résultats seront peu probants sur nos écoulements. Plusieurs pistes sont à envisager. 1) il faudrait améliorer l'exploration des hyper-paramètres et ne pas faire un brassage aussi brutal dès la première itération [31] ; 2) il faudrait avoir un œil de mathématicien (pas uniquement appliqué) sur l'algorithme d'optimisation alternée ; 3) il faudrait complexifier la définition de l'architecture du réseau et ne pas se contenter des même activations et même nombre de neurones dans chaque couche.

4.4 | Quantification de la qualité des prédictions

Dans la section précédente, on a présenté différents choix d'observables pour approximer l'opérateur de Koopman du système. Chaque approximation fournit un modèle dynamique pour prédire récursivement l'état du système ou ses observations. Pour des systèmes complexes, un mauvais choix d'observables mène vraisemblablement à des fonctions propres fallacieuses et donc un pouvoir prédictif limité. Dans cette section, on présente les métriques utilisées pour quantifier cette erreur.

Stratégie de découpage

Le modèle prédictif étant vraisemblablement erroné, il serait décourageant de comparer une seule *longue* trajectoire avec sa prédiction à partir de la condition initiale : on aurait une erreur de 100%. L'idée est plutôt de découper la trajectoire d'entraînement ou de test en plusieurs petites trajectoires d'horizon raisonnable. Si

31. L'algorithme complet définit d'ailleurs des paramètres. On peut choisir le nombre minimal d'itérations avant promotion et attendre d'avoir plusieurs configurations dans le dernier palier.

on note H l'horizon maximal [32] et m le nombre de données disponibles, on peut définir $B = m - H - 1$ sous trajectoires ou *batchs* de taille $H + 1$. En notant G la matrice des clichés observés (d'entraînement ou de test), la b-ième sous trajectoire (avec $b = 1$ à $b = B$) est donnée par $G_{[:,b]} \to G_{[:,b+H]}$.

Erreur à un horizon fixé

L'erreur quadratique moyenne pour la prédiction récursive à $h \in [1..H]$ fixé est calculée suivant :

$$\forall h \in [1..H],\ \mathcal{E}_{\text{obs}}(h) = \sum_{b=1}^{B} ||G_{[:,b+h]} - K^h G_{[:,b]}||_F^2$$

L'erreur dans l'espace d'état s'écrit de la même manière :

$$\forall h \in [1..H],\ \mathcal{E}_{\text{état}}(h) = \sum_{b=1}^{B} ||a_{[:,b+h]} - B_{[\text{MC}]} K^h G_{[:,b]}||_F^2$$

Où a est la matrice de tous les clichés. Dans la troisième partie du manuscrit, on présentera cette erreur sous sa forme normalisée, comme dans le chapitre 2. On parlera *d'erreur de prédiction par récursion.*

Erreur de prédiction moyenne

L'erreur pour toutes les sous trajectoires et tout l'horizon est obtenue en moyennant sur l'horizon maximal H. Dans l'espace d'état ou l'espace des observables, on écrit donc :

$$\mathcal{E} = \frac{1}{H} \sum_{h=1}^{H} \mathcal{E}(h)$$

On parlera *d'erreur de prédiction.* Encore une fois, on présentera plutôt sous la forme normalisée, en divisant par la variance totale des données.

Qualité locale d'une fonction propre

Parmi les fonctions propres identifiées à partir des vecteurs propres gauches de l'opérateur de Koopman, il y en a qui sont plus fallacieuses que d'autres. Si on

32. Pour nos écoulements, on considérera $H = 16\Delta t^*$ où le pas de temps sans dimension vaut $\Delta t^* = 1$. Ce temps est caractérisque de l'instationnarité macroscopique de l'écoulement et sert (souvent) à définir l'espacement des clichés (Kaiser *et al.*, 2014). Le choix de seize pas de temps est arbitraire mais pourrait être optimisé, en utilisant par exemple l'exposant de Hurst (voir l'article de Mohan et Gaitonde (2018) qui utilise des réseaux de neurones récurrents pour prédire des écoulements turbulents). Dans une approche plus rigoureuse, on pourrait calculer l'échelle de temps intégrale ou l'inverse de l'exposant maximal de Lyapunov (Mohan *et al.*, 2017). Toutefois, peu d'articles fonctionnent dans ce sens et la plupart font une étude paramétrique de l'influence de l'horizon (Wang *et al.*, 2020).

note $\Phi \in \mathbb{R}^{l\times b}$ la trajectoire des fonctions propres et $\Lambda \in \mathbb{R}^{l\times l}$ les valeurs propres associées, la qualité $h \in [1..H]$ pas de temps dans le futur de la fonction $j \in [1..l]$ est estimée par :

$$\forall h \in [1..H], \forall j \in [1..l],\ R_j^2(h) = 1.0 - \frac{\sum_{b=1}^{B}\left(\Phi_{[j,b+h]} - \lambda_j^h \Phi_{[j,b]}\right)^2}{\sum_{b=1}^{B}\left(\Phi_{[j,b+h]} - \frac{1}{B}\sum_{b=1}^{B}\Phi_{[j,b+h]}\right)^2}$$

Ce qui représente l'écart à l'unité de la variance résiduelle par rapport à la variance totale.

Remarque 1 : dans la dernière partie du manuscrit, on présentera deux types de graphiques. Le premier superpose l'erreur de prédiction par récursion (*barres bleues*) et les qualités locales de chaque fonction propre (*courbes vertes*). Si les barres ne sont pas nulles et que les courbes ne sont pas égales à l'unité, c'est que les observables choisis ne forment pas un sous espace invariant par l'opérateur de Koopman.

Remarque 2 : l'erreur $\mathcal{E}(1)$ est l'erreur DMD que l'on rencontre dans beaucoup d'articles. Elle renseigne sur la qualité de prédiction *un pas de temps* en supposant à chaque fois la bonne condition initiale.

Remarque 3 : avec le score R_j^2, on pourrait potentiellement identifier un sous ensemble de bonnes fonctions propres pour construire l'approximation de l'opérateur de Koopman (Kaiser *et al.*, 2018).

4.5 | Prédiction long terme par assimilation de données

Attention

Cette section est à titre informatif. L'algorithme ne sera pas appliqué dans la troisième partie de ce manuscrit, sauf dans une version simplifiée présentée en conférence et répertoriée en annexe H.

Pour une introduction didactique à l'assimilation de données, on renvoie au chapitre 21 de Kutz (2013).

Dans le chapitre sur les méthodes de reconstruction, nous avions présenté trois méthodes d'estimation de l'état du système. Seules deux (estimations directe et régressive) avaient été détaillées. La méthode *d'assimilation de données* requérait

un modèle dynamique et n'avait donc pas été discutée. Grâce au travail des sections précédentes, on est maintenant en mesure de prédire l'état et donc d'envisager l'assimilation de données. L'idée est de mélanger deux types d'estimation :

→ **Par reconstruction**. On se place dans le cas d'une reconstruction régressive. Disposant de la mesure $y_{[:,t]}$ et d'un modèle de reconstruction $\hat{f}_r$, on peut écrire :

$$\hat{a}^{(R)}_{[:,t]} = \hat{f}_r\left(y_{[:,t]}\right)$$

Cette estimation est bien sûre erronée. De plus, les mesures qui servent à la reconstruction peuvent être bruitées.

→ **Par prédiction**. Disposant d'une condition initiale $a_{[:,t-1]}$, d'un choix d'observables G et d'une approximation finie de l'opérateur de Koopman $(B_{[\mathrm{MC}]}, K)$, on peut écrire :

$$\hat{a}^{(P)}_{[:,t]} = B_{[\mathrm{MC}]} K G\left(a_{[:,t]}\right)$$

Cette estimation est aussi erronée. De plus, la condition initiale peut être bruitée.

Pour une estimation continue de l'état, faire uniquement de la reconstruction n'est pas envisageable car les mesures peuvent ne pas être disponibles à tout instant. Faire uniquement de la prédiction récursive est aussi inenvisageable car les erreurs finiraient par trop s'accumuler. L'assimilation de données utilise **les reconstructions ponctuelles comme garde fou pour mettre à jour les conditions initiales au cours de l'estimation**. On reprend donc les idées du filtre de Kalman mais les modèles mis en jeu sont purement orientés donnés.

4.5.1 | Les sources d'erreur

Supposons que l'on assimile les données $y_{[:,t]}$ toutes les F prédictions. Pour mettre en place le filtre de Kalman, on a besoin de connaître les informations suivantes :

1. Matrice de covariance Q des erreurs de prédiction sur un horizon F dans l'espace des observables. Cette matrice est telle que :

$$G\left(a_{[:,t+F]}\right) \sim \mathcal{N}\left(K^F G\left[a_{[:,t]}\right], Q\right)$$

On estime la matrice Q sur les données d'apprentissage, en calculant l'erreur quadratique moyenne empirique $\mathcal{E}_{\mathrm{obs}}(F)$.

2. Matrice de covariance R des erreurs de mesure. Les capteurs sont bruités indépendamment avec :

$$\forall i_p \in [1..p],\ y_{[i_p,t]} \sim \mathcal{N}\left(\bar{y}_{[i_p,t]}, R_{[i_p,i_p]} = \left[I_y \max[y^{\mathrm{train}}_{[i_p,:]}]\right]^2\right)$$

Chaque mesure à t est donc bruitée d'un niveau I_y (entre 0 et 1) de la contribution maximale possible, ici celle observée sur le set d'apprentissage. **Attention** ! L'opération $\bar{.}$ est une moyenne d'ensemble obtenue à partir de plusieurs réalisations à l'instant t.

3. Matrice de covariance P_0 des erreurs de condition initiale. Le tout premier état à partir duquel l'estimation longue durée débute est bruité selon :

$$\forall i_r \in [1..r],\ a_{[i_r,t]} \sim \mathcal{N}\left(\bar{a}_{[:,t]}, P_{0,[i_r,i_r]} = \left[I_a \max[a^{\text{train}}_{[i_r,:]}]\right]^2\right)$$

Même remarque sur l'opération $\bar{.}$ qui indique la moyenne d'ensemble de l'état latent au temps t, calculée à partir de plusieurs réalisations à cet instant.

4.5.2 | L'algorithme

On indique ci-dessous l'algorithme d'assimilation de données. La démonstration des équations est très largement accessible dans la littérature (Mons *et al.*, 2016).

Assimilation de données

Requis : horizon de prédiction F, covariance de prédiction Q, covariance de mesure R, covariance de condition initiale sur l'état P^s_t, taille des ensembles S, condition initiale $\bar{a}_t$, opérateur de Koopman K et poids de Koopman B, mesures espacées de F pas de temps, modèle de reconstruction $\hat{f}_r$.

1. **Créer un ensemble d'états initiaux**.

   On tire S échantillons de $\mathcal{N}(\bar{a}_t, P^s_t)$.

2. **Observer l'ensemble des états initiaux**.

   L'ensemble des observations a pour moyenne empirique $\bar{g}_t$ et pour matrice de covariance empirique P^o_t.

3. **Propager les observations et leur covariance.**

   Comme la dynamique des observations est linéaire, on peut utiliser les équations du filtre de Kalman classique. On a donc :

$$\begin{array}{lll} \bar{g}_t & \text{donne} & \bar{g}^{(P)}_{t+F} = K^F \bar{g}_t \\ P^o_t & \text{donne} & P^{[\text{o-(P)}]}_{t+F} = K^T P^o_t [K^F]^T + Q \end{array}$$

   La première ligne décrit le transport de l'observation moyenne sur F pas de temps. La deuxième ligne décrit le transport de la covariance

des observables par le modèle linéaire.

4. **Créer un ensemble d'états transportées**.
   (a) Tirer S échantillons de $\mathcal{N}\left(\bar{g}_{t+F}^{(P)}, P_{t+F}^{[\text{o-(P)}]}\right)$.
   (b) Utiliser les poids de Koopman pour repasser dans l'espace d'état.
   (c) Calculer l'état moyen et la covariance d'état empiriques. On note respectivement $\bar{a}_{t+F}^{(P)}$ et $P_{t+F}^{[s-(P)]}$.
5. **Créer un ensemble de reconstructions à** $t+F$.
   (a) Tirer S mesures aléatoires de $\mathcal{N}\left(\bar{y}_{t+F}, R\right)$.
   (b) Mesurer l'ensemble via $\hat{f}_r$.
   (c) Calculer l'état reconstruit moyen et la covariance de reconstruction empiriques. On note respectivement $\bar{a}_{t+F}^{(R)}$ et $P_{t+F}^{[s-(R)]}$.
6. **Calculer l'incertitude puis le gain de Kalman**.

   Ces quantités sont notées respectivement S_{t+F} et Ga_{t+F} et se calculent selon :

$$\begin{aligned} S_{t+F} &= P_{t+F}^{[s-(P)]} + P_{t+F}^{[s-(R)]} \\ \text{Ga}_{t+F} &= P_{t+F}^{[s-(P)]} \left[S_{t+F}\right]^{-1} \end{aligned}$$

7. **Conclure par l'étape d'assimilation**.

   Connaissant l'état moyen prédit, l'état moyen reconstruit et le gain de Kalman, on peut mettre à jour et calculer :

$$\begin{aligned} \bar{a}_{t+F} &= \bar{a}_{t+F}^{(P)} + \text{Ga}_{t+F}\left[\bar{a}_{t+F}^{(R)} - \bar{a}_{t+F}^{(P)}\right] \\ P_{t+F} &= \left(I - \text{Ga}_{t+F}\right) P_{t+F}^{[s-(P)]} \end{aligned}$$

   Ces deux équations font écho au schéma prototype de l'état de l'art (figure 1.16). L'état analysé est vu comme un curseur se déplaçant entre l'état prédit et l'état reconstruit. Pour un gain *nul*, on fait aveuglément confiance à la prédiction tandis que pour un gain valant *l'identité*, on fait aveuglément confiance à la reconstruction.

8. **Refaire l'algorithme avec** $t \to t+F$.

Remarque 1 : la difficulté de l'algorithme est liée au fait que le modèle

dynamique est appris dans l'espace des observables. Avec l'algorithme DMD, on s'affranchit de cette difficulté. C'est ce qui est présenté en annexe H.

Remarque 2 : les étapes 2 et 5 sont critiquables. Une distribution gaussienne reste gaussienne lorsque l'on applique des transformations affines. Or dans les étapes 2 et 5, on utilise des fonctions observations et un modèle de reconstruction potentiellement non linéaires.

Remarque 3 : on peut envisager de l'assimilation régressive, comme présentée en annexe G sur l'attracteur de Lorenz.

Remarque 4 : pour être consistant avec les notations de reconstruction et de prédiction, on pourrait noter $\hat{a}^{(A)}_{[:,t]}$ l'état moyen analysé plutôt que $\bar{a}_t$.

4.6 | Synthèse du chapitre

Dans ce chapitre méthodologique, on a présenté l'opérateur de Koopman et différentes stratégies pour l'approximer. L'idée est de trouver des observations de l'état qui évoluent linéairement plutôt que d'apprendre un modèle dynamique non linéaire *boîte noire* de l'état. Tout l'enjeu consiste à choisir les observations pour 1) faire une prédiction linéaire dans l'espace des observables et 2) repasser dans l'espace d'état.

La méthode la plus simple est la décomposition modale dynamique où les observations sont directement les composantes de l'état latent. L'approximation de l'opérateur de Koopman est donc le meilleur modèle dynamique linéaire *un pas de temps* pour les données. Si la dynamique réelle est non linéaire, le modèle DMD est naturellement erroné. D'autres stratégies sont donc envisagées :

- La décomposition modale dynamique **d'ordre élevé**. On rajoute l'histoire de l'état latent. Plus le nombre de retards est important, plus le spectre recouvrable est grand. On peut potentiellement prédire n'importe quel système dynamique.
- La décomposition modale dynamique **étendue**. Les observations sont des fonctions non linéaires de l'état comme des monômes au plus quadratiques ou des fonctions radiales.
- La décomposition modale dynamique avec apprentissage des observables. Les observations sont les sorties d'un réseau de neurones qu'il faut optimiser.

Une fois les observations décidées, il suffit de faire une régression linéaire pour approximer l'opérateur de Koopman. Si les observations ne forment pas un sous espace invariant, les fonctions propres identifiées seront fallacieuses. Il est essentiel de régulariser la régression et de faire une validation croisée pour apprendre un modèle robuste sur de nouvelles trajectoires. Pour une prédiction dans l'espace d'état, il faut inverser les observables, une étape faite *naïvement* par moindres carrés mais qui peut ne pas être suffisante pour des observations complexes. Toujours est-il qu'avec cette procédure, on peut rapidement disposer d'un modèle

dynamique pour prédire le futur de l'état. En adjoignant un modèle de reconstruction, on peut même faire de l'assimilation de données.

Finalement, à l'issu des trois chapitres méthodologiques de ce chapitre, on sait 1) réduire une base de données ; 2) estimer l'état latent du système à partir de mesures ponctuelles ; 3) prédire le futur de l'état latent. On s'intéresse maintenant à l'application de ces outils pour estimer un système de haute dimension : un écoulement. Pour tester la scalabilité des outils, on considère quatre écoulements de complexité croissante dont les données brutes sont présentées dans le chapitre 5.

Troisième partie

Estimation orientée données de quatre écoulements de complexité croissante

5. Présentation des quatre écoulements retenus

Dans ce chapitre, on présente les différentes bases de données sur lesquelles on effectuera l'apprentissage des auto-encodeurs, des méthodes de reconstruction et des approximations de l'opérateur de Koopman. On considère quatre écoulements de complexité croissante : 1) un écoulement laminaire autour d'un cylindre 2D ; 2) un écoulement de couche de mélange spatiale ; 3) un écoulement turbulent autour d'un cylindre 3D ; 4) un écoulement autour d'une tour plongée dans une couche limite atmosphérique. L'objectif étant de présenter les données en entrée des algorithmes d'apprentissage automatique, on ne se concentre pas sur les méthodes numériques.

Attention

Les simulations sont effectuées sous *OpenFOAM*, un code volume finis *open source* pour résoudre des équations aux dérivées partielles. Des détails numériques sont fournis en annexe B.

Sommaire

5.1 | Principe de résolution des équations

En écoulement incompressible, les équations à résoudre sont la conservation de la masse et la conservation de la quantité de mouvement. Ces équations, dites de Navier-Stokes, s'écrivent pour un fluide newtonien :

$$\begin{cases} \boldsymbol{\nabla}.\boldsymbol{U} = 0 \\ \dfrac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{\nabla}.(\rho \boldsymbol{U}\boldsymbol{U}) = -\boldsymbol{\nabla} p + \mu \boldsymbol{\nabla}^2 \boldsymbol{U} + \rho \boldsymbol{g} \end{cases}$$

où $\boldsymbol{U}$ est le champ de vitesse, p est la pression, μ est la viscosité dynamique, ρ est la masse volumique et $\boldsymbol{g}$ est l'accélération de la pesanteur. Pour résoudre numériquement ces équations, une possibilité est d'utiliser la méthode des *volumes finis* qui consiste à 1) discrétiser le domaine en volumes élémentaires ; 2) écrire la conservation des flux d'advection et de diffusion au niveau de chaque volume ; 3) traduire ces équations sous la forme d'un système linéaire ; 4) résoudre le système et avancer dans le temps (Eymard *et al.*, 2000). C'est la méthode implémentée dans le solveur *OpenFOAM* pour *Open Field Operations And Manipulations*[1] et explicitée en annexe B. Lorsque le nombre de Reynolds est important, la résolution directe des équations (DNS pour *Direct Numerical Simulation*) devient très coûteuse car la physique impose de regarder l'écoulement sur une myriade d'échelles spatiales et temporelles. Pour limiter cette complexité, on sacrifie le maillage et change de philosophie de modélisation. Les deux approches classiques consistent soit à résoudre les équations de Navier Stokes **moyennées** (URANS pour *Unsteady Reynolds Average Navier Stokes*) soit à résoudre les équations de Navier Stokes **filtrées** (LES pour *Large Eddy Simulation*).

Résolution des équations moyennées

Le champ de vitesse est décomposé en un champ moyen $\bar{\boldsymbol{U}}$ et un champ de fluctuations $\boldsymbol{u}$ selon $\boldsymbol{U}(\boldsymbol{x},t) = \bar{\boldsymbol{U}}(\boldsymbol{x},t) + \boldsymbol{u}(\boldsymbol{x},t)$. Ici, $\boldsymbol{x}$ désigne la variable spatiale, t désigne la variable temporelle et $\bar{.}$ est une moyenne d'ensemble[2]. L'équation de quantité de mouvement pour le champ de vitesse moyen fait intervenir les *tensions de Reynolds* qui ouvrent le système. Ce tenseur $\boldsymbol{\tau}_{\mathrm{Re}}$ représente l'action de *toutes* les fluctuations sur le champ moyen de sorte que l'équation de quantité de mouvement s'écrit :

$$\frac{\partial \rho \bar{\boldsymbol{U}}}{\partial t} + \boldsymbol{\nabla}.(\rho \bar{\boldsymbol{U}}\bar{\boldsymbol{U}}) = -\boldsymbol{\nabla}\bar{p} + \mu \boldsymbol{\nabla}^2 \bar{\boldsymbol{U}} + \boldsymbol{\nabla}.\boldsymbol{\tau}_{\mathrm{Re}} + \rho \boldsymbol{g}$$

1. Il existe d'autres méthodes pour la mécanique des fluides numériques *e.g* les différences finies, les éléments finis, les méthodes spectrales, Lattice Boltzmann, méthode de Galerkin, etc. On renvoie à l'article de Hosain et Fdhila (2015) pour une classification des méthodes. Pour une présentation plus scolaire des méthodes classiques, on renvoie au livre Tu *et al.* (2018).

2. Avec une moyenne temporelle classique, on retire l'information en temps et on calculerait une solution stationnaire. Dans une approche instationnaire, l'opérateur moyenne doit être vu comme un lissage des fluctuations à un instant donné et il faudrait donc disposer d'un ensemble de réalisations à cet instant.

En modélisant le tenseur $\boldsymbol{\tau}_{\text{Re}}$ par une viscosité turbulente $\mu_t(\boldsymbol{x}, t)$, on ferme les équations. Pour calculer cette viscosité supplémentaire, l'approche classique consiste à résoudre les équations de transport de quantités turbulentes, par exemple l'énergie cinétique turbulente k et son taux de dissipation spécifique ω (Menter, 1994).

Simulation des grandes échelles

Résoudre les équations moyennées (en l'occurrence instationnaires ici) permet uniquement de *modéliser* les effets de la turbulence. Aucune partie du spectre d'énergie turbulente n'est résolue. Dans la simulation des grandes échelles, on reproduit une partie du spectre en résolvant le champ de vitesse filtré. On travaille sur un maillage plus lâche que celui requis pour une simulation numérique directe et on modélise par un tenseur *sous maille* l'effet des échelles non résolues sur celles résolues. L'équation de quantité de mouvement s'écrit (Sagaut et Deck, 2009) :

$$\frac{\partial \rho \tilde{\boldsymbol{U}}}{\partial t} + \boldsymbol{\nabla}.(\rho \tilde{\boldsymbol{U}} \tilde{\boldsymbol{U}}) = -\boldsymbol{\nabla} \tilde{p} + \mu \boldsymbol{\nabla}^2 \tilde{\boldsymbol{U}} + \boldsymbol{\nabla}.\boldsymbol{\tau}_{\text{sgs}} + \rho \boldsymbol{g}$$

où $\tilde{.}$ est l'opération de filtrage qui ne conserve qu'une partie des échelles jusqu'à un certain nombre d'onde de coupure. Tout l'enjeu consiste à définir le tenseur sous maille $\boldsymbol{\tau}_{\text{sgs}}$. Classiquement, le tenseur est modélisé via une viscosité sous maille [3] qu'il faut calculer par exemple avec un modèle de Smagorinski (Zhiyin, 2015).

Remarques

La simulation numérique directe est la plus coûteuse numériquement mais c'est aussi la plus simple à résoudre. Les équations moyennées sont moins coûteuses à résoudre mais demandent un grand effort de modélisation. La simulation des grandes échelles est un compromis entre la fidélité du résultat et le coût de calcul. Avec le développement des techniques d'intelligence artificielle, on cherche aujourd'hui à *apprendre* des modèles de fermeture. On peut citer les travaux de Ling *et al.* (2016) qui utilisent un réseau de neurones contraint par la physique pour apprendre les coefficients d'un modèle de viscosité turbulente. On peut également citer les travaux de Novati *et al.* (2021) qui utilisent l'apprentissage par renforcement pour la simulation des grandes échelles. L'article de Duraisamy *et al.* (2019) expose les limitations et promesses de telles approches.

5.2 | Le cylindre 2D : écoulement périodique

Le premier écoulement étudié est le sillage laminaire d'un cylindre. C'est le cas de validation le plus utilisé dans la littérature. Dans ce qui suit, on présente la

3. On écrit $\boldsymbol{\tau}_{\text{sgs}} = 2\rho\nu_{\text{sgs}}\tilde{\boldsymbol{S}} - \frac{2}{3}\rho k_{\text{sgs}}\boldsymbol{I}$ où $\tilde{\boldsymbol{S}}$ est le tenseur de déformations du champ filtré et k_{sgs} est la part d'énergie cinétique modélisée. Si on filtre toutes les échelles i.e. on modélise toute l'énergie cinétique k, on retombe sur la formulation RANS et la viscosité supplémentaire est ν_t.

physique de l'écoulement et les conditions d'obtention de la base de données. À noter que la configuration est reprise d'un tutoriel OpenFOAM [4].

5.2.1 | Physique de l'écoulement

Dans l'état de l'art, on a présenté la topologie de l'écoulement autour d'un cylindre à partir d'un cours de Feynman *et al.* (1965). À très bas nombre de Reynolds, l'écoulement est rampant car les forces de viscosité l'emportent sur les forces d'inertie. En augmentant légèrement le nombre de Reynolds, on observe la formation de deux tourbillons contra-rotatifs stables. À partir du nombre de Reynolds critique $Re_{c_1} \approx 47$, ce champ de base se déstabilise et on observe des lâchés tourbillonnaires alternés, comme illustré sur la figure 5.1 (Zebib, 1987). Le caractère périodique de l'écoulement est associé au cycle limite d'une bifurcation supercritique de Hopf, identifiée en 2003 par un modèle réduit empirique [5] (Noack *et al.*, 2003).

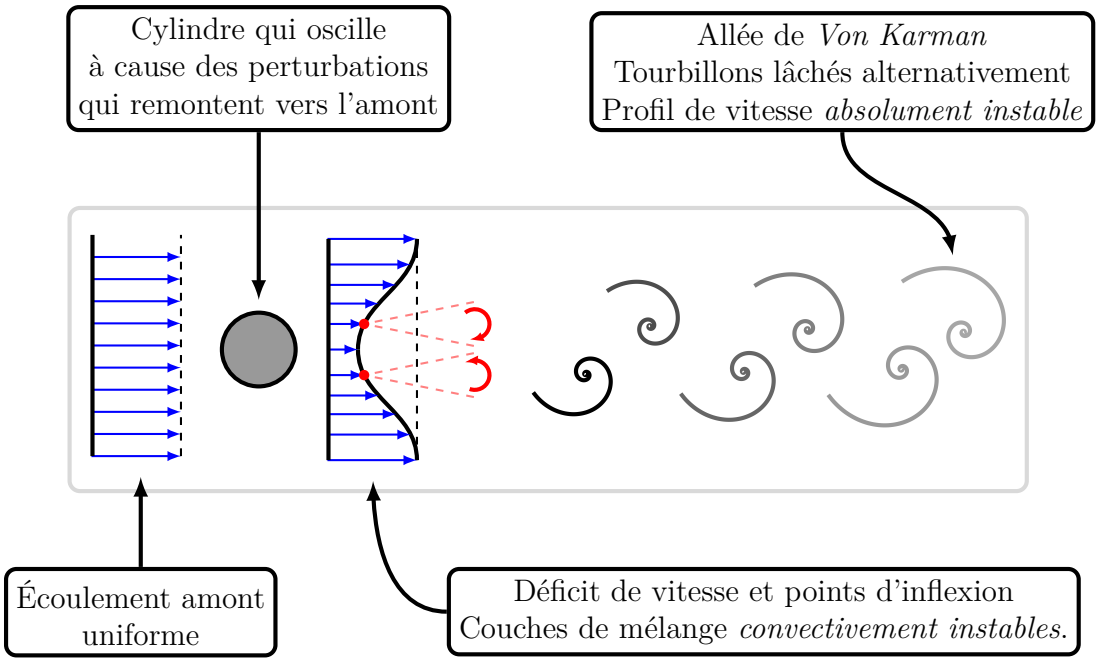


Figure *5.1* – *Topologie prototype de l'écoulement laminaire autour d'un cylindre bidimensionnel à bas Reynolds. À noter que les oscillations du cylindre ne sont visibles que s'il n'est pas contraint.*

4. `https://wiki.openfoam.com/Vortex_shedding_by_Joel_Guerrero_2D`

5. Le champ de vitesse $\boldsymbol{u}$ est décomposé selon $\boldsymbol{u} \approx a_1\boldsymbol{u}_1 + a_2\boldsymbol{u}_2 + a_\Delta\boldsymbol{u}_\Delta$. Les vecteurs $\boldsymbol{u}_1$ et $\boldsymbol{u}_2$ sont les modes les plus énergétiques d'une décomposition modale propre. Le vecteur $\boldsymbol{u}_\Delta$ est construit sur la différence entre le champ moyen et le champ obtenu par moyenne empirique des données disponibles. La dynamique des coefficients (a_1, a_2, a_Δ) reproduit complètement la bifurcation de Hopf.

La forme particulière du sillage vient de l'interaction des deux couches cisaillées formées par le champ de vitesse moyen. Chaque couche est *convectivement instable* car les perturbations sont amplifiées et advectées vers l'aval. Le profil de vitesse complet est *absolument instable* car les perturbations peuvent se propager vers l'amont. Elles viennent d'ailleurs impacter le cylindre qui se met à osciller (s'il le peut) à la fréquence des lâchés tourbillonnaires, avec un nombre de Strouhal d'environ 0.12 (Kumar et Mittal, 2006). L'écoulement commence à se tridimensionnaliser à partir du nombre de Reynolds $Re_{c_2} \approx 180$ et subit une suite de bifurcations (Noack et Eckelmann, 1994; Zhang *et al.*, 1995).

Remarque : l'article de Bagheri (2013) donne une décomposition analytique du sillage laminaire d'un cylindre. C'est un exemple d'application de la théorie de Koopman.

5.2.2 | Configuration calculée

On place un cylindre 2D (fixe) de section circulaire dans un écoulement uniforme. Le diamètre est $D = 2m$ et la vitesse de référence est $U_\infty = 1m.s^{-1}$. Le nombre de Reynolds basé sur (D, U_∞) est $Re = 200$. Le domaine de calcul est défini par $x \in [-10D, 15D]$ et $z \in [-10D, 10D]$ où x est la direction longitudinale et z est la verticale. Le domaine est discrétisé en 9200 cellules et les conditions limites sont telles qu'indiquées sur la figure 5.2.

Le code de calcul résout les équations de Navier-Stokes sans modèle de turbulence (DNS). La solution est calculée sur $450s$ avec un pas de temps $\Delta t = 0.05s$. Pour atteindre le régime périodique des lâchés tourbillonnaires, environ $200s$ sont requises. Les $250s$ restantes sont gardées pour construire les ensembles d'entraînement et de test. La solution est échantillonnée tous les $\Delta t^* = 1$ où t^* est le temps sans dimension défini par :

$$t^* = \frac{tU_\infty}{D}$$

Ainsi, $\Delta t^* = 1$ correspond à une solution échantillonnée tous les $2\Delta t$ et les 250 secondes de calcul permettent de définir 125 clichés. Sur la figure 5.3, on trace la densité spectrale de puissance [6] pour le coefficient de portance et le coefficient de traînée. Le coefficient de portance oscille à la fréquence $f = 0.0955$Hz et permet de calculer le nombre de Strouhal :

6. Pour un signal temporel $s(t)$ de transformée de Fourier $\hat{s}(f)$, la densité spectrale d'énergie est $|\hat{s}(f)|^2$. La densité spectrale de puissance fait en plus intervenir une moyenne temporelle et se définit par $\lim_{T\to\infty} \frac{1}{T}|\hat{s}_T(f)|^2 df$ où $|\hat{s}_T(f)|^2$ est la transformée de Fourier du produit de convolution de $s_T^*(-t)$ avec $s_T(t)$, l'indice T indiquant que l'on observe le signal sur une période T. Pour estimer cette densité spectrale de puissance, on calcule le périodogramme avec la bibliothèque *scipy* de python.

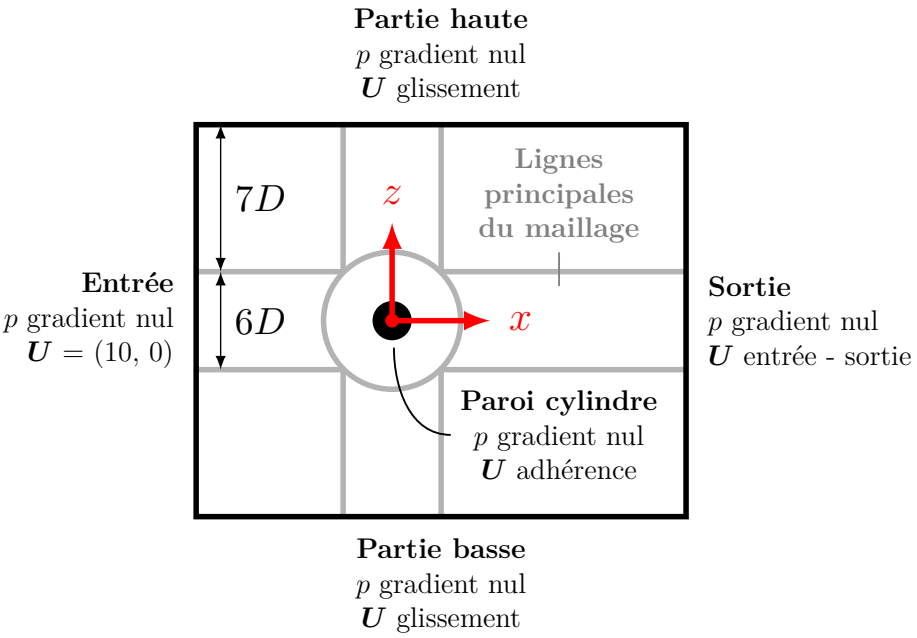
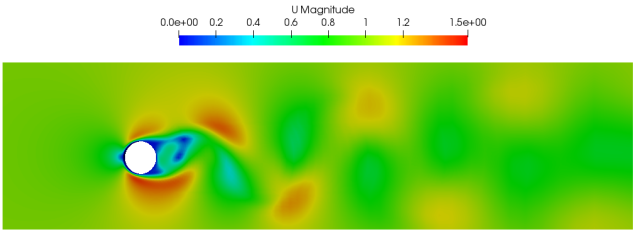


Figure 5.2 – *Conditions limites pour l'écoulement laminaire autour du cylindre. La condition* gradient nul *reprend la quantité de la cellule la plus proche. La condition* entrée - sortie *agit comme un* gradient nul *pour les écoulements dans la direction d'advection ; elle agit comme* valeur fixe *pour des écoulements retour. Les conditions* glissement *et* adhérence *imposent respectivement* $\boldsymbol{U}.\boldsymbol{n} = 0$ *et* $\boldsymbol{U} = \boldsymbol{0}$ *où* $\boldsymbol{n}$ *est la normale à la paroi. La vitesse dans le domaine est initialisé à* $(10\ m.s^{-1}, 0)$. *La pression de référence est* $0\ N.m^{-2}$ *mais cela importe peu car OpenFOAM calcule des différences de pression.*

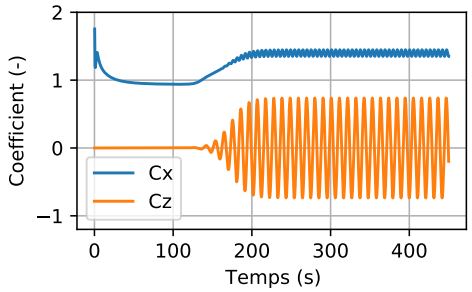
$$\boxed{\mathrm{St} = \frac{fD}{U_\infty} = 0.191}$$

Cette valeur est cohérente avec les résultats rencontrés dans la littérature (Braza *et al.*, 1986). Le coefficient de traînée oscille deux fois plus rapidement que le coefficient de portance car la traînée n'est pas sensible au signe de la vorticité. Un cycle de lâché correspond à un temps sans dimension $T^* = 5.23$ donc 24 cycles sont approximativement observés avec les clichés disponibles. Avec un découpage 70/30 des données, on apprend sur 16 lâchés et teste sur les 8 restants.

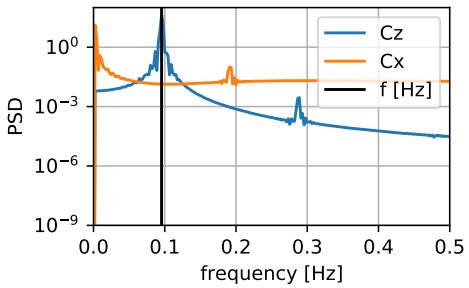
Les outils de réduction, reconstruction et prédiction ne seront pas testées sur tout le domaine de calcul. On se focalise sur la région centrée autour du cylindre et définie par $x \in [-5D, 15D]$ et $z \in [-3.75D, 3.75D]$. Cela fait un total de 5840



(a) Champ de vitesse instantané coloré par le module de vitesse. Les valeurs s'étendent de 0 $m.s^{-1}$ (en bleu) à 15 $m.s^{-1}$ (en rouge)



(b) Coefficients de traînée C_x et de portance C_z.



(c) Densité spectrale de puissance (PSD) des coefficients de force.

Figure *5.3* – *Résultats du calcul numérique de l'écoulement laminaire autour d'un cylindre.*

cellules. Comme on résout deux composantes de la vitesse, la dimension de l'état s'élève à $n = 11680$.

5.3 | Couche de mélange spatiale : écoulement large bande

Le second écoulement étudié est une couche de mélange spatiale. C'est une configuration observée lorsque deux couches de fluide ayant une vitesse différente se rencontrent (voir figure 5.4). Ce cas de validation est inspiré de deux articles qui appliquent la science des données en mécanique des fluides. Le premier propose une méthode non supervisée pour réduire un écoulement et notamment une couche de mélange à une chaîne de Markov (Kaiser *et al.*, 2014). Le second étudie la robustesse des méthodes d'estimation directe pour reconstruire, entre autres, le champ de vorticité d'une couche de mélange (Callaham *et al.*, 2019).

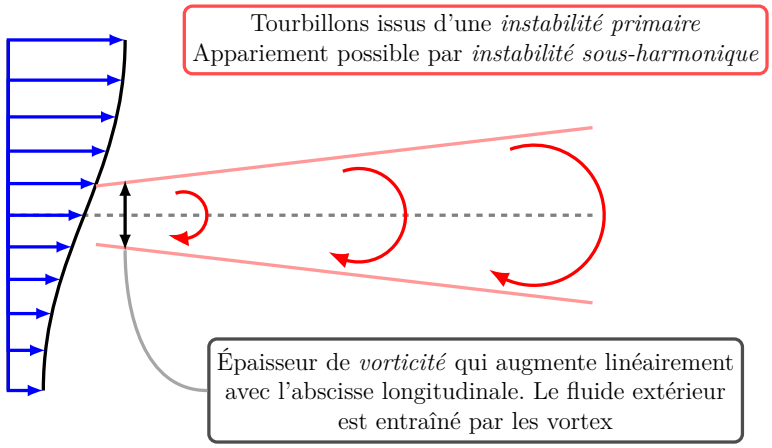


Figure *5.4* – *Topologie d'un écoulement de couche mélange spatiale.*

5.3.1 | Physique de l'écoulement

Le champ de vitesse d'une couche de mélange est *convectivement* instable car toute perturbation de la vitesse est amplifiée en étant convectée vers l'aval. On parle d'instabilité de *Kelvin-Helmhotz*, interprétable en termes d'induction mutuelle [7] (voir figure 5.5) et compréhensible par une analyse de stabilité linéaire (Michalke, 1964). Pour une perturbation de longueur d'onde λ de la nappe de vorticité initiale, on finit par observer une allée de tourbillons discrets espacés de λ. Cette répartition est également instable et les tourbillons peuvent s'apparier pour donner une nouvelle allée de tourbillons espacés de 2λ. Ce processus peut continuer et on parle d'instabilités *sous harmoniques* (Winant et Browand, 1974). Contrairement à l'écoulement laminaire autour du cylindre où une seule fréquence de l'écoulement était amplifiée, la couche de mélange a un spectre large bande car toute une gamme de fréquences est amplifiée (Mansour *et al.*, 1988). Dans la région développée d'une couche de mélange de mélange, le profil de vitesse moyenne est autosimilaire et l'épaisseur de vorticité augmente linéairement avec la distance (Brown et Roshko, 1974).

5.3.2 | Configuration calculée

On effectue une DNS 2D d'une couche de mélange spatiale. L'écoulement haut (rapide) et l'écoulement bas (lent) ont pour vitesses respectives $U_1 = 30\ m.s^{-1}$ et $U_2 = 10\ m.s^{-1}$. En entrée du domaine, on impose le profil de vitesse :

$$U(x=0,z) = U_c + \frac{1}{2}\Delta U \tanh\left(\frac{2z}{\delta_{\omega,0}}\right)$$

où $\Delta U = U_1 - U_2 = 20\ m.s^{-1}$ est la différence de vitesse, U_c est la vitesse de

7. D'après la loi de Biot-Savart, une poche de vorticité induit un champ de vitesse. La vitesse est inversement proportionnelle au cœur de la poche.

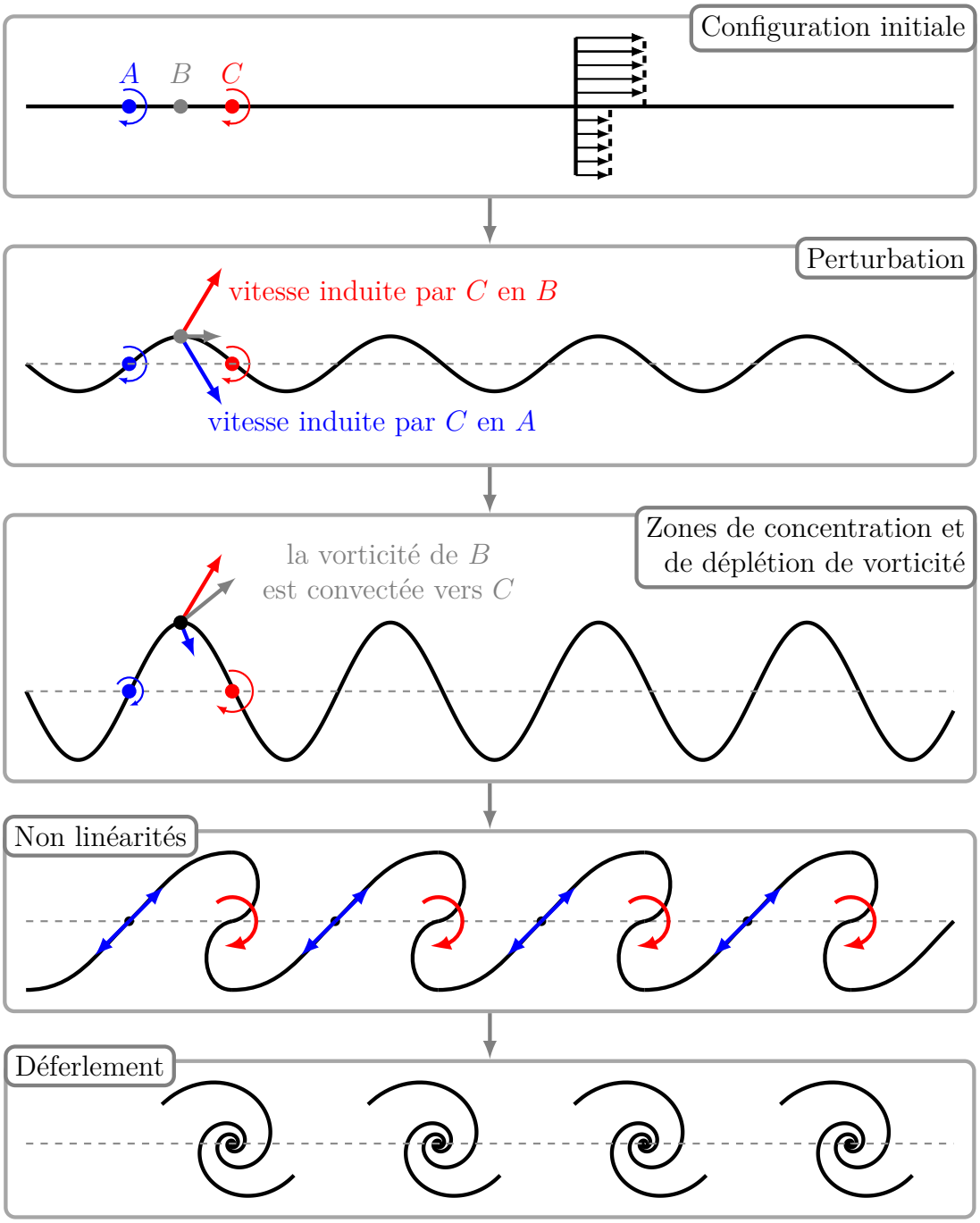


Figure *5.5* – *Mécanisme d'instabilité primaire d'une couche de mélange spatiale. Une perturbation initiale de la nappe de vorticité est amplifiée par induction mutuelle. On observe des zones de concentration et de déplétion de vorticité. La nappe finit par se rompre et on observe une allée de tourbillons discrets. Figure inspirée d'un cours de Laurent Jacquin à Polytechnique.*

convection (vitesse moyenne) et $\delta_{\omega,0} = 1\ m$ est l'épaisseur de vorticité initiale [8]. Le nombre de Reynolds basé sur $\delta_{\omega,0}$ et ΔU est fixé à 500. Pour déclencher l'instabilité de Kelvin-Helmotz, on ajoute des perturbations stochastiques au profil de vitesse : à chaque pas de temps et dans la région $z \in [-2\delta_{\omega,0}; 2\delta_{\omega,0}]$, la vitesse en entrée du domaine est perturbée de ϵ échantillonné d'une loi normale $\mathcal{N}(0, [0.01U_c]^2)$. Ce choix n'a pas de justification physique particulière [9].

Le domaine de calcul est défini par $x \in [0, 180\delta_{\omega;0}]$ et $z \in [-28\delta_{\omega,0}; 28\delta_{\omega,0}]$. Dans la direction longitudinale, les mailles sont de longueur δ_{ω_0}. Dans la direction z, les mailles sont géométriquement étirées de $\delta_{\omega,0}/4$ à $\delta_{\omega,0}/2$ jusque $z = \pm 10\delta_{\omega,0}$. Au delà, les mailles sont géométriquement étirées pour atteindre $\delta_{\omega,0}$ aux extrémités hautes et basses du domaine. Au total, le maillage est composé de 13680 cellules. Les conditions limites sont indiquées sur la figure 5.6.

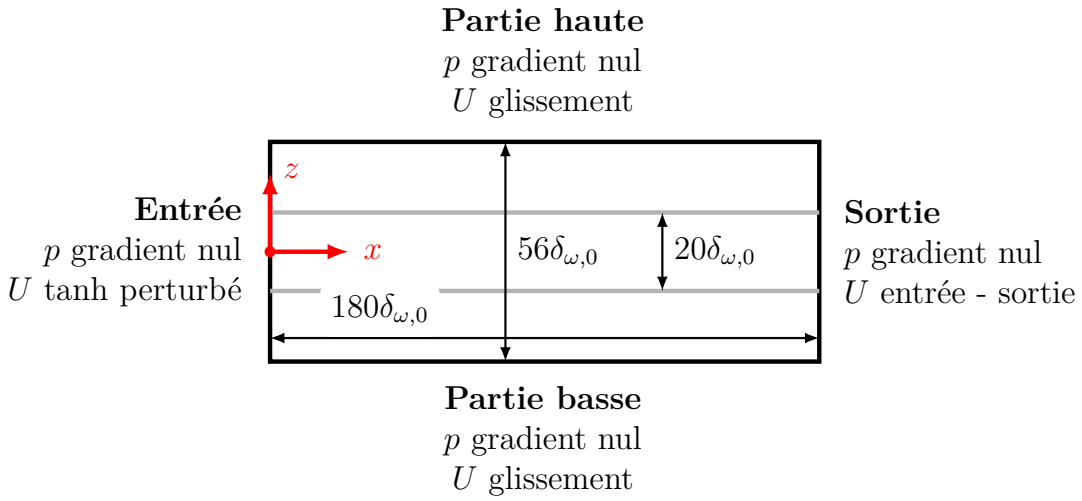


Figure *5.6* – *Conditions aux limites pour la DNS 2D de la couche de mélange.*

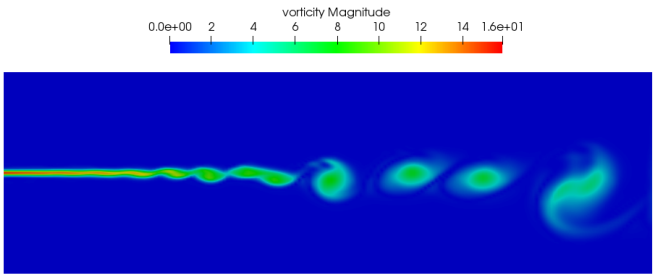
L'écoulement est simulé avec un pas de temps $\Delta t = 0.01\ s$. La solution est d'abord calculée sur $250s$ pour assurer la convergence des statistiques (champ moyen et écart type). En repartant du dernier champ, on simule $135s$ de nouvelles données qui serviront à construire la matrice des clichés. Pour valider le calcul, l'épaisseur de quantité de mouvement [10] δ_θ adimensionnalisée par l'épaisseur de vorticité

8. Définie par $\delta_\omega = \dfrac{\Delta U}{\left(\dfrac{\partial U}{\partial z}\right)_{\max}}$.

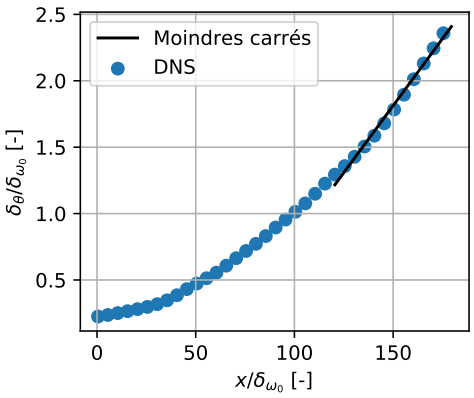
9. À noter qu'en jouant sur les perturbations, on pourrait contrôler l'appariement des vortex (Ko *et al.*, 2008). On peut comprendre comment l'excitation principale impacte l'instabilité grâce au concept de *réceptivité* (Balsa, 1988) mais ce n'est pas l'objectif ici.

10. Définie par $\delta_\theta(x) = \int_{-\infty}^{\infty} \dfrac{U(x,z)}{U_c}\left(1 - \dfrac{U(x,z)}{U_c}\right) dz$.

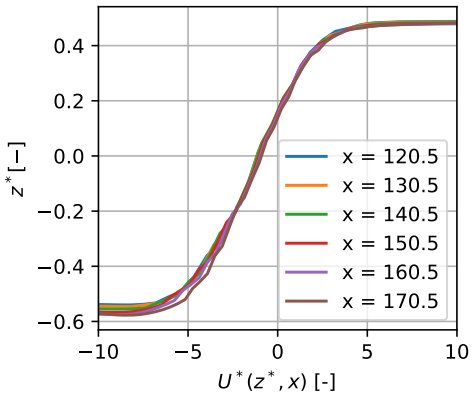
initiale $\delta_{\omega,0}$ est tracée en fonction de la position longitudinale sans dimension $x/\delta_{\omega,0}$. Avec une régression linéaire moindres carrés, on calcule $\mathrm{d}\delta_\theta(x)/dx \approx 0.020$ dans la portion linéaire, une valeur cohérente avec les résultats de la littérature (Attili et Bisetti, 2012). On trace également $U^* = (\bar{U}(x, z^*) - U_c)/\Delta U$ pour plusieurs stations x où z^* est la variable sans dimension $z/\delta_\theta(x)$ et $\bar{U}$ est la vitesse longitudinale moyenne. Les profils se superposent, signe que le champ de vitesse moyen est bien autosimilaire (Yoder *et al.*, 2015).



(a) Module de la vorticité (s^{-1}) pour un instantané.



(b) Évolution de l'épaisseur δ_θ en fonction de la position x. Une régression linéaire dans $x \in [120\delta_{\omega,0}, 180\delta_{\omega,0}]$ indique une croissance de coefficient directeur ≈ 0.0201.



(c) Vitesse moyenne adimensionnée U^* à plusieurs stations x dans la région linéaire. Les profils se superposent, signe que la solution est auto-similaire.

Figure *5.7* – *Résultats du calcul numérique de la couche de mélange spatiale.*

Le temps sans dimension est défini par $t^* = t\Delta U/\delta_{\omega,0}$. La solution est échantillonnée tout les $\Delta t^* = 1$, ce qui permet d'obtenir 2700 clichés. La convection d'un tourbillon prend au plus $L/U_c = 9s$ où L est la longueur totale du domaine, ce qui correspond à une période sans dimension $T^* = 180$. Ainsi, les clichés disponibles couvrent au moins 15 cycles de convection, avec appariement de

vortex possible.

Remarque : pour déterminer les lignes principales du maillage, on a supposé que l'épaisseur de vorticité évoluait linéairement dans tout le domaine. Selon la référence Yoder *et al.* (2015), le taux de croissance est d'environ 0.161λ avec λ le taux d'étalement calculé par $(1-r)/(1+r)$ où $r = 0.33$ est le rapport de vitesse. Ainsi, on estime *a priori* que l'épaisseur de vorticité au bout du domaine est $\delta_{\omega,L} \approx \delta_{\omega,0} + 0.5 \times 0.161 \times 180\delta_{\omega,0} = 15.49\delta_{\omega_0}$. C'est donc dans la portion $z = \pm 10\delta_{\omega,0}$ que l'on met les mailles les plus fines.

5.4 | Cylindre 3D : écoulement turbulent

Le troisième écoulement étudié est le sillage turbulent d'un cylindre de section carrée. Ce cas est reproduit à partir d'un tutoriel OpenFOAM [11].

5.4.1 | Physique de l'écoulement

Comme pour l'écoulement de sillage en aval d'un cylindre de section circulaire, l'écoulement autour d'un cylindre de section carrée subit une suite de bifurcations. En reprenant l'étude de (Bai et Alam, 2018), on distingue différents régimes :

→ Lorsque le nombre de Reynolds dépasse 50, des tourbillons sont lâchés de manière alternée du cylindre et forment une allée de Von Karman laminaire. L'écoulement décolle aux bords avals pour $Re < 120$ tandis qu'il décolle aux bords amonts pour $Re > 120$. Le fait que les décollements soient géométriques (imposés par les bords aigus) est une distinction forte par rapport au cylindre circulaire où les décollements se font par gradient de pression adverse.

→ Le sillage se tridimensionnalise dans la plage $150 < Re < 200$. Les tourbillons se disloquent en envergure avec une longueur d'onde de l'ordre $5.2D$. On observe des boucles tourbillonnaires (rotation autour de la direction d'advection) (Luo *et al.*, 2003). On parle de mode d'instabilité A.

→ Dans la plage $Re \in [190, 250]$, la longueur d'onde de dislocation est plus petite (environ $1.2D$). On parle de mode d'instabilité B.

→ Aux alentours de $Re = 200$, on observe un troisième mode (dit S) dont la longueur d'onde est d'environ $2.8D$. Les structures associées sont lâchés à une fréquence deux fois plus importante que la fréquence de lâché fondamentale.

→ Après l'apparition du mode d'instabilité B, l'allée de Von Karman est turbulente. Les coefficients des forces exercées sur le cylindre varient en fonction du nombre de Reynolds jusque $Re = 2 \times 10^4$. Au delà de cette valeur, les coefficients (et le Strouhal) sont indépendants du nombre de Reynolds (Sohankar, 2006).

11. `https://wiki.openfoam.com/Vortex_shedding_by_Joel_Guerrero_3D`

5.4.2 | Configuration calculée

Un cylindre de diamètre $D = 0.04\ m$ est plongé dans un écoulement uniforme de vitesse $U_\infty = 0.5\ m.s^{-1}$. Le nombre de Reynolds basé sur D et U_∞ vaut $Re = 20000$ et l'écoulement est pleinement turbulent. La solution est initialisée avec un calcul RANS $k-\omega$ SST sur 1000 secondes Menter *et al.* (2003). On effectue alors une simulation des grandes échelles (LES) avec un modèle de Smagorinsky (Smagorinsky, 1963).

Le domaine de calcul est défini par $x \in [-4.5D + D/2, 15D + D/2]$, $y \in [-5D, 5D]$ et $z \in [-7D, 7D]$. Le nombre total de cellules est 271 040 et les conditions limites sont précisées sur la figure 5.8. Le pas de temps est fixé à $\Delta t = 5 \times 10^{-4}\ s$ et 120 secondes sont calculées. Seulement les 105 dernières secondes sont conservées pour construire les clichés. La solution est échantillonné tous les $\Delta t^* = 1$ où le temps sans dimension est $t^* = tU_\infty/D$. On peut donc construire 1312 clichés.

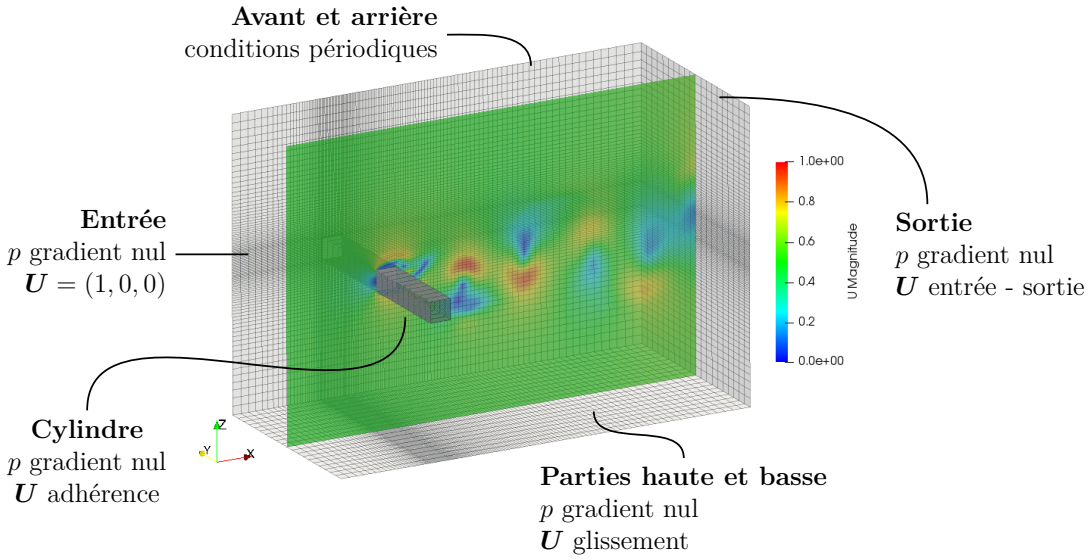
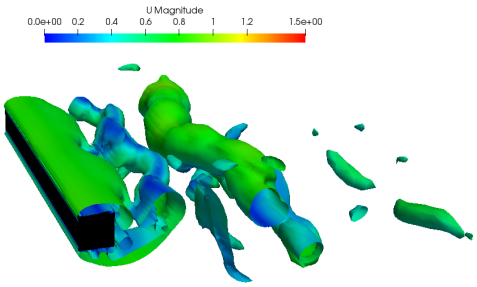


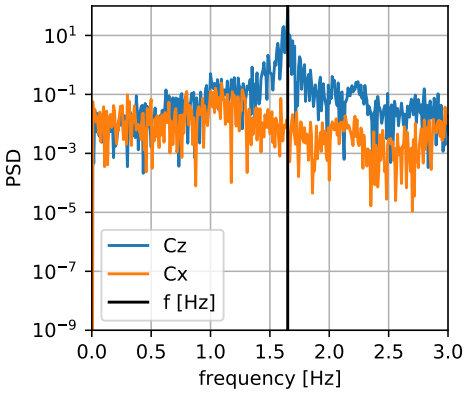
Figure 5.8 – *Conditions aux limites pour la LES de l'écoulement turbulent autour d'un cylindre rectangulaire. On utilise des conditions périodiques sur les faces avant et arrière car les structures de l'écoulement se répètent en envergure. Une coupe permet de visualiser le module de vitesse d'un cliché, avec les zones de survitesse (en rouge) et de déficit de vitesse (en bleu).*

Le calcul est validé grâce aux coefficients de force. Sur la figure 5.9, on trace la densité spectrale de puissance pour le coefficient de portance et le coefficient de traînée. Le Strouhal basé sur la fréquence dominante $f = 1.65$ Hz vaut $\mathrm{St} = 0.132$. Cette valeur est cohérente avec celles rencontrées dans la littérature (Sohankar, 2006). On en déduit qu'un cycle de lâché a une période sans dimension $T^* = 7.58$ donc environ 173 cycles sont observés dans les clichés. Sur la même figure, on trace

également les isosurfaces de $Q = 50$ pour un cliché [12].



(a) Isosurfaces $Q = 50$ d'un cliché. Les isosurfaces sont colorées par le module de la vitesse ($m.s^{-1}$). Un tourbillon en envergure est disloqué.



(b) Densité spectrale de puissance (PSD) des coefficients de force. Le coefficient de portance oscille à la fréquence dominante $f = 1.65$ Hz.

Figure *5.9* – *Résultats pour l'écoulement turbulent autour d'un cylindre rectangulaire.*

Les outils présentés dans la deuxième partie du manuscrit seront testés dans le domaine défini par les points $(-1.25D, -1.25D, -3D)$ et $(12.5D, 1.25D, 3D)$. Cette boîte contient 48 023 cellules et on s'intéresse aux trois composantes de vitesse, d'où une dimension $n = 144\,069$.

5.5 | Écoulement urbain idéalisé : configuration réelle

Pour le dernier écoulement, on considère une tour isolée dans une couche limite atmosphérique. C'est une configuration simplifiée d'écoulement urbain.

5.5.1 | Contexte

Le calcul a été réalisé par la post doctorante H. Toubin. Le travail consistait à utiliser des données expérimentales (Sheng *et al.*, 2018) pour générer

12. Le critère Q permet d'identifier les structures tourbillonnaires (Haller, 2005). Pour $Q > 0$, les rotations sont localement plus grandes que les déformations. Formellement, on a $Q = \frac{1}{2}(\Omega : \Omega - S : S)$ où S et Ω sont respectivement les matrices de déformation et de rotation définies à partir du gradient du vecteur vitesse. L'opération : désigne la double contraction

numériquement les bons taux de turbulence. Ces travaux sont actuellement continués par le post doctorant M. Diop. Dans la suite, on présente surtout la création de la matrice des clichés.

Attention

Pour cette configuration, on est simplement utilisateur.

5.5.2 | Physique de l'écoulement

La couche limite atmosphérique est une couche limite turbulente se développant sur paroi rugueuse. Dans la sous couche inertielle, le profil de vitesse moyenne suit une loi logarithmique (Buschmann et Gad-el Hak, 2003). Le spectre des fluctuations fait quant à lui apparaître de nombreuses échelles spatiales et temporelles, ce qui rend le vent incertain et difficile à prédire (Li *et al.*, 2019). Lorsque qu'un prisme à base carré est plongée dans cet écoulement, on observe la topologie moyenne suivante (McClean et Sumner, 2014) :

→ À l'amont du prisme, on observe un tourbillon fer à cheval qui résulte de la déviation de l'écoulement par l'obstacle (Martinuzzi et Tropea, 1993).

→ Sur les faces latérales et supérieures, il se forme des couches limites qui décollent. Pour des bords aigus, les décollements sont géométriques.

→ Les trois couches limites décollées interagissent et forment un sillage complexe. Selon le rapport hauteur/largeur du prisme, on observe soit la formation d'un tourbillon en arche soit la formation d'une allée de Von Karman. Ces structures sont convectées vers l'aval (Sakamoto et Arie, 1983).

La norme européenne *Eurocode 1* (partie 1-4) permet de caractériser l'action du vent sur un bâtiment et fournit des modèles pour la vitesse moyenne et l'intensité turbulente en fonction du terrain.

5.5.3 | Configuration calculée

La configuration étudiée correspond aux essais réalisés à la soufflerie NSA du CSTB (Sheng *et al.*, 2018). Les données expérimentales, en accord avec les profils de l'Eurocode, sont utilisées pour générer à chaque instant des tourbillons en entrée du domaine de calcul (Mathey *et al.*, 2006). Des détails sur la méthode de génération des vortex sont donnés en annexe C.

On considère une tour de diamètre $D = 10\ cm$ et de hauteur $H = 49\ cm$ plongée dans un écoulement de couche limite atmosphérique avec $U_\infty = 10\ m.s^{-1}$. On définit la hauteur de demi-veine $H_v = 50\ cm$. Le maillage est identique à celui décrit dans la thèse Sheng (2017). Le domaine s'étend sur $8H_v$ dans le sens longitudinal, $2H_v$ dans le sens transversal et $2H_v$ en hauteur. La tour est placée au centre du domaine. La discrétisation spatiale compte $109 \times 28 \times 28$ cellules (voir figure 5.10). Concernant les conditions limites :

— au niveau des parois de la tour et du sol, on impose une condition d'adhérence pour la vitesse et une condition de gradient nulle pour la pression.
— au niveau des parois latérales et supérieures, on utilise une condition de glissement pour la vitesse et un gradient nul pour la pression.
— au niveau de la sortie, on impose une condition entrée - sortie pour la vitesse et gradient nul pour la pression.
— au niveau de l'entrée, on utilise la *vortex method* pour générer la turbulence (voir annexe C).

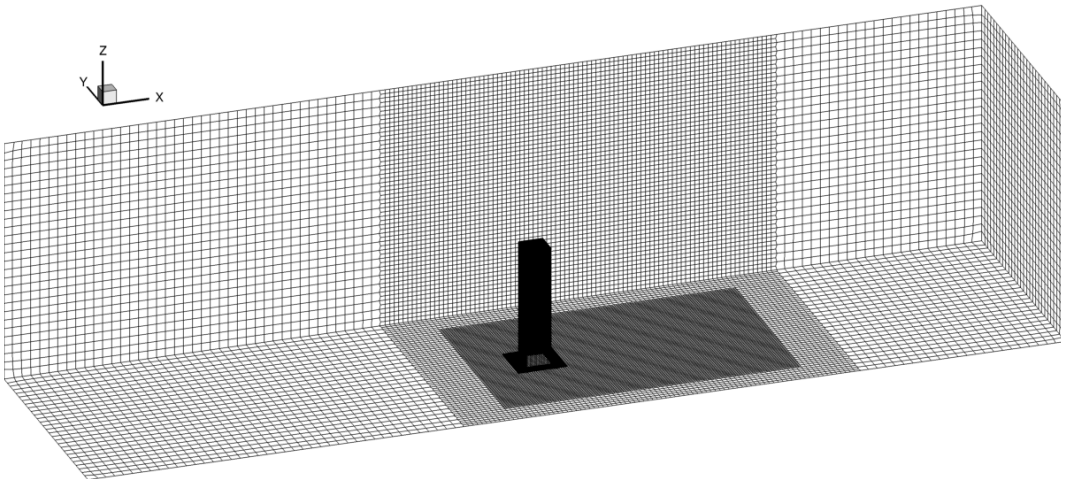
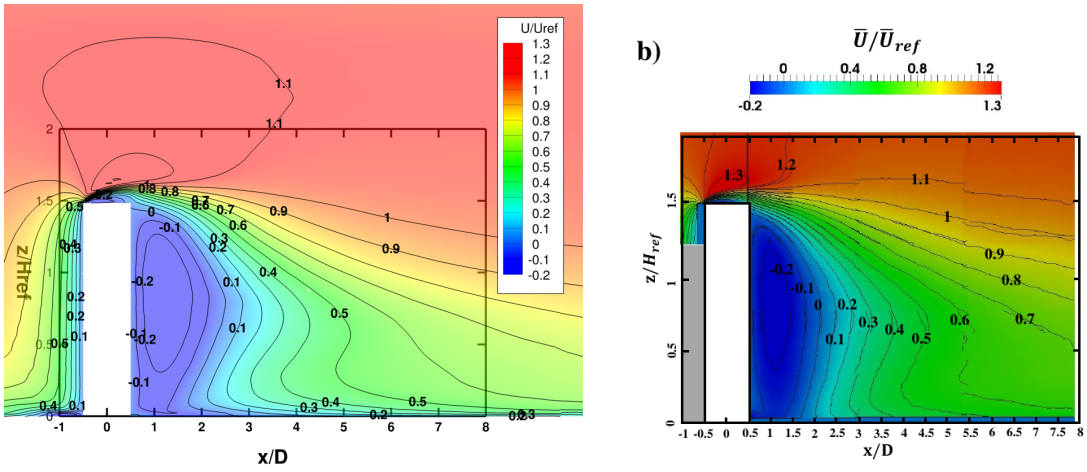


Figure *5.10* – *Maillage pour l'écoulement autour de la tour isolée. La tour est définie par la boîte* $(-D/2, -D/2, 0)$ *à* $(D/2, D/2, H)$. *Plusieurs niveaux de raffinement sont considérés.*

On réalise une simulation des grandes échelles avec un modèle de Smagorinski standard pour la viscosité sous maille (Gousseau *et al.*, 2013). La simulation est effectuée sur 126 secondes, avec un pas de temps $\Delta t = 2.5 \times 10^{-4}$ s. On ne conserve que les 16 dernières secondes pour construire la matrice des clichés. La solution est échantillonnée tous les $\Delta t^* = 1$ où le temps sans dimension est $t^* = tU_\infty/D$. On travaille donc avec 1600 clichés. D'après les résultats expérimentaux[13], on peut s'attendre à un strouhal de l'ordre de 0.1 donc une période sans dimension $T^* = 10$. Ainsi, les données contiennent environ 160 cycles de lâchés.

Avec les figures 5.11 à 5.13, on vérifie qualitativement que les champs de vitesse moyenne concordent avec les images expérimentales. Les outils de réduction et de reconstruction ne seront pas testées dans le domaine tridimensionnel car la matrice de clichés est très lourde à charger en mémoire[14]. On teste donc les outils dans différents plans, définis dans le tableau 5.1.

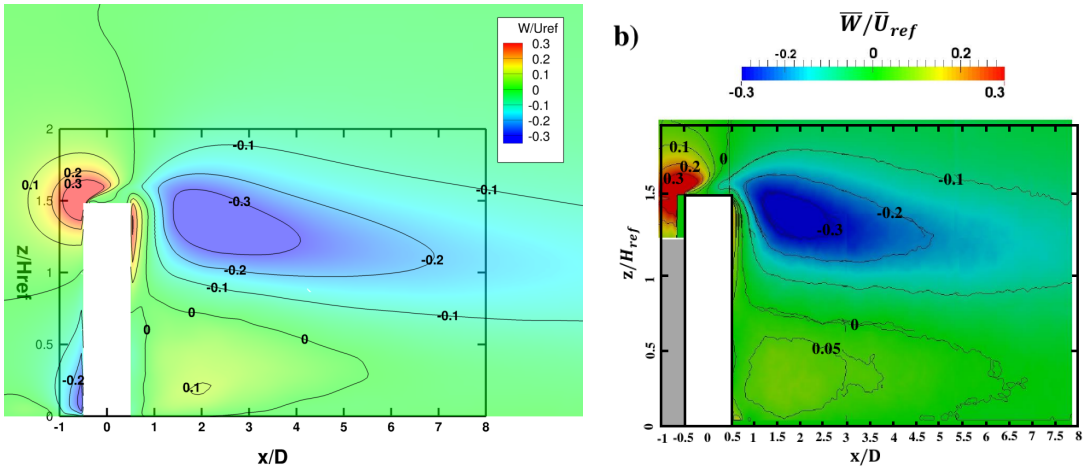
13. Le calcul n'a pas été validé quantitativement pendant le post doctorat. Les capteurs n'ayant pas été complètement définis, les spectres de pression sur les parois de la tour n'ont pas pu être



(a) Résultats numériques

(b) Résultats expérimentaux (PIV)

Figure *5.11* – *Contours de la vitesse longitudinale moyenne (dans un plan perpendiculaire au sol et dans l'axe de la tour).*



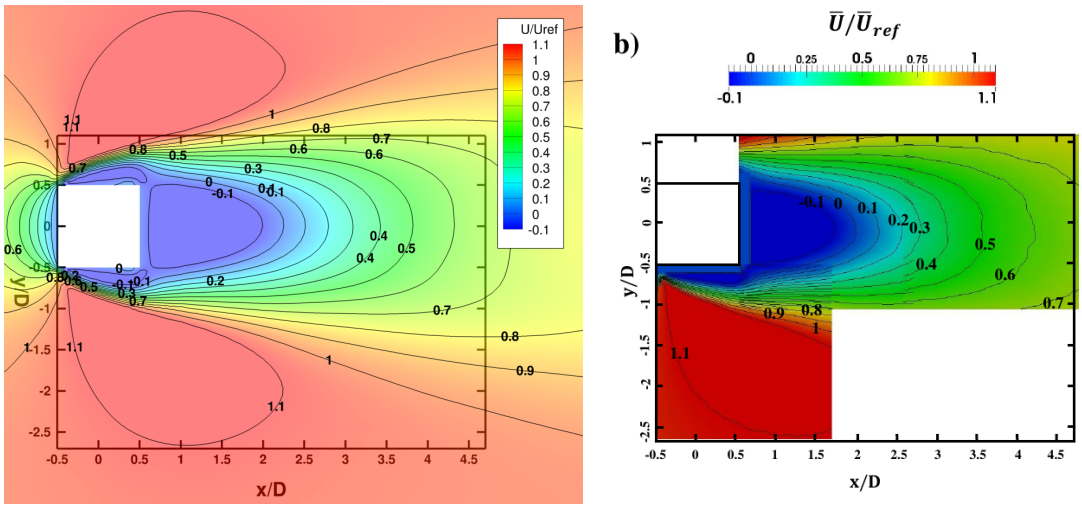
(a) Résultats numériques

(b) Résultats expérimentaux (PIV)

Figure *5.12* – *Contours de la vitesse verticale moyenne (dans un plan perpendiculaire au sol et dans l'axe de la tour).*

comparés.

14. Environ 50 Go. Il faut des machines avec une grande mémoire vive pour effectuer directement des calculs comme la décomposition en valeurs singulières. C'est un problème opérationnel : dans le cas d'écoulements réalistes, le volume des données pour mettre en œuvre les méthodes exposées devient prohibitif.



(a) Résultats numériques **(b)** Résultats expérimentaux (PIV)

Figure *5.13* – *Contours de la vitesse longitudinale moyenne (dans un plan parallèle au sol, à mi-hauteur de la tour).*

5.6 | Synthèse du chapitre

Dans ce chapitre, on a présenté quatre écoulements de complexité croissante sur lesquels les outils développés dans la deuxième partie du manuscrit seront testés. On dispose de données pour :

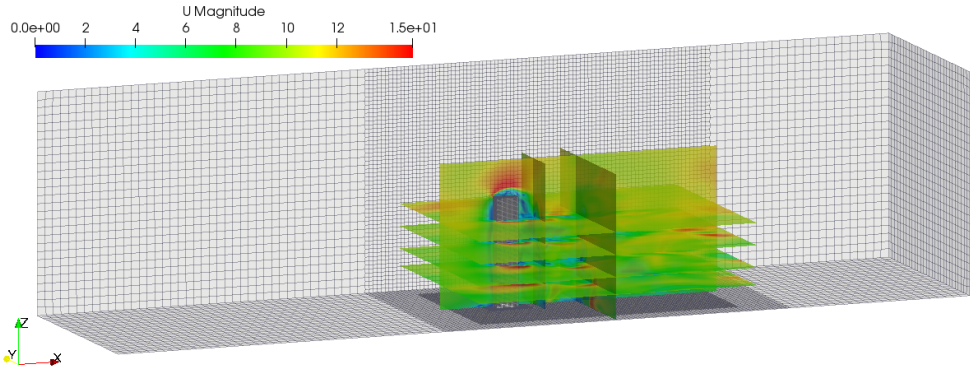
- L'écoulement laminaire autour d'un cylindre. C'est le cas de validation le plus classique utilisé dans la littérature.
- Un écoulement de couche de mélange. C'est une configuration 2D.
- L'écoulement turbulent autour d'un cylindre rectangulaire. C'est un cas 3D.
- L'écoulement autour d'une tour isolée. On ne testera que les méthodes de réduction et de reconstruction dans des petits volumes tridimensionnels. Dans l'idéal, il faudrait reprendre cette simulation et interpoler sur un maillage régulier et plus lâche [15].

Le nombre de données à utiliser est défini à partir d'un pas de temps sans dimension. Disposant maintenant de la matrice de clichés u et donc de u^{train} et u^{test} pour quatre écoulements, on peut appliquer les méthodes de réduction, reconstruction et prédiction. Les résultats sont présentés dans le **chapitre 6**.

15. Si on reprend le problème initial de navigation sécurisée d'un drone, une partie des échelles est filtrée. On n'a donc pas besoin d'estimer l'écoulement en tous les points de maillage qui étaient requis pour calculer un écoulement physique.

Nom du plan	$(x_{\min}, y_{\min}, z_{\min})$	$(x_{\max}, y_{\max}, z_{\max})$	Nombre cellules
11	$(-0.3, -0.3, 0.09)$	$(1.0, 0.3, 0.11)$	24 414
12	$(-0.3, -0.3, 0.19)$	$(1.0, 0.3, 0.21)$	31 446
13	$(-0.3, -0.3, 0.29)$	$(1.0, 0.3, 0.31)$	31 454
14	$(-0.3, -0.3, 0.39)$	$(1.0, 0.3, 0.41)$	24 414
21	$(-0.3, -0.01, 0.0)$	$(1.0, 0.01, 0.65)$	28 432
31	$(0.14, -0.15, 0.0)$	$(0.16, 0.15, 0.65)$	4 964
32	$(0.34, -0.2, 0.0)$	$(0.36, 0.2, 0.65)$	6 424

(a) Définitions numériques des plans.



(b) Visualisation des plans dans le volume.

Tableau 5.1 – *Définition des petits volumes 3D (que l'on appelle plans) sur lesquels on testera les méthodes de réduction et de reconstruction.*

6. Estimation orientée données des quatre écoulements

Ce chapitre présente les résultats obtenus suite à l'application des méthodes d'estimation aux différents écoulements. Pour ne pas noyer le lecteur avec des tableaux de valeurs, la majeure partie des scores/erreurs d'estimation est en annexe. L'objectif du chapitre est plutôt de caractériser les différentes méthodes orientées données pour estimer un écoulement. La première partie est consacrée à la réduction des écoulements. On étudie en particulier les spectres des champs auto-encodés et la robustesse des différents décodeurs. La seconde partie est focalisée sur les méthodes de reconstruction. En plus de présenter les grilles de capteurs et des exemples d'estimation, on étudie la robustesse des méthodes régressives aux mesures bruitées. La troisième section traite du problème de prédiction. On y présente des modèles dynamiques de l'état latent construits par approximation finie de l'opérateur de Koopman.

Sommaire

6.1 | Réduction des écoulements

Cette section compare quatre méthodes orientées données pour la réduction du champ de vitesse fluctuante. On rappelle que les outils impliqués sont la décomposition orthogonale propre, l'auto-encodeur linéaire, l'auto-encodeur linéaire variationnel et l'auto-encodeur non linéaire variationnel.

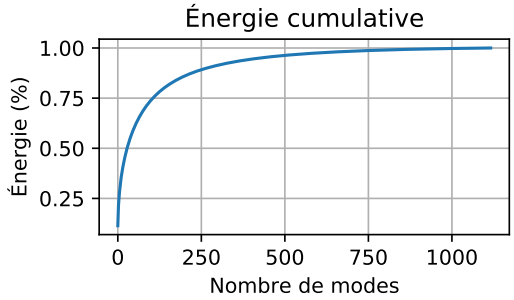
6.1.1 | Décomposition en modes propres orthogonaux

La décomposition en modes propres orthogonaux (POD) détermine une base de $\mathbb{R}^n$ dans laquelle les champs de vitesse fluctuante d'entraînement sont décorrélés. Les directions sont construites hiérarchiquement pour restituer la variance [1] et la troncature aux r premiers modes donne une approximation de faible rang des clichés. Cette idée est illustrée sur la figure 6.1 avec le champ de vitesse dans le plan vertical de la tour. Pour de nombreuses applications, le choix du rang est empirique, basé sur un pourcentage de l'énergie à conserver. Dans la suite, on considère plusieurs niveaux de réduction. Pour l'écoulement laminaire autour d'un cylindre, le nombre de modes doit restituer 40%, 80% ou 99% de l'énergie cinétique. Pour la couche de mélange spatiale et l'écoulement turbulent autour du cylindre rectangulaire, les niveaux sont de 40%, 80% ou 95% de la variance. Pour tous les plans de l'écoulement autour du bâtiment, seule une réduction à 80% d'énergie est considérée [2]. Les tables 6.1 et 6.2 donnent les dimensions correspondantes, déterminées à partir de l'énergie cumulée [3] en fonction du nombre de modes.

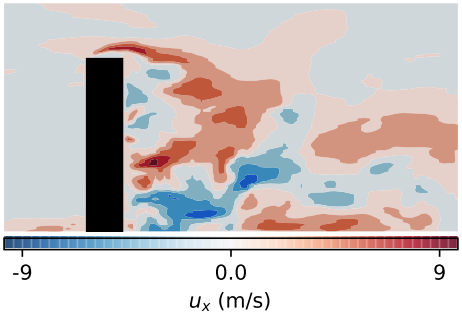
Configuration	n	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$ (cylindre laminaire) ou $r_{0.95}$ (autres)
Cylindre laminaire	11 680	1	2	5
Couche de mélange	27 360	3	11	32
Cylindre turbulent	144 069	2	21	177

Tableau 6.1 – *Dimensions pour les trois premières configurations. On rappelle que n désigne la dimension du champ complet (nombre de points de maillage à multiplier par le nombre de composantes de vitesse) et r est la dimension réduite (indexée par le pourcentage d'énergie)*

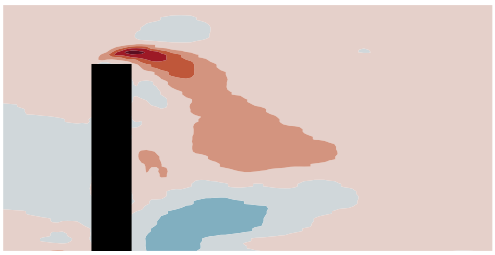
1. C'est-à-dire l'énergie cinétique turbulente car on décompose des champs de vitesse.
2. On rappelle que la POD n'est pas effectuée dans le volume complet. Charger en mémoire la matrice des clichés (environ 50Go pour 1600 pas de temps) n'était pas envisageable, d'où le travail sur de plus petits volumes que l'on appelle plans (environ 800 Mo chacun).
3. On rappelle qu'elle est calculée à partir des valeurs propres de la matrice de covariance ou les valeurs singulières de la matrice des clichés.



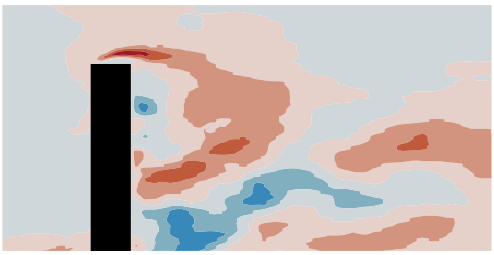
(a) Énergie cumulative en fonction du nombre de modes.



(b) Vitesse fluctuante instantanée (composante longitudinale) de référence



(c) Approximation qui restitue 40% de la variance (rang 10)



(d) Approximation qui restitue 80% de la variance (rang 139)

Figure *6.1* – *Variance cumulative des clichés d'entraînement en fonction du nombre de modes pour l'écoulement urbain (plan 21). Une approximation de rang faible du champ de vitesse instantané est une combinaison linéaire finie des directions principales.*

Plan de l'écoulement	11	12	13	14	21	31	32
n	73 242	94 338	94 362	73 242	85 296	14 892	19 272
$r_{0.8}$	106	119	113	102	139	68	63

Tableau 6.2 – *Dimensions pour l'écoulement urbain, dans les différents plans considérés.*

Les directions principales (modes propres) sont des objets adaptés pour réécrire *linéairement* et de manière *orthogonale* le film de l'écoulement. La hiérarchie qui est imposée par la décomposition dessine un lien avec le principe de cascade énergétique mais *un* mode n'équivaut pas à *un* mécanisme physique en particulier. Les figures 6.2 (cas 2D) et 6.3 (cas 3D) illustrent quelques modes propres. Sur la

couche de mélange, il apparaît clairement que le premier mode est formé de grosses structures à dynamique lente tandis que le cinquième mode est composé de plus petites structures à dynamique plus rapide. En ce qui concerne l'écoulement laminaire autour du cylindre, les premier et second modes sont semblables, juste déphasés. Cette paire de modes est caractéristique d'ondes transportées et chaque mode a une dynamique linéaire (cosinus et sinus) qui traduit que les lâchés tourbillonnaires sont des fluctuations périodiques autour du champ de vitesse moyenne. Concernant le cylindre rectangulaire et l'écoulement urbain (ici le plan 11), l'évolution temporelle des modes est clairement plus complexe.

Pour l'exemple de la couche de mélange, les modes POD sont plus spécifiquement caractérisés dans la littérature. Les deux premières paires représentent la convection de structures tourbillonnaires dans la direction longitudinale (Rajaee *et al.*, 1994). La première paire (modes 1 et 2) est associée à l'instabilité sous-harmonique [4] tandis que la deuxième paire (modes 3 et 4) représente les structures de Kelvin-Helmotz (fréquence plus importante et nombre d'onde croissant dans la direction d'advection, voir Laizet *et al.* (2010)). Les modes d'ordre plus élevé contiennent un mélange des fréquences. En partitionnant les coefficients POD, Kaiser met en évidence qu'une diminution de l'amplitude des deux premiers modes est associée à une augmentation de l'amplitude des troisième et quatrième modes (Kaiser *et al.*, 2014). Les auteurs définissent alors des super-clusters pour caractériser l'appariement de vortex et les structures de Kelvin-Helmotz. Cet exemple est intéressant car il illustre qu'une décomposition POD (structures résolues par énergie) est ré-interprétable dans le même sous espace décrit différemment [5] (structures résolues par phase).

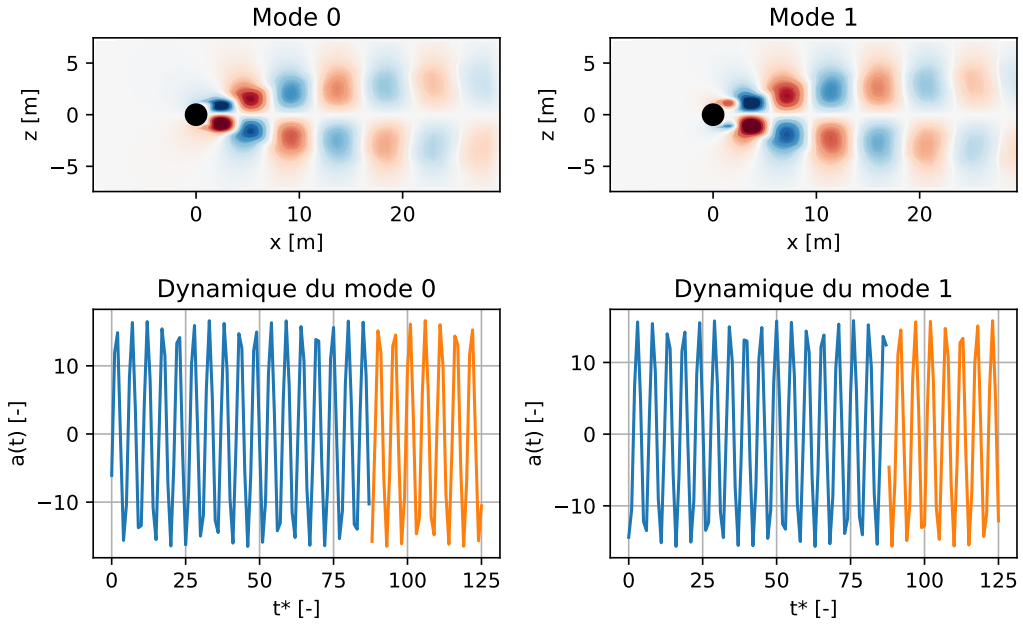
6.1.2 | Réduction par auto-encodeur neuronal

La décomposition en modes propres orthogonaux peut se reformuler en un réseau de neurones avec une seule couche cachée dont le nombre de neurones est r. Si la condition d'orthogonalité n'est pas imposée, le réseau est un auto-encodeur linéaire non orthogonal (LAE) dont les poids du décodeur sont interprétés comme les modes spatiaux. L'apprentissage de ces modes s'effectue par descente de gradient, en minimisant l'erreur quadratique moyenne entre les clichés réels et les clichés auto-encodés. Sur la figure 6.4, on peut voir l'histoire de la fonction coût et un exemple de modes pour les deux écoulements de cylindre. Il apparaît que l'ajout des non orthogonalités facilite l'interprétation des modes de l'écoulement, qui font clairement apparaître la forme de l'allée de Von Karman [6].

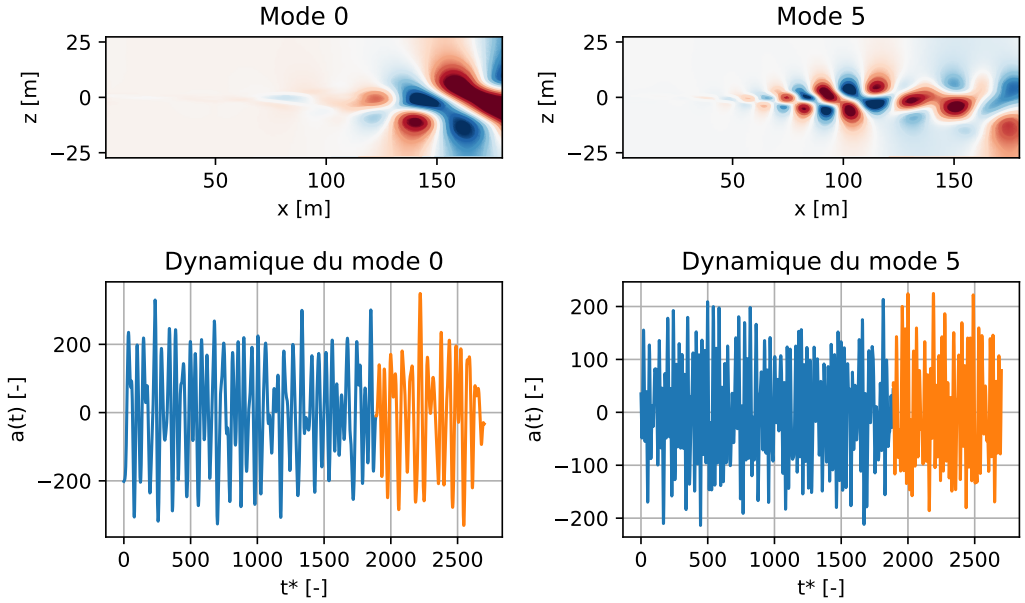
4. Lorsque l'entrée est mécaniquement forcée à une fréquence angulaire ω, les interactions non linéaires avec la sous-harmonique $\omega/2$ sont à l'origine de l'appariement des vortex. On parle de croissance sous-harmonique, un phénomène analysé pour la première fois en 1967 (Kelly (1967), Mansour *et al.* (1988)).

5. Les vecteurs de base étant les centres des partitions, obtenus par combinaison linéaire des modes POD.

6. Attention à ne pas sur-interpréter, un mode est toujours un objet mathématique. Mais contrairement à l'analyse en composantes principales, l'auto-encodeur ne favorise pas des directions de variance maximale *orthogonale* donc l'aspect convectif est plus visible.

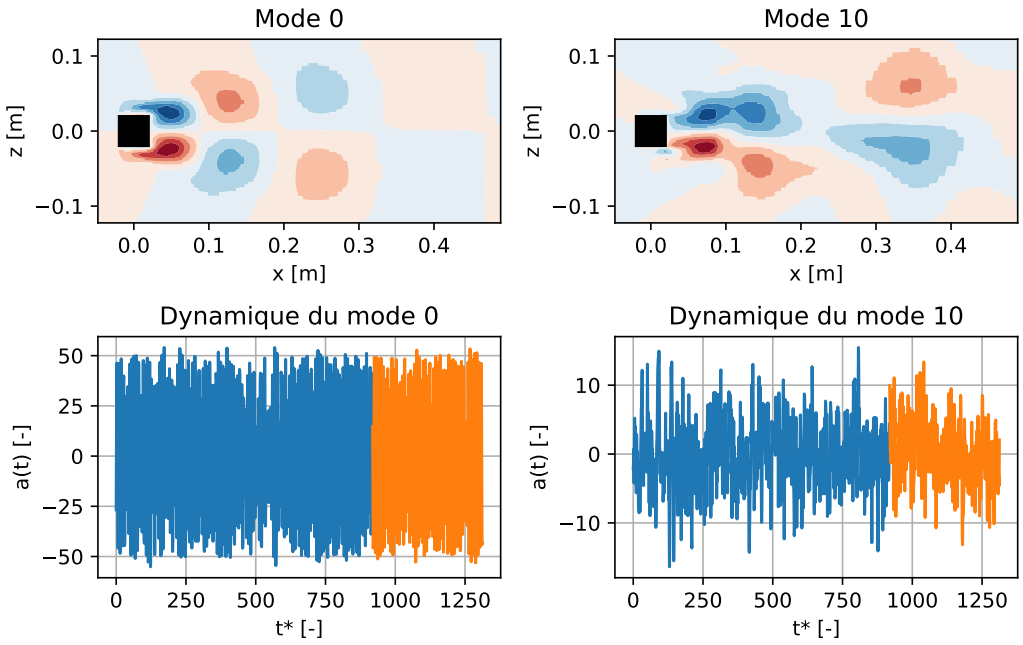


(a) Les deux premiers modes POD de l'écoulement laminaire autour du cylindre. La dynamique des modes est périodique ce qui est cohérent avec la dynamique du lâché tourbillonnaire.

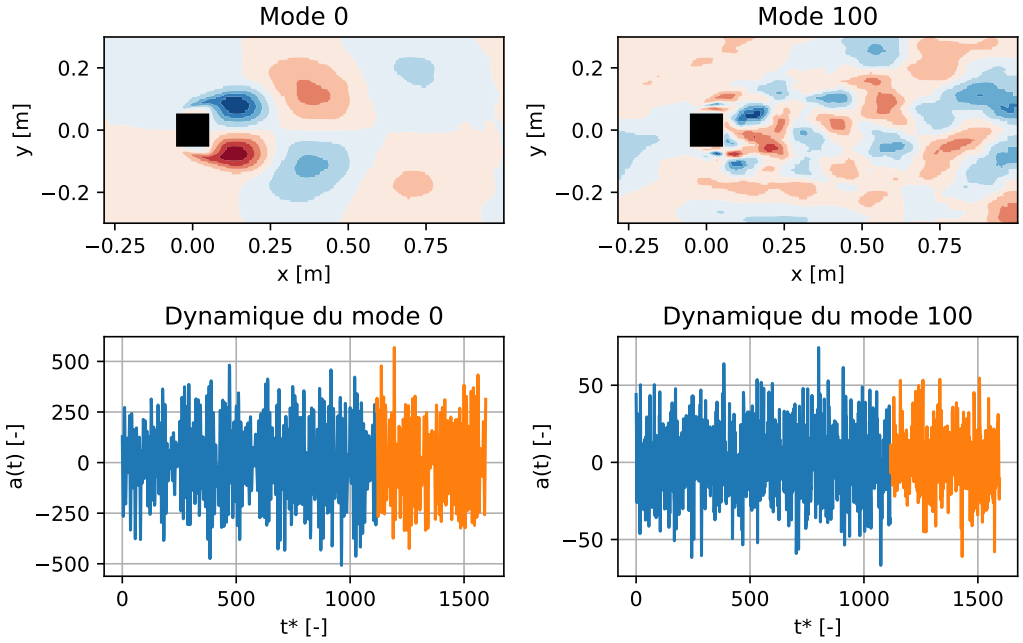


(b) Deux modes POD pour la couche de mélange spatiale. La dynamique du cinquième mode est plus rapide que la dynamique du mode le plus énergétique.

Figure *6.2* – *Quelques modes POD pour les configurations 2D. Les modes sont extraits sur les clichés d'entraînement, ici encodées en séries temporelles bleues. Les séries oranges correspondent aux clichés de test encodés. Les modes sont représentés par les contours de la vitesse fluctuante longitudinale.*

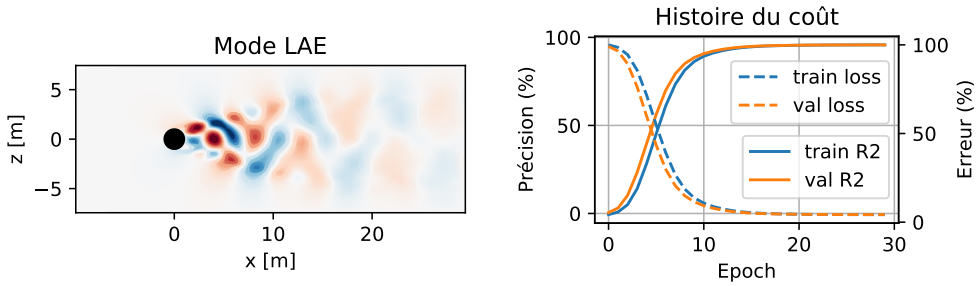


(a) Deux modes POD pour l'écoulement autour du cylindre rectangulaire.

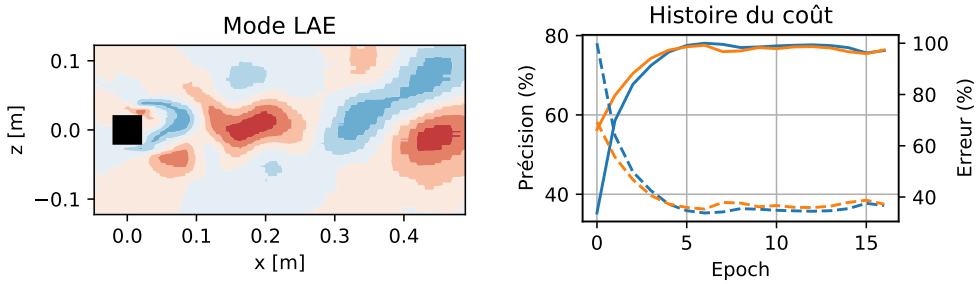


(b) Deux modes POD pour l'écoulement urbain dans le plan 11.

Figure *6.3* – *Quelques modes POD pour les configurations 3D. Comme pour la figure précédente, les modes sont représentés par les contours de leur composante longitudinale. Pour effectuer les visualisations, les données sont d'abord interpolées sur une grille cartésienne puis le plan médian est extrait.*



(a) Cas de l'écoulement laminaire autour du cylindre circulaire



(b) Cas de l'écoulement turbulent autour du cylindre rectangulaire

Figure *6.4* – *Exemple de modes obtenus par LAE. Contrairement à la POD, les modes sont non orthogonaux et déterminés par calcul itératif. La précision est quantifiée par le coefficient de détermination moyen tandis que l'erreur renvoie à l'erreur quadratique moyenne normalisée par l'erreur d'entraînement au premier epoch. L'erreur de validation est utilisée pour stopper l'apprentissage.*

Pour régulariser l'espace latent, les champs de vitesse sont plutôt encodés en distributions. Avec un encodage et un décodage linéaire, le réseau modifié est appelé auto-encodeur linéaire variationnel (LVAE). Bien que les directions POD, LAE et LVAE soient différentes, elles génèrent le même sous espace de $\mathbb{R}^n$ (voir Plaut (2018) pour la démonstration). Ce résultat est justifié sur le cas du cylindre rectangulaire avec la figure 6.5 où l'on compare les matrices de covariance de l'état latent pour les trois méthodes de réduction. Écrites dans les bases initiales, les matrices de corrélation LAE et LVAE sont pleines tandis que la matrice de corrélation POD est diagonale. Mais écrites dans les bases principales des décodeurs, les matrices de corrélation LAE et LVAE sont diagonales et les corrélations sont *quasi-identiques* à l'énergie restituée par la POD.

En considérant un encodage et un décodage non linéaires, le réseau LVAE devient un auto-encodeur variationnel (VAE). Dans ce cas, il faut écrire la

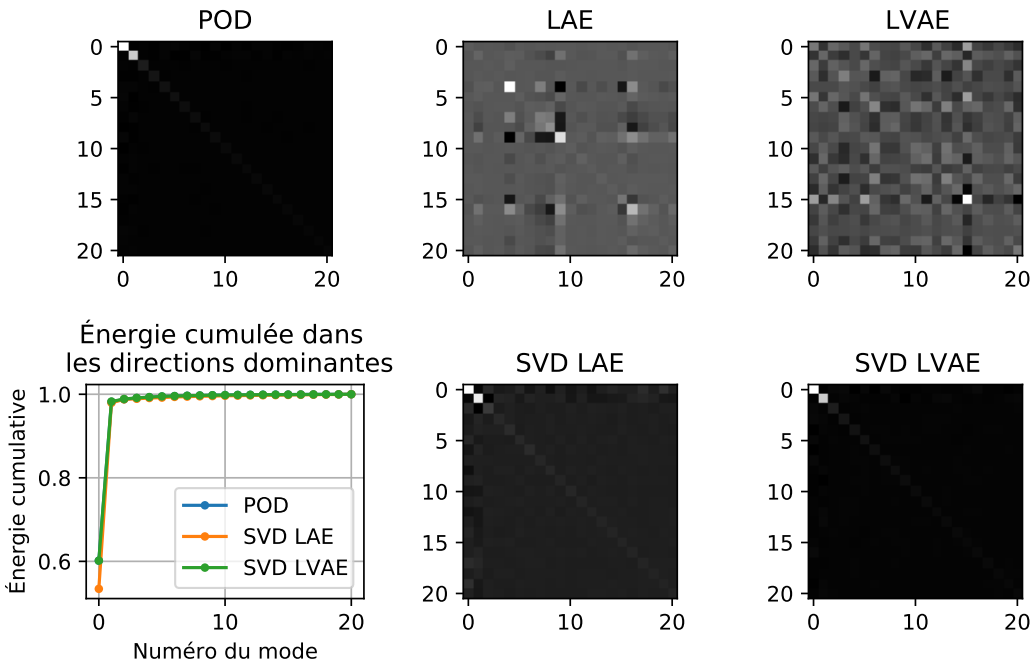


Figure *6.5* – *Matrices de covariance de l'état latent obtenu par réduction linéaire. Le cas considéré est l'écoulement autour du cylindre rectangulaire avec $r = 21$. Dans les bases initiales (première ligne), les matrices de covariance pour les réductions LAE et LVAE sont pleines. Dans les bases principales des décodeurs (deuxième ligne), les matrices de corrélations sont diagonales et les niveaux d'énergie correspondent à la réduction POD.*

décomposition en valeurs singulières des poids du décodeur pour avoir des modes interprétables [7]. La figure 6.6 illustre deux modes dominants pour le cas de la tour (plan vertical) et pour le cas du cylindre rectangulaire. Si les premiers modes ont des similarités avec les modes POD, ce n'est pas le cas pour les modes d'ordre élevé. Cela signifie que le décodeur capte la majeure partie de la variabilité (erreur quadratique) mais ne se focalise pas uniquement sur ce critère énergétique (compromis avec la divergence de Kullback-Leibler). En étudiant l'histoire de la fonction coût, on observe du sur-apprentissage pour l'écoulement autour de la tour car les performances entre la réduction des données d'entraînement et de test sont significativement différentes. De plus, la précision (en termes de coefficient de détermination) atteint au mieux 20% sur les données de validation (figure 6.7). Cette faiblesse peut s'attribuer à l'architecture du réseau résolument basique qui n'a pas été optimisée. Enfin, on note que l'encodage VAE respecte difficilement une continuité temporelle. La dynamique de l'état latent fait apparaître des *sauts* qui ne sont pas observés avec les réductions linéaires. Cette propriété, illustrée

7. Lus tels quels, les modes ressemblent à du bruit.

pour la couche de mélange sur la figure 6.8, est rédhibitoire pour l'estimation des écoulements. Bien que ce problème de régularité est observé dans une moindre mesure pour les écoulements de cylindre [8], il semble nécessaire (pour des investigations futures) d'ajouter une contrainte temporelle lors de la construction d'un espace latent VAE.

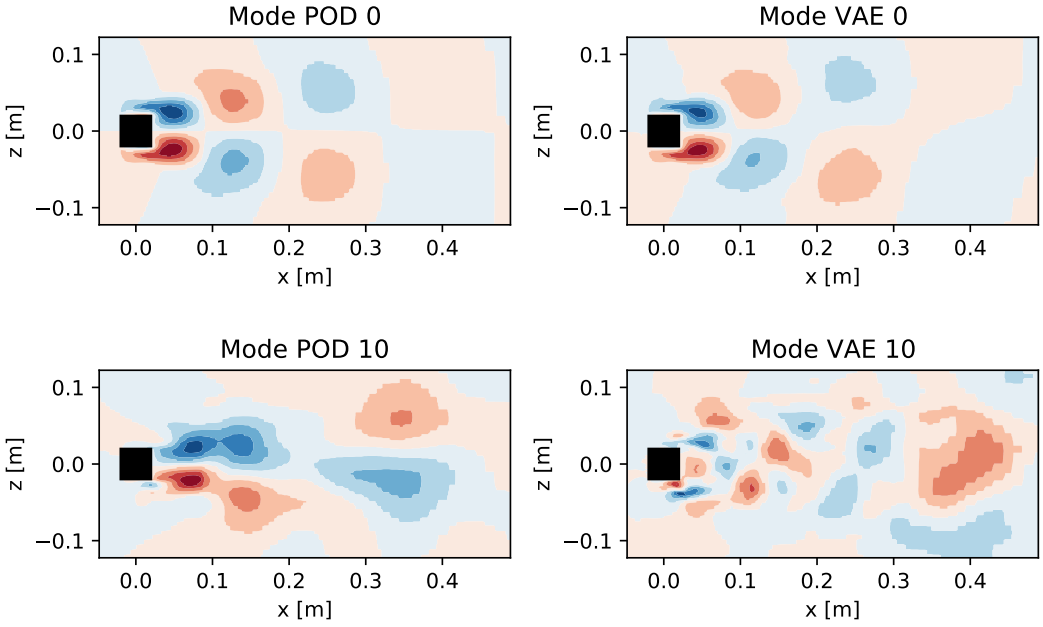
Sur la figure 6.9, on visualise en 2D l'espace latent des auto-encodeurs LAE, LVAE et VAE. Ces visualisations sont obtenues en extrayant les deux composantes principales de l'état latent moyen (notées $a_{2D,1}$ et $a_{2D,2}$) et en projetant les ellipses de confiance à 95% selon les directions principales. Les champs de vitesse ont été partitionné en trois classes par k-moyennes afin de faciliter l'interprétation. On est d'abord rassuré pour l'écoulement laminaire autour du cylindre car les trajectoires LAE, LVAE et VAE se situent sur un cycle limite. On est ensuite rassurés pour les réductions variationnelles des trois écoulements car les domaines de variations sont restreints et les distributions sont encouragées à se superposer. Ainsi, on peut s'attendre à ce que des champs de vitesse pertinents soient obtenus lors du décodage d'état latents qui ne se situeraient pas exactement sur la trajectoire d'apprentissage.

Un résultat intéressant concerne la dynamique latente. Une transformée de Fourier sur les séries temporelles de l'état réduit permet de calculer les fréquences d'oscillation des modes. La figure 6.10 montre l'histogramme des fréquences qui dominent pour chaque méthode de réduction et trois écoulements (cylindres et plan 11 de la tour). Concernant la POD, la hiérarchie imposée par l'algorithme est clairement respectée : les premiers modes ont fondamentalement une dynamique lente et les modes d'ordre élevé ont une dynamique fondamentalement rapide. Pour les réductions non orthogonales, la plupart des modes oscillent fondamentalement à la même fréquence i.e. $f = 0.0952$ Hz pour l'écoulement laminaire du cylindre, $f = 1.65$ Hz pour l'écoulement turbulent du cylindre et $f = 10.08$ Hz pour l'écoulement de la tour. Ces fréquences ne sont pas anodines et correspondent aux fréquences macroscopiques de l'écoulement, utilisées pour calculer les nombres de Strouhal.

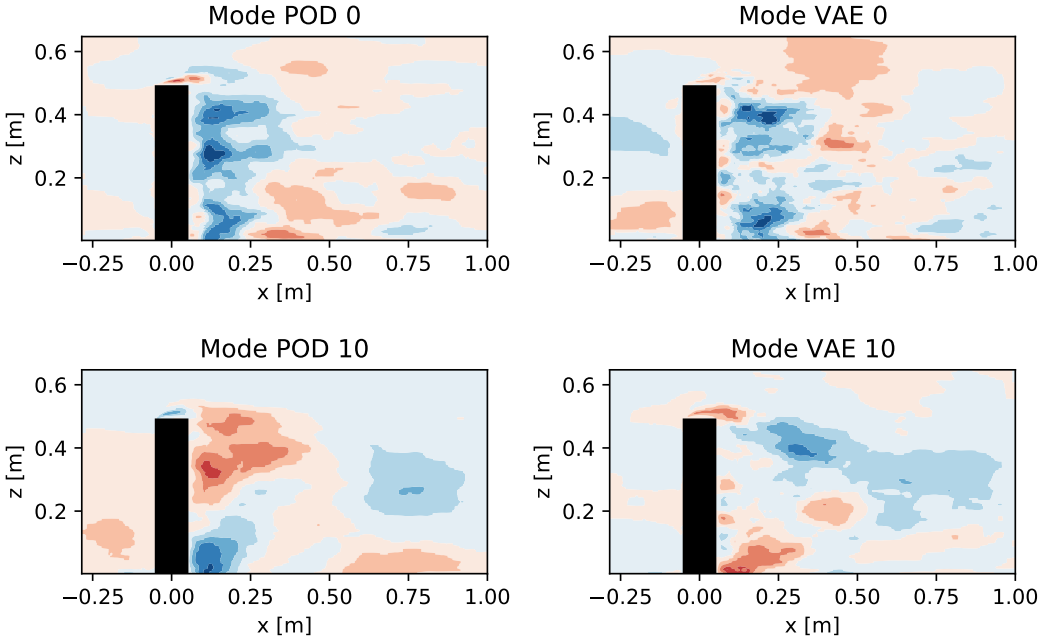
6.1.3 | Quantification de l'erreur d'auto-encodage

La qualité de l'auto-encodage est quantifiée pour toutes les réductions par l'erreur moyenne normalisée. Dans sa version globale, l'erreur est calculée en faisant une moyenne sur toutes les composantes de vitesse et tous les points de maillage. Les résultats sont rassemblés dans les tableaux de l'annexe D. Pour chacune des méthodes, on s'aperçoit que l'erreur de réduction diminue lorsque la dimension de l'espace latent augmente. En ce qui concerne les réductions linéaires, les méthodes ont des performances similaires. Ce n'est pas étonnant quand on sait

8. Une explication possible viendrait de la richesse fréquentielle des écoulements. La couche de mélange a un spectre large bande, résultat d'une instabilité convective. Pour les écoulements de cylindre, le spectre est discret, résultat d'une instabilité absolue. L'auto-encodeur non linéaire variationnel a sans doute du mal à encoder facilement toute l'information spectrale des clichés de la couche de mélange.

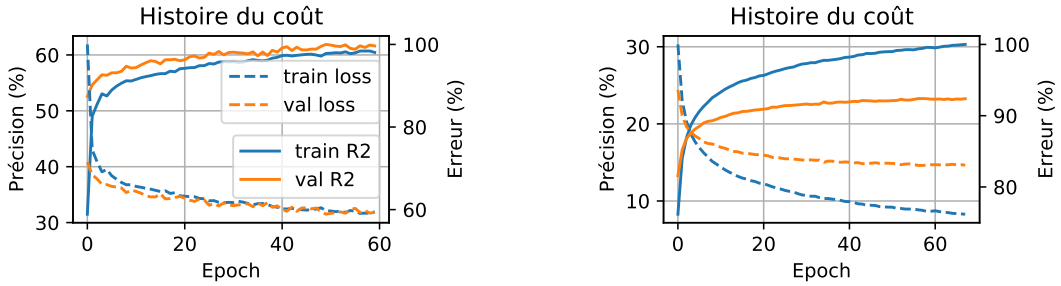


(a) Écoulement turbulent autour du cylindre



(b) Plan vertical de l'écoulement autour de la tour

Figure *6.6* – *Comparaison entre deux modes POD et deux modes dominants VAE. Le premier mode est semblable pour les deux méthodes de réduction, signe que l'auto-encodeur non linéaire respecte bien le critère énergétique. Ce n'est plus le cas pour les modes d'ordre élevé, signe que l'auto-encodeur non linéaire ne se concentre pas uniquement sur ce critère.*



(a) Cylindre 3D **(b)** Plan 21 de la tour

Figure *6.7* – *Histoire de la fonction coût pour la réduction VAE. La précision est le coefficient de détermination moyen. L'erreur est la somme de l'erreur quadratique et de la divergence de Kullback-Leibler, le tout normalisé par la valeur d'entraînement au premier epoch. Du sur-apprentissage est notable pour l'écoulement autour de la tour, avec des performances d'entraînement et de validations différentes.*

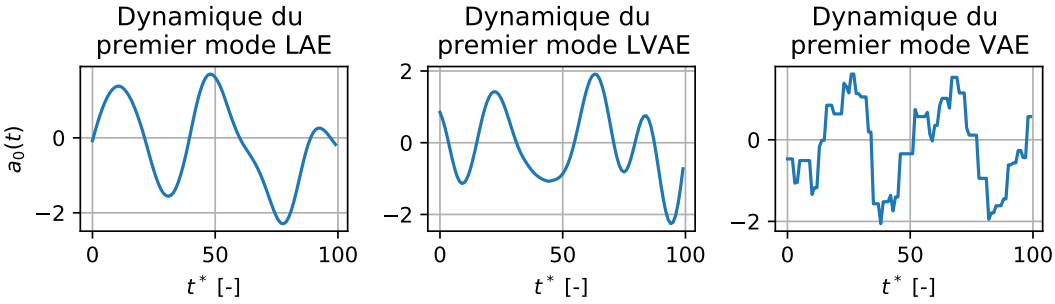
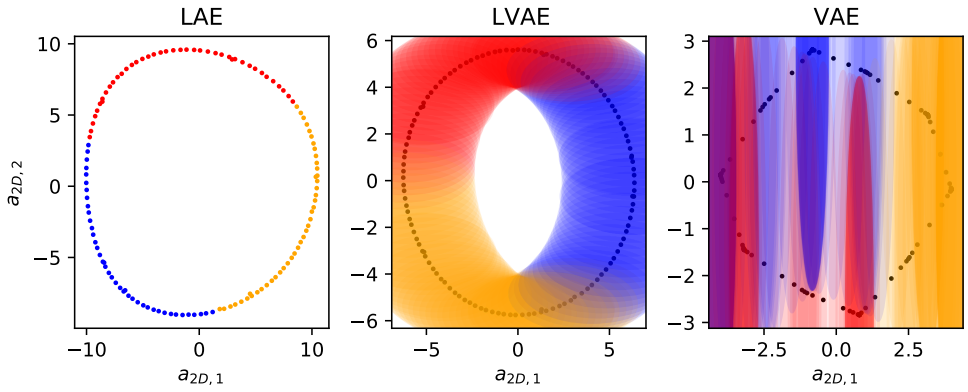
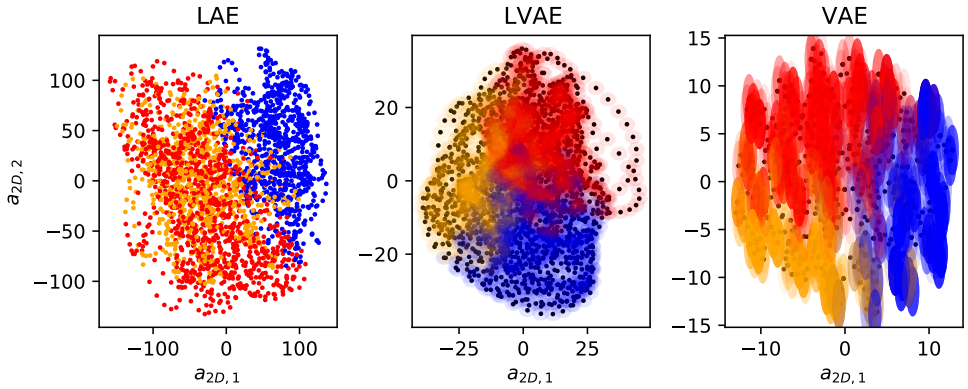


Figure *6.8* – *Comparaison des encodages linéaires et de l'encodage non linéaire sur la couche de mélange. L'état latent déterminé par le VAE fait apparaître des sauts dans sa dynamique.*

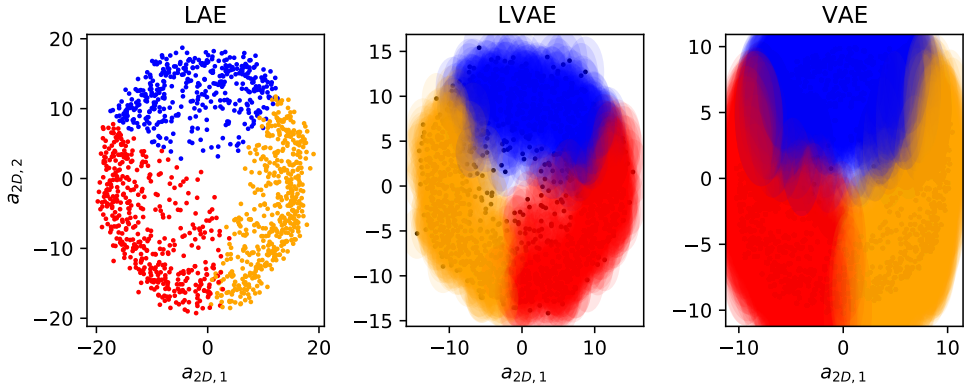
que les modes calculés par ces décompositions décrivent le même sous espace. Les erreurs les plus importantes sont obtenues pour la réduction non linéaire, ce qui peut s'expliquer par le choix non optimisé des hyper-paramètres et le nombre de clichés. On note également que sur les trois premières configurations, les erreurs d'auto-encodage des champs d'entraînements sont similaires aux erreurs d'auto-encodage des champs de test. Cela signifie que les modes déterminés par l'apprentissage se généralisent à de nouveaux clichés, une propriété recherchée pour faire de l'estimation d'écoulement. En revanche, le sur-apprentissage est



(a) Écoulement laminaire autour du cylindre (réduction $r_{0.99} = 5$ modes)

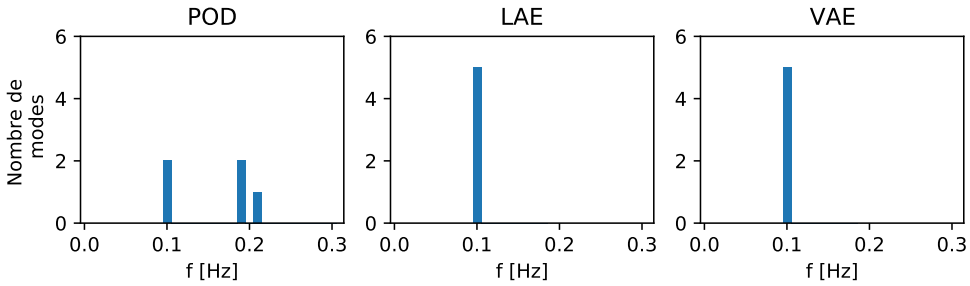


(b) Écoulement de couche de mélange (réduction $r_{0.8} = 11$ modes)

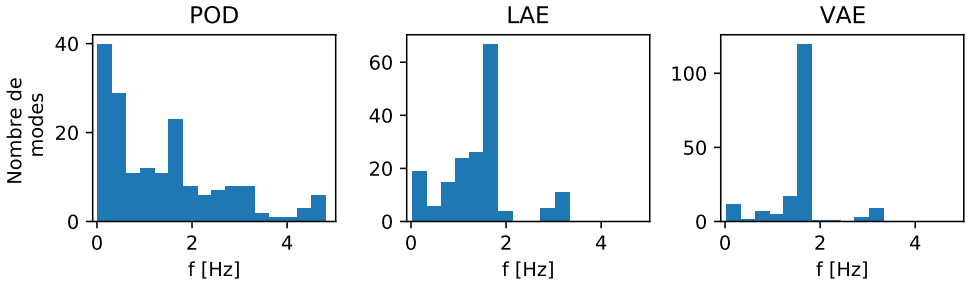


(c) Écoulement turbulent autour du cylindre rectangulaire (réduction $r_{0.95} = 21$ modes)

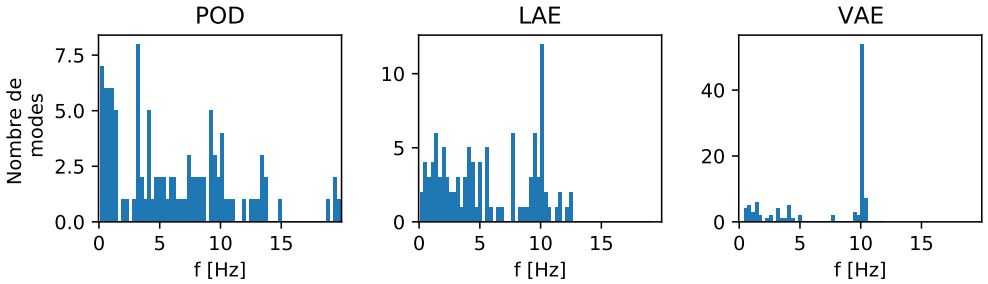
Figure *6.9* – *Visualisation 2D de l'espace latent pour les trois premières configurations. Les états latents moyens sont projetés selon les deux premières directions principales. Pour les réductions variationnelles, les ellipses de confiance à 95% sont également projetées. Les couleurs indiquent l'appartenance à une partition (parmi trois) du champ de vitesse.*



(a) Cylindre 2D et réduction $r_{0.99} = 5$ modes



(b) Cylindre 3D et réduction $r_{0.95} = 177$ modes



(c) Plan 11 de la tour et réduction $r_{0.8} = 106$ modes

Figure *6.10* – *Histogrammes des fréquences dominantes d'oscillation des modes pour les différentes méthodes de réduction. Avec une réduction non orthogonale ou non linéaire, la majeure partie des modes oscille à la fréquence macroscopique de l'écoulement qui fixe le Strouhal.*

important pour chaque méthode dans le cas de la tour, ce qui est symptomatique d'une réduction trop brutale (trop peu de modes) ou non linéaire pas assez profonde (pas assez de couches pour un traitement efficace des données).

Concernant les cartes d'erreurs locales, celles-ci indiquent les zones où les

fluctuations de vitesse sont difficilement (dans un sens moindres carrés) capturées par la réduction. La figure 6.11 montre quelques exemples de cartes :

a) pour le cylindre 2D réduite non linéairement à 5 modes, l'erreur d'auto-encodage est non nulle dans tout le sillage. Ce n'est pas étonnant étant car le critère *moindres carrés* n'est pas la seule contrainte lors de l'apprentissage. Pour les réductions linéaires, l'erreur est presque nulle partout.

b) pour la couche de mélange réduite linéairement à 80% d'énergie, l'erreur est maximale au niveau du point d'origine de la couche de mélange. Avec une réduction à 95%, la capture de ce point est améliorée.

c) pour le cylindre 3D réduit à 95% d'énergie, les réductions linéaires POD et LAE ont des cartes d'erreur similaires, ce qui n'est pas étonnant étant donné le lien fort entre les deux méthodes.

6.1.4 | Spectre du champ auto-encodé

Lors de l'encodage puis du décodage du champ de vitesse, de l'information est perdue. En comparant les valeurs singulières des champs réels et les valeurs singulières des champs auto-encodés, on peut détecter les contributions énergétiques non restituées par la réduction. La figure 6.12 compare les spectres pour les différentes configurations. Ces spectres sont calculés sur les données de test pour voir dans quelle mesure les auto-encodeurs appris se généralisent à de nouvelles données. On s'aperçoit que les méthodes de réduction linéaires agissent comme un filtre passe-bas [9] pour fournir l'approximation de rang faible. Au contraire, la méthode de réduction non linéaire essaye de restituer l'intégralité de l'information spectrale. À noter que pour l'écoulement laminaire autour du cylindre, on peut être déçu des résultats de l'auto-encodeur variationnel. On attribue cette faiblesse au nombre résolument petit de clichés disponibles [10].

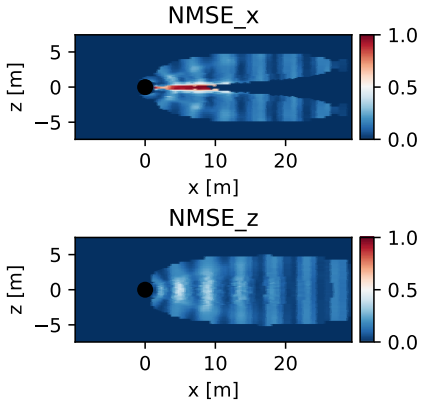
6.1.5 | Robustesse lors du décodage

Les auto-encodeurs variationnels régularisent l'espace latent en encodant les champs de vitesse en distributions plutôt qu'en simple points. Avec les activations non linéaires (VAE), l'erreur d'auto-encodage du champ de vitesse est plus élevée que pour les réductions linéaires. Toutefois, la réduction VAE est plus robuste au bruit appliqué sur l'état latent lors du décodage. Pour s'en rendre compte, on ajoute du bruit gaussien sur les composantes de l'état latent, en suivant :

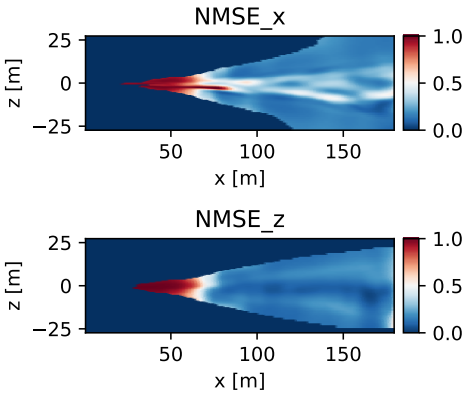
$$a_{[i_r,t]} \leftarrow a_{[i_r,t]} + \epsilon \text{ avec } \epsilon \sim \mathcal{N}\left(0, \left[I_a \max\left[a_{[i_r,:]}\right]\right]^2\right)$$

De cette manière, les contributions de chaque mode sont perturbées d'un pourcentage I_a de la contribution maximale du mode associé. Les états latents sont

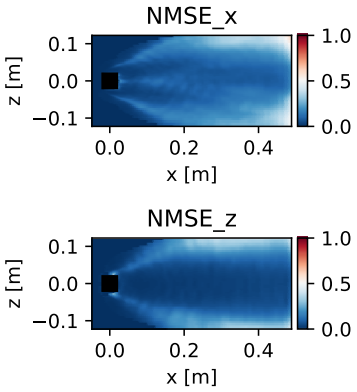
9. Le travail de Arnault *et al.* (2016) étudie plus en profondeur cet aspect
10. Pour l'entraînement de l'auto-encodeur, seulement 88 clichés. Pour le test et le calcul des spectres, seulement 37 clichés.



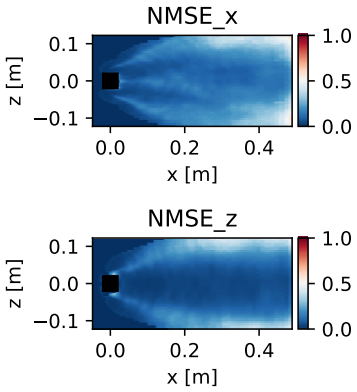
(a) Cylindre 2D - réduction VAE avec cinq modes



(b) Couche de mélange - réduction POD avec 11 modes



(c) Cylindre 3D - réduction POD avec 177 modes



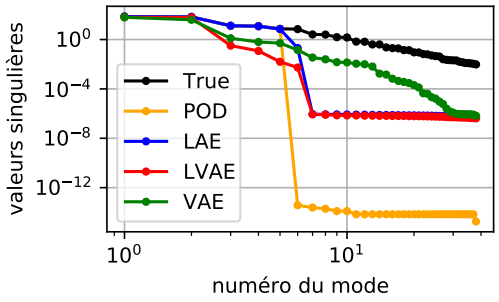
(d) Cylindre 3D - réduction LAE avec 177 modes

Figure *6.11* – *Exemples de cartes d'erreurs de test.*

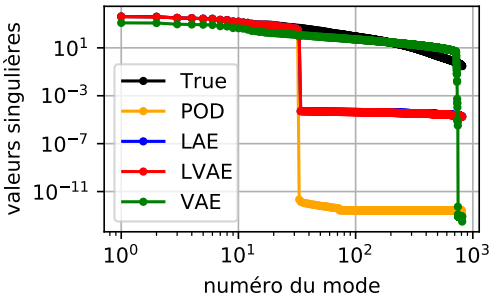
ensuite décodés et on calcule l'erreur moyenne normalisée entre les champs réels et les champ décodés. Formellement, on utilise le rapport :

$$\gamma = \frac{\text{NMSE}_0}{\text{NMSE}_{I_a}}$$

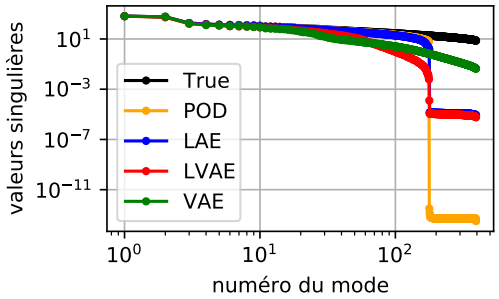
où NMSE_0 est l'erreur lorsque l'état latent n'est pas bruité (erreur de réduction en annexe D) et NMSE_{I_a} est l'erreur lorsque l'état latent est bruité avec une intensité I_a. Lorsque γ est proche de 1, le bruit n'a pas d'influence dans le processus de décodage. Lorsque γ est proche de 0, le décodeur est très sensible aux perturbations sur l'état latent. La figure 6.13 montre les erreurs de test pour différents cas. En l'absence de bruit, l'auto-encodeur variationnel a l'erreur de réduction la plus élevée, ce qui rejoint les conclusions de la section 6.1.3. Mais en augmentant l'intensité



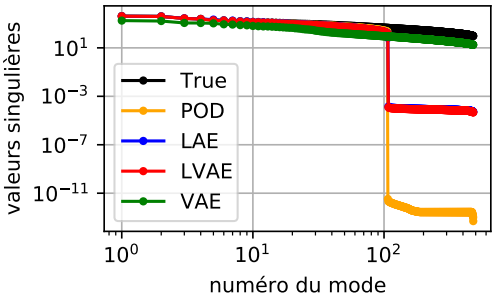
(a) Cylindre 2D avec $r_{0.99}$ modes.



(b) Couche de mélange avec $r_{0.95}$ modes



(c) Cylindre 3D avec $r_{0.95}$ modes



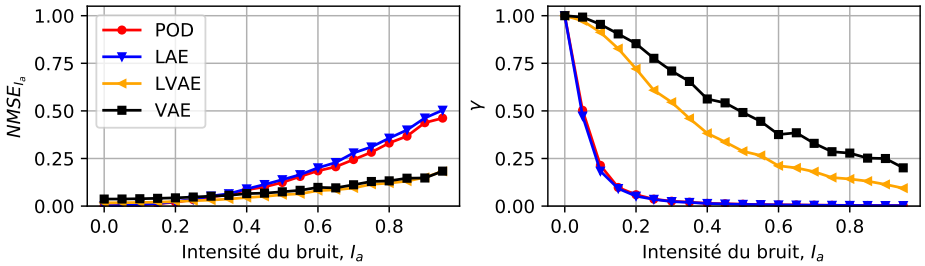
(d) Plan 11 de la tour avec $r_{0.8}$ modes

Figure *6.12* – *Valeurs singulières de la matrice des clichés et valeurs singulières de la matrice des clichés auto-encodés. Les spectres sont calculés sur les données de test et le nombre de valeurs singulières est limité par le nombre de clichés. Les réductions linéaires filtrent l'information peu énergétique tandis que la réduction non linéaire cherche à reproduire tout le spectre.*

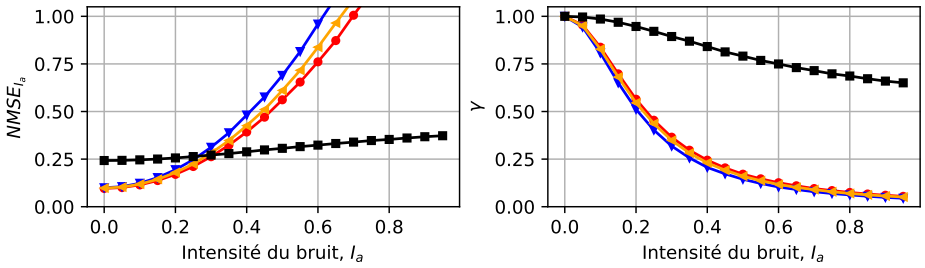
du bruit sur l'état latent, le VAE semble plus robuste au décodage. Ce résultat conforte l'aspect génératif derrière la contrainte variationnelle : l'auto-encodeur apprend à générer des données pour des échantillons de son espace latent non utilisés pendant l'apprentissage. À noter que dans sa version linéaire, le LVAE ne semble pas plus robuste que la POD et le LAE. Les non linéarités semblent donc requises pour avoir un auto-encodeur capable de tâches génératives.

6.1.6 | Conclusion

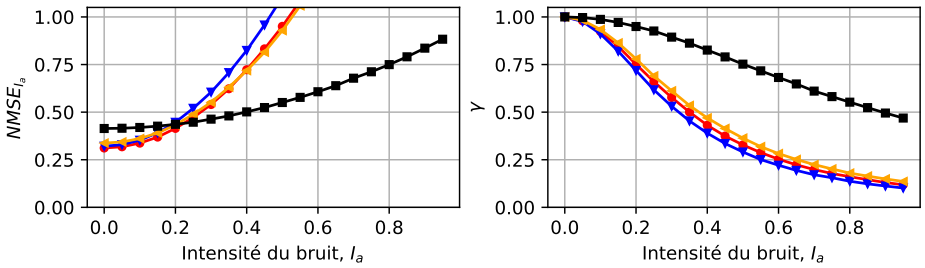
Dans cette section, différentes méthodes de réduction ont permis d'extraire les structures dominantes de quatre écoulements. La première méthode est la



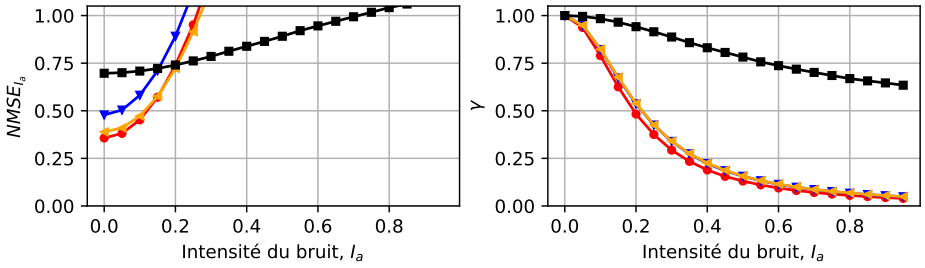
(a) Cylindre laminaire ($r_{0.99}$ modes)



(b) Couche de mélange ($r_{0.8}$ modes)



(c) Cylindre turbulent ($r_{0.8}$ modes)



(d) Plan 11 de la tour ($r_{0.8}$ modes)

Figure *6.13* – *Étude de la robustesse au bruit lors du décodage de l'état latent. Les courbes montrent les erreurs calculées pour les données de test. Figures de gauche : erreur moyenne normalisée d'auto-encodage lorsque l'état latent est perturbé d'un niveau I_a. Figures de droite : ratio entre l'erreur d'auto-encodage sans bruit et l'erreur d'auto-encodage avec bruit. Il apparaît que le VAE est la méthode de réduction la plus robuste lors du décodage.*

décomposition en modes propres orthogonaux (POD) qui classe les structures *orthogonalement* par importance énergétique. La seconde méthode est l'auto-encodeur linéaire (LAE). Les structures dominantes sont déterminées de manière itérative par transformations linéaires mais la contrainte d'orthogonalité n'est pas imposée. Les modes apparaissent plus interprétables que la POD et oscillent majoritairement à la fréquence dominante de l'écoulement. En modifiant la fonction coût pour faire un compromis entre la variance des données à restituer et la régularité de l'espace latent, l'auto-encodeur linéaire devient variationnel (LVAE). Cette troisième méthode détermine des modes qui génèrent le même sous espace que les modes POD et LAE. En considérant des transformations non linéaires, l'auto-encodeur est simplement qualifié de variationnel (VAE). En complétant l'espace latent, cette dernière méthode se montre plus robuste lors du décodage d'états latents perturbés. L'objectif est maintenant d'estimer l'état réduit à partir de mesures pour faire de la *reconstruction*.

6.2 | Reconstruction de l'état latent

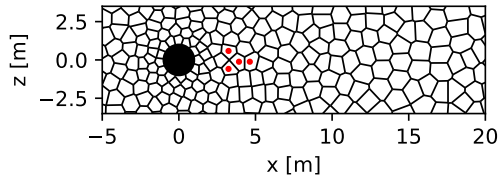
Cette section compare différentes méthodes pour la reconstruction du champ de vitesse des écoulements. Après une discussion sur les grilles de capteurs retenues, on présente les scores de reconstruction en fonction de la méthode de réduction, de la méthode d'estimation et de la densité de capteurs. La validation des hyper-paramètres est illustrée sur quelques cas et la robustesse des méthodes au bruit de mesure est étudiée. On rappelle que les méthodes impliquées sont la reconstruction directe (DR), la régression linéaire mono-tâche (LS) ou multi-tâches (LM), la régression par vecteurs supports (SVR), la régression par réseau de neurones (NN) et la régression par arbres de décisions boostés (GB).

6.2.1 | Le placement des capteurs

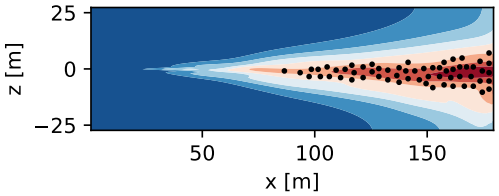
Le placement des capteurs est effectué par apprentissage non-supervisé, selon un partitionnement amélioré. Celui-ci consiste à placer les capteurs au centre des partitions les plus énergétiques du maillage. En fixant le nombre de clusters à $K_c = 500$, les capteurs correspondent aux centroïdes des partitions qui cumulent 20%, 40% ou 80% de la variance des clichés d'entraînement. La figure 6.14 illustre quelques exemples de placement de capteurs tandis que le tableau 6.3 résume le nombre de mesures pour estimer chaque écoulement.

Pour l'écoulement laminaire autour du cylindre, le nombre de capteurs $p_{0.8}$ est élevé. La littérature montre que peu de capteurs sont requis pour avoir de bonnes estimations régressives de cet écoulement (par exemple avec le réseau superficiel d'Erichson et ses cinq capteurs de pression pariétale, voir Erichson *et al.* (2020)). C'est un défaut du placement systématique que l'on a suivi : le choix de 500 partitions n'est pas universel et devrait être optimisé [11]. Un nombre important de capteurs augmente aussi le risque de multi-colinéarités mais cela sera amoindri par

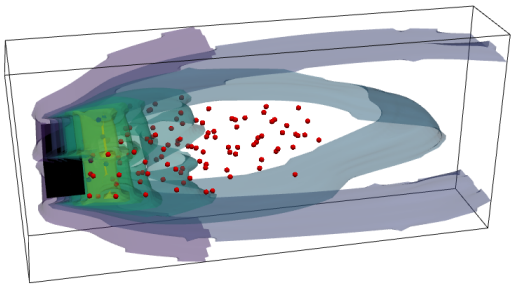
11. Par exemple avec le critère du coude ou de la silhouette (Kodinariya et Makwana, 2013).



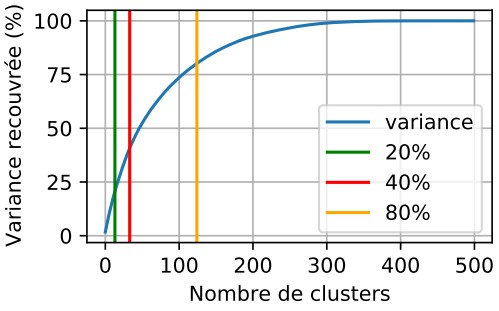
(a) Cylindre laminaire - $p_{0.2}$. Cellules de Voronoï du maillage.



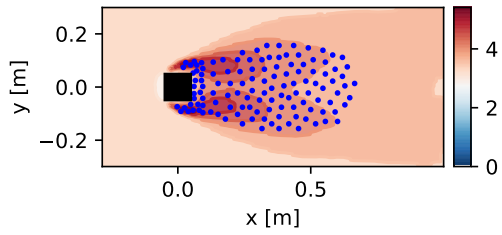
(b) Couche de mélange spatiale - $p_{0.8}$



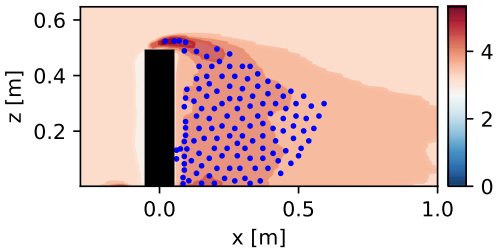
(c) Cylindre turbulent - $p_{0.8}$



(d) Cylindre turbulent - variance en fonction du nombre de partitions



(e) Écoulement urbain - plan 11 - $p_{0.8}$



(f) Écoulement urbain - plan 21 - $p_{0.8}$

Figure *6.14* – *Quelques résultats du placement de capteurs. Sur la figure (d), on représente la variance des clichés d'entraînement restituée en fonction du nombre de partitions. Selon le pourcentage de variance à restituer, on définit le nombre de capteurs. Sur la couche de mélange, le cylindre 3D et la tour, on trace les contours de la RMS de la composante longitudinale de vitesse.*

Configuration	0	1	2	11	12	13	14	21	31	32
$p_{0.2}$	8	18	39				-			
$p_{0.4}$	22	42	99				-			
$p_{0.8}$	92	122	372	375	405	381	249	393	627	549

Tableau 6.3 – *Nombre de capteurs pour les différentes configurations. Pour les trois premiers écoulements (cas 0 pour le cylindre laminaire, 1 pour la couche de mélange et 2 pour le cylindre turbulent), on envisage trois niveaux de densité. Pour l'écoulement urbain et les différents plans, seul un niveau de densité est considéré. Par exemple : pour l'écoulement laminaire autour du cylindre, quatre capteurs mesurant chacun les deux composantes de la vitesse ($p_{0.2} = 4 \times 2$) sont utilisés pour estimer l'état latent.*

la validation croisée des hyper-paramètres [12]. De toute manière, si un modèle venait à être déployé dans une configuration réelle, il faudrait fortement limiter le nombre de capteurs pour des raisons pratiques.

6.2.2 | Erreurs de reconstruction et quelques exemples

Les tableaux de l'annexe E quantifient la qualité des reconstructions des écoulements considérés. On y trouve les erreurs d'estimation du champ de vitesse (NMSE) et les scores d'estimation des états latents (R^2) en fonction de la méthode de réduction et de la méthode de reconstruction (directe ou régressive). Ces résultats sont peu interprétables lorsqu'ils sont pris isolément ; les valeurs numériques servent plutôt de référence pour guider le choix d'une méthode dans des applications futures. Deux tendances se dégagent toutefois.

Premièrement, augmenter la densité de capteurs améliore l'estimation. Sur la figure 6.15, on illustre la reconstruction dans l'espace physique d'un cliché de test et la reconstruction dans l'espace latent d'un coefficient POD. Dans les deux cas, l'ajout d'informations permet une estimation plus fidèle de l'état attendu. Dans l'annexe E, la reconstruction dans l'espace latent est quantifiée par le coefficient de détermination moyen. Celui-ci synthétise la qualité de reconstruction de *tous* les coefficients temporels pour *toutes* les données d'entraînement ou de test. De manière générale, les scores de reconstruction sont *bons* (R^2 entre 0.6 et 1) et les états latents sont bien estimés par les mesures. C'est surtout vrai pour les reconstructions régressives : quelques reconstructions directes, notamment pour les réductions linéaires variationnelles et les problèmes sous-déterminés, ont un score de zéro [13]. Concernant l'estimation du champ de vitesse, le passage vers l'espace

12. Notamment LASSO qui sélectionne les variables pour les régressions linéaires
13. Plusieurs explications possibles. Déjà, la grille de capteurs a été choisie par un procédé

physique introduit de l'erreur car l'ensemble des modes ne décrit pas toute la variabilité de l'écoulement. On retrouve cet impact avec les erreurs moyenne normalisées (annexe E) qui font la synthèse sur toutes les composantes de vitesse et tous les points de maillage. La réduction VAE ayant l'erreur d'auto-encodage la plus élevée, c'est naturellement celle qui a les plus grandes erreurs de reconstruction.

Deuxièmement, les scores se détériorent avec la complexité de l'écoulement. Les reconstructions sont excellentes sur l'écoulement laminaire autour du cylindre, ce qui est rassurant étant donné la nature simple et périodique de l'écoulement. Pour la couche de mélange et l'écoulement turbulent autour du cylindre, les estimations ont une qualité raisonnable et le sur-apprentissage est limité : les erreurs de test sont semblables aux erreurs d'entraînement. Concernant l'écoulement autour de la tour, les résultats sont plus mitigés : les erreurs normalisées sont plutôt élevées (de l'ordre de 60% à 70%) et le sur-apprentissage est important (différence d'au moins 10% entre l'erreur d'entraînement et l'erreur de test). Il est particulièrement marqué avec la reconstruction par vecteurs supports, ce que l'on illustre dans l'article de conférence en annexe H. Le sur-apprentissage est en revanche limité pour la reconstruction linéaire multi-tâches. On illustre ce résultat avec la figure 6.16 où les scores de la régression LM pour toutes les méthodes de réduction sont tracés. La hiérarchie imposée par la POD est clairement visible dans la courbe de score, avec des grosses structures bien estimées mais des petites structures plus difficiles à reconstruire. Pour les réductions par réseau de neurones, chaque mode est reconstruit avec une performance similaire, signe que les auto-encodeurs neuronaux ne favorisent pas de directions spatiales.

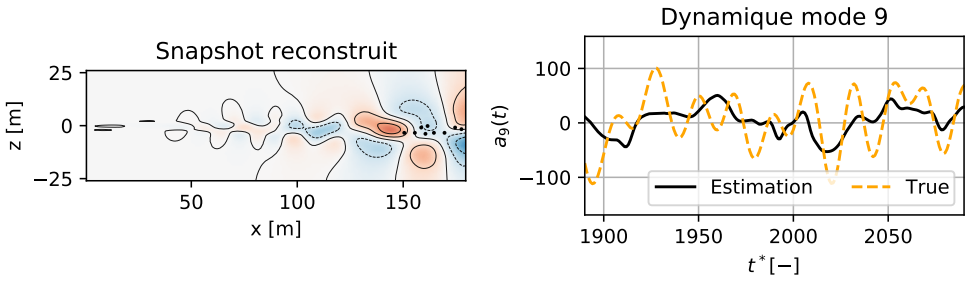
Si les scores d'entraînement et de test sont similaires pour chaque méthode de reconstruction et chaque méthode de réduction, c'est parce que les hyper-paramètres des méthodes ont été systématiquement validés. Dans la suite, on montre quelques exemples de cette validation.

6.2.3 | Exemples de validation des hyper paramètres

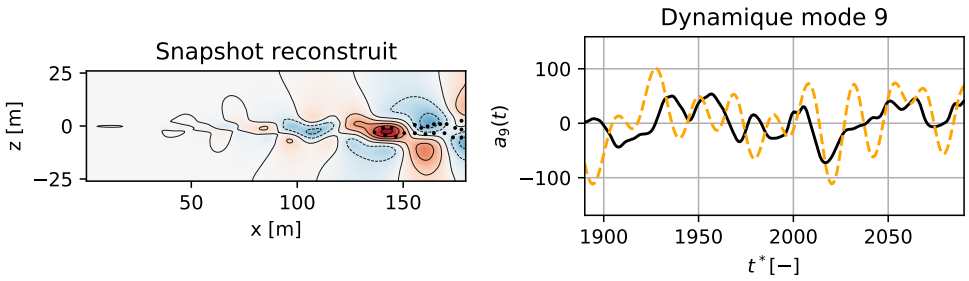
La régression linéaire est la méthode d'estimation la plus simple (un produit matriciel). L'erreur quadratique moyenne est pénalisée par un coût l_1 (cas mono-tâche) ou un coût l_{21} (cas multi-tâches) pour régulariser l'apprentissage. La contribution α de cette pénalité est optimisée par validation croisée [14], les candidats étant sélectionnés dans une grille. La figure 6.17 montre les résultats sur

systématique mais on sait qu'une grille aléatoire est préférable pour l'acquisition comprimée. Ensuite, le problème sous-déterminé promeut la parcimonie dans la solution. Si l'état latent déterminé est le vecteur nul pour chaque estimation, les scores seront nuls.

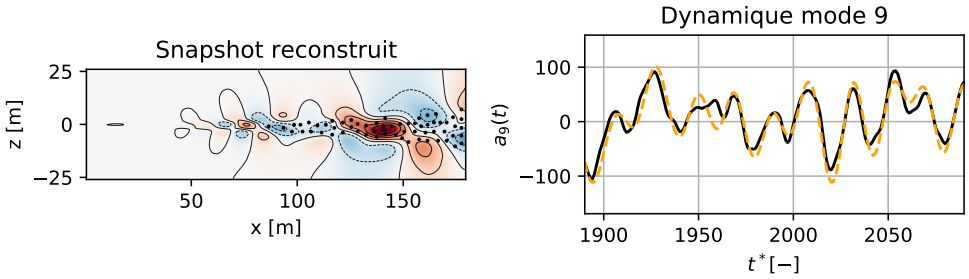
14. Pour la reconstruction, on rappelle que l'erreur de généralisation est calculée par blocs. Les données d'apprentissage sont divisées en trois blocs, deux étant réservé à l'apprentissage et un étant réservé à la validation. Pour chaque hyper-paramètre testé, trois modèles sont appris car trois découpage sont possibles. On choisit l'hyper-paramètre à l'erreur de validation minimale ou au score de validation maximal. Puis, on apprend le modèle final sur toutes les données d'entraînement.



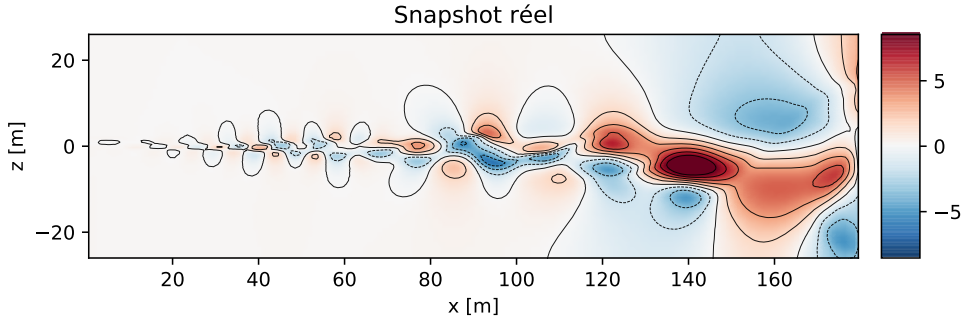
(a) Grille de capteurs $p_{0.2} = 18$



(b) Grille de capteurs $p_{0.4} = 42$



(c) Grille de capteurs $p_{0.8} = 122$



(d) Cliché de test de référence

Figure *6.15* – *Résultats de reconstructions pour la couche de mélange. On visualise l'estimation dans l'espace physique (un cliché de test) et l'estimation d'une composante de l'état latent POD (plusieurs états de test). En augmentant la densité de capteurs, les reconstructions sont plus fidèles aux donnés de référence. Le cliché est coloré par la composante longitudinale de la vitesse fluctuante.*

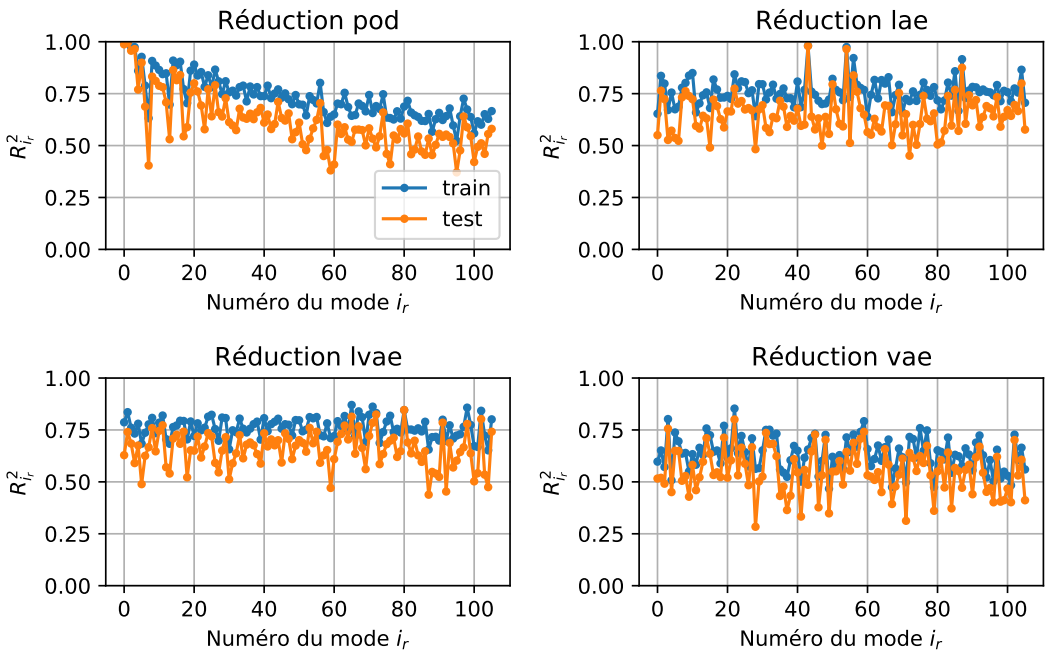
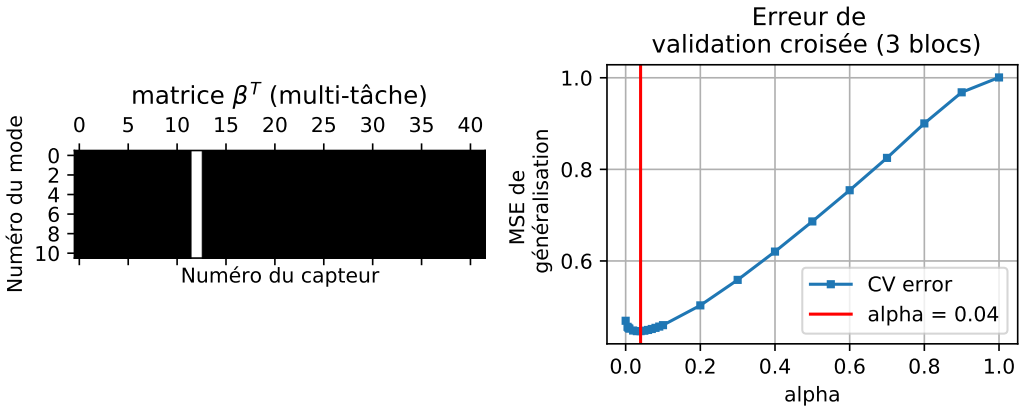


Figure *6.16* – *Scores de reconstruction pour l'écoulement urbain (plan 11) avec une régression linéaire multi-tâches et* $p_{0.8} = 375$ *capteurs. La hiérarchie imposée par la réduction orthogonale se retrouve dans les scores de reconstruction. Pour les réductions neuronales, chaque composante de l'état latent est reconstruit avec une performance similaire.*

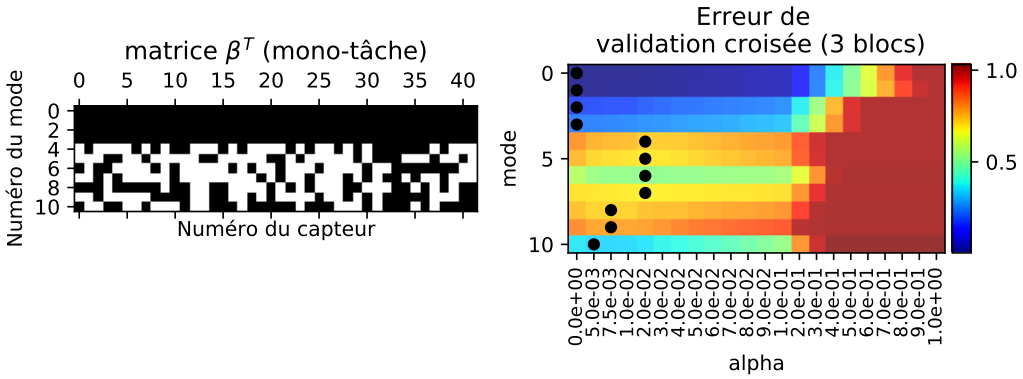
la couche de mélange. Dans le cas multi-tâche, la norme l_{21} a pour effet de sélectionner les mêmes capteurs pour *tous* les modes, ce qui peut présenter un avantage pratique [15]. Dans le cas mono-tâche, la sélection est spécifique à chaque mode. À noter qu'une parcimonie trop importante des variables explicatives reviendrait à toujours estimer par des fluctuations nulles. L'erreur d'estimation serait alors importante. On retrouve ce compromis dans la matrice des erreurs : augmenter la pénalité augmente l'erreur de généralisation.

La régression par vecteurs supports est une régression linéaire dans un espace de dimension plus élevé que l'espace latent. La régularisation est contrôlée par la taille de la boite C_B (limite des multiplicateurs de Lagrange), le noyau N (polynomial ou gaussien à base radiale) et ses paramètres (degré ou paramètre γ). Les candidats sont échantillonnés aléatoirement. La figure 6.18 montre un exemple de candidats et de scores pour la couche de mélange. La formulation de la SVR étant mono-tâche, une régression par mode est requise. Pour la régression par

15. Le capteur peut complètement être neutralisé car il n'est utile pour aucun des modes.



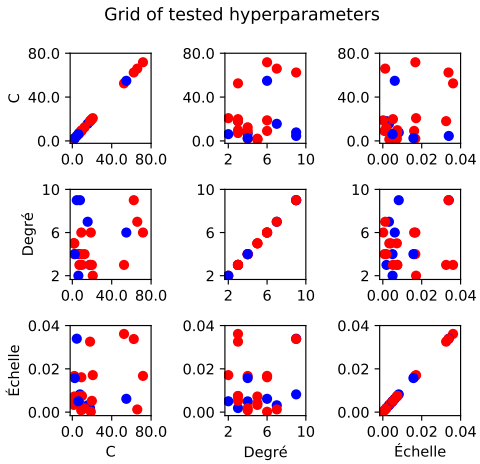
(a) Cas multi-tâches. Le paramètre de sélection est unique pour tous les modes. À gauche : matrice des paramètres de la régression, un seul capteur à neutraliser (en blanc). À droite : courbe d'erreur de validation et paramètre LASSO.



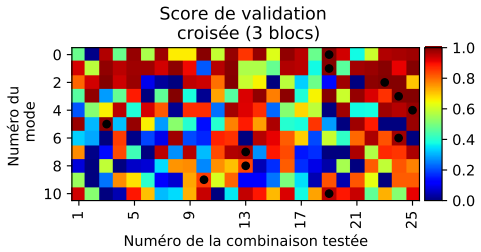
(b) Cas mono-tâche. Le paramètre de sélection est différent pour chaque mode. À gauche : matrice des paramètres de la régression. Les capteurs à garder (en noir) sont différents pour chaque mode ; pour les trois premiers modes, tous les capteurs sont à conserver car aucun ne nuit à la robustesse de la reconstruction. À droite : matrice des erreurs de validations. Les points noirs correspondent aux erreurs de généralisation minimales, donc aux paramètres LASSO.

Figure *6.17* – *Optimisation du paramètre LASSO. Illustré sur la couche de mélange $r_{0.8}$ modes (POD) et $p_{0.4}$*

arbres de décisions boostés, le même type de résultats est obtenu. Seuls les hyper-paramètres changent : il s'agit du nombre d'arbres, de la profondeur maximale, de la vitesse d'apprentissage et du nombre de feuilles terminales. Contrairement à la régression linéaire, la recherche aléatoire des hyper-paramètres rend plus difficile la lecture de la matrice des scores et trop peu de candidats (ici 25) limite les chances de trouver une combinaison optimale.



(a) Hyper-paramètres testés. Un point rouge correspond à un noyau polynomial tandis qu'un point bleu correspond à un noyau gaussien.

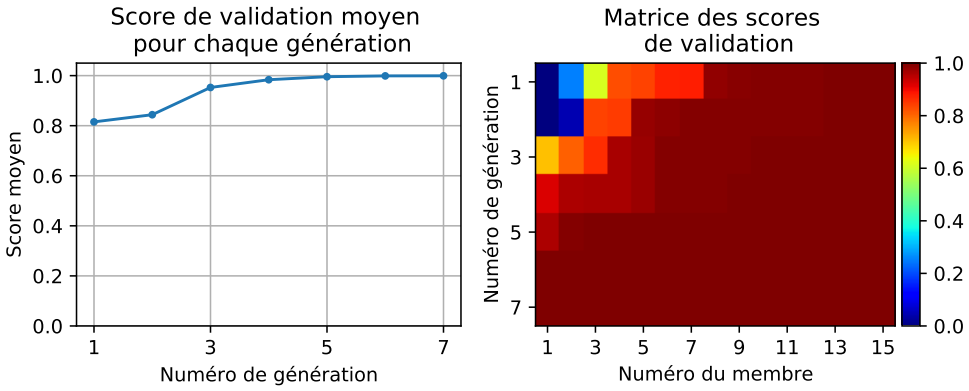


(b) Matrice des scores de validation. Plusieurs hyper-paramètres sont à optimiser et la sélection des candidats n'est pas ordonnée, ce qui se retrouve dans la matrice.

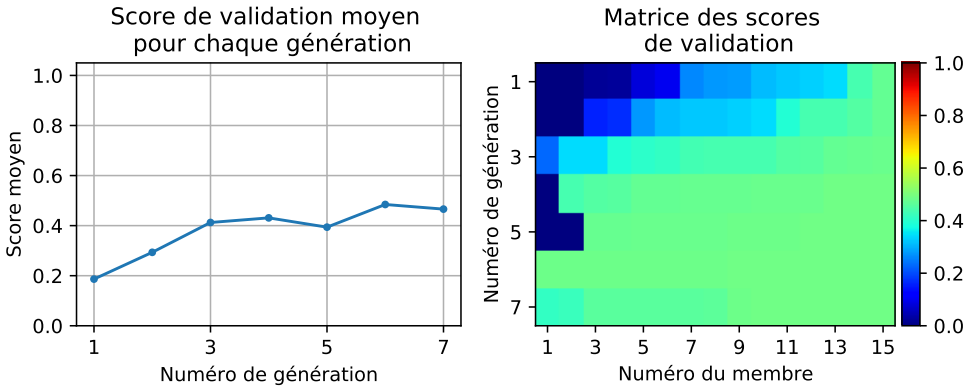
Figure *6.18* – *Optimisation des hyper-paramètres de la régression par vecteurs supports. Illustré sur la couche de mélange avec $r_{0.8}$ modes (POD) et $p_{0.4}$ mesures. Même type de résultats pour la régression par arbres de décisions boostés.*

Concernant la reconstruction par réseau de neurones, les hyper-paramètres à optimiser sont le nombre de couches, le nombre de neurones par couche, les fonctions d'activation et le taux d'oubli. Une population de quinze réseaux est évoluée sur sept générations avec un algorithme génétique. La figure 6.19 montre un exemple d'évolution pour l'écoulement laminaire autour du cylindre et un plan de la tour. Sur le cas du cylindre, l'algorithme génétique aboutit à une population finale avec de bons candidats (score moyen proche de 100%). Sur l'exemple de la tour, les candidats finaux ont un score assez faible (proche de 40%), ce qui est symptomatique d'une grille d'hyper-paramètres pas assez riche. En plus d'une grille initiale plus riche, il faudrait envisager des stratégies d'évolution plus élaborées (comme l'évolution avec adaptation de la matrice de covariance (Igel *et al.*, 2007)).

La validation des hyper-paramètres est une étape *primordiale* dans la construction des fonctions d'estimation. La régression linéaire multi-tâche est la plus simple à optimiser car il n'y a qu'un seul hyper-paramètre à valider pour tous les modes. Sur les autres méthodes régressives, la validation est plus élaborée et demande un effort d'implémentation important. Dans tous les cas, cette validation systématique permet des scores d'entraînement et de test semblables pour les trois premiers écoulements. Sur l'écoulement de la tour, le compromis obtenu par validation croisée est mitigé. Pour avoir de meilleurs résultats, il faudrait disposer



(a) Optimisation génétique pour l'écoulement laminaire autour du cylindre (réduction POD $r_{0.8}$ et $p_{0.8}$ capteurs)



(b) Optimisation génétique pour le plan 11 de la tour (réduction POD)

Figure *6.19* – *Optimisation des hyper-paramètres de la régression par réseau de neurones. Pour l'écoulement laminaire autour du cylindre, la population finale est composé de réseaux dont l'architecture permet d'avoir un bon score de validation. Pour le cas de la tour, le score est plus mitigé.*

de données plus importantes, mieux choisir la grille de capteurs et bien calibrer les exigences [16].

6.2.4 | Robustesse au bruit de mesures

Dans la première section de ce chapitre, on s'est intéressé à la robustesse des décodeurs lorsque du bruit était appliqué sur l'état latent. Dans une application

16. A-t'on besoin d'estimer le champ de vitesse sur un maillage fin ou peut-il être plus lâche ? A-t'on besoin de restituer l'intégralité de l'énergie ? Typiquement, pour un drone en milieu urbain, les petites échelles énergétiques seront filtrées, ce qui calibrera les besoins du modèle.

pratique, un état latent perturbé correspond à une mauvaise estimation par la mesure. On s'intéresse donc à la robustesse des estimations lorsque les mesures sont imparfaites. Pour ce faire, du bruit est ajouté selon :

$$y_{[i_p,t]} \leftarrow y_{[i_p,t]} + \epsilon \text{ avec } \epsilon \sim \mathcal{N}\left(0, \left[I_y \max\left[y_{[i_p,:]}\right]\right]^2\right)$$

De cette manière, chaque mesure est perturbée d'un pourcentage I_y de la contribution maximale possible. Les états latents sont ensuite estimés et on calcule le score moyen [17]. La figure 6.20 montre l'évolution du score en fonction de I_y pour les cas 1 et 2. Pour chaque combinaison de réduction/reconstruction, les scores calculés sont similaires : l'ajout du bruit détériore l'estimation mais la perte de qualité est globalement identique [18]. Ce résultat est attribué à la validation systématique des hyper-paramètres qui pousse *toutes* les méthodes régressives dans leur derniers retranchements. Pour l'écoulement plus complexe autour de la tour (réduction POD), la figure 6.21 illustre l'impact du bruit sur l'estimation de chaque composante de l'état de latent. Les modes rapides sont très affectés par le bruit mais les premiers modes sont bien estimés. Sur une fenêtre d'estimation, la dynamique de l'état reconstruit est pertinente et seule l'amplitude de la contribution est affectée.

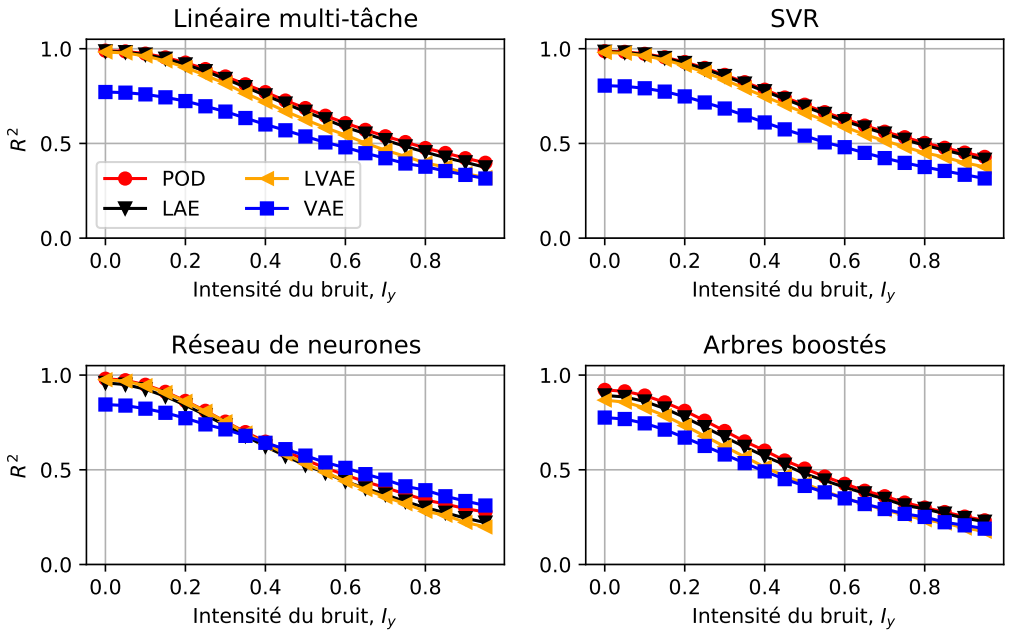
6.2.5 | Estimation des incertitudes de reconstruction

Dans la partie méthodologique du manuscrit, on a proposé deux idées pour estimer les incertitudes de reconstruction. On les illustre ici sur la couche de mélange reconstruite par réseau de neurones. La première méthode utilise des processus gaussiens pour expliquer le résidu d'estimation en fonction de l'état reconstruit. Les paramètres de la matrice de covariance sont optimisés en maximisant leur vraisemblance vis à vis des couples (reconstructions, résidus) d'entraînement. La seconde méthode dite par *dropout* n'est applicable qu'aux réseaux de neurones, bien qu'on pourrait l'étendre aux autres méthodes régressives en généralisant le concept d'oubli. L'idée consiste à activer les couches d'oubli pendant la phase d'estimation et faire un ensemble de reconstructions. Le taux d'oubli est optimisé selon un critère heuristique : l'intervalle de confiance à $c_i\%$ doit contenir $c_i\%$ de bonnes estimations. La figure 6.22 illustre les résultats obtenus (sur des données de test) avec un intervalle de confiance à 95%. Il apparaît que l'intervalle gaussien est plus adapté pour les modes d'ordre élevés (par exemple le dixième) tandis que l'intervalle par oubli contient plus d'états réels pour les modes lents. D'un point de vue complexité, la méthode par dropout est simple à mettre en œuvre car elle se base sur le modèle déjà appris et est multi-tâches. Pour le processus gaussien, une régression est requise par mode [19]. Sans s'attarder sur les résultats, on est convaincu

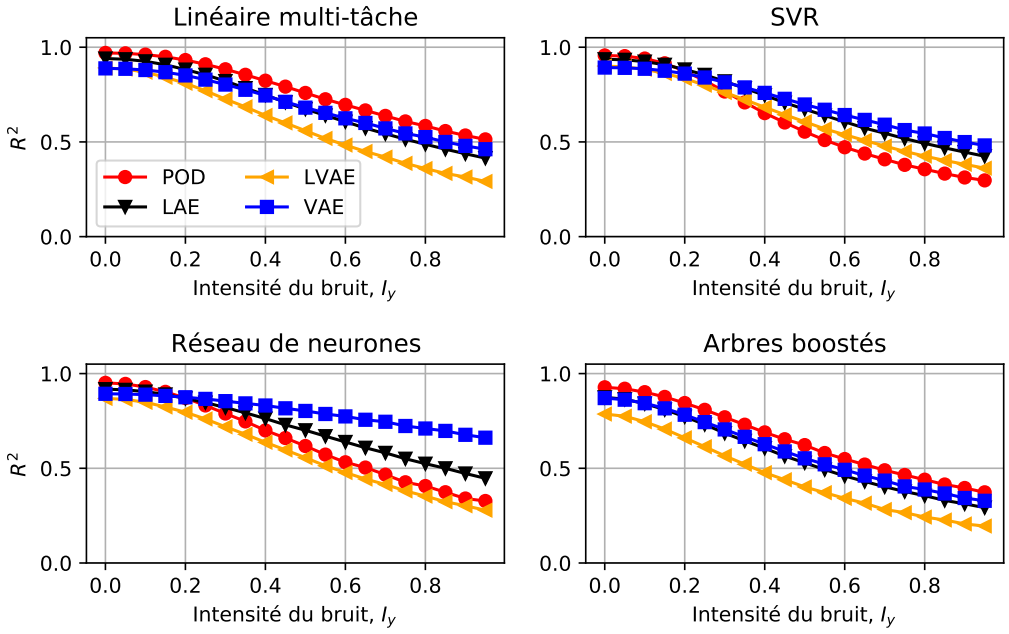
17. Un ensemble de quinze estimations est effectué. Puis on fait la moyenne sur ces quinze estimations, tous les pas de temps d'entraînement ou de test et toutes les composantes de l'état latent

18. À quelques exceptions près. Pour l'écoulement autour du cylindre rectangulaire, la reconstruction par réseau de neurones est plus robuste si la réduction est effectuée par VAE. Pour la couche de mélange, la réduction VAE est la plus difficile à estimer.

19. On pourrait complexifier la méthode avec des processus gaussien multi-tâches. Mais il faudrait définir des matrices de co-régionalisation, ce qui sort clairement du cadre de ce manuscrit.



(a) Résultats pour la couche de mélange ($r_{0.8}$ et $p_{0.8}$)



(b) Résultats pour le cylindre rectangulaire ($r_{0.8}$ et $p_{0.8}$)

Figure *6.20* – *Impact de mesures bruitées sur l'estimation de l'état latent. Les scores sont calculés sur les donnés de test. De manière générale, la détérioration de l'estimation est similaire pour toutes les méthodes de reconstruction, quelque soit la méthode de réduction. On attribue cette réussite à la validation croisée des hyper-paramètres.*

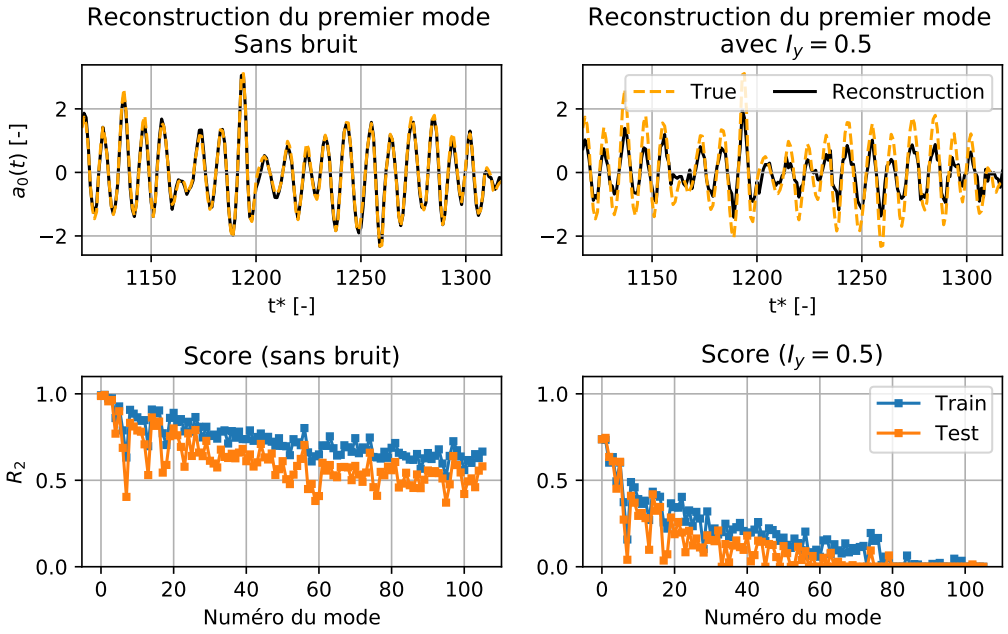


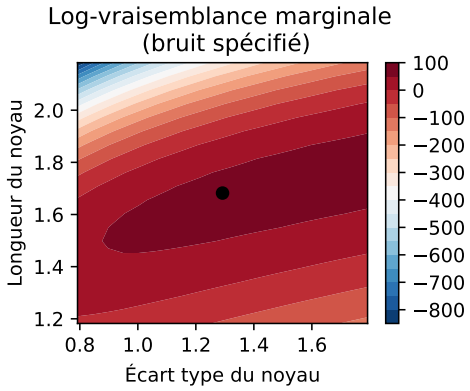
Figure *6.21* – *Les mesures de l'écoulement urbain (plan 11) sont perturbés d'un niveau $I_y = 0.5$. L'estimation des coefficients POD est faite par régression linéaire multi-tâches. Le bruit affecte principalement les modes d'ordre élevé.*

que ces méthodes d'estimation d'incertitudes méritent d'être citées.

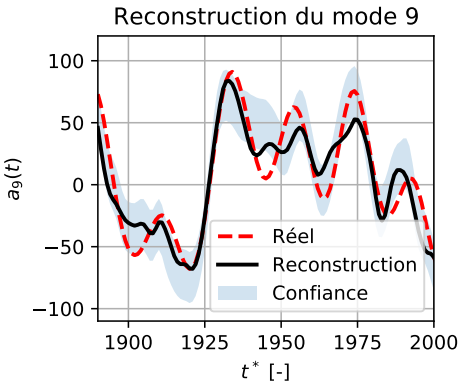
6.2.6 | Conclusion

Dans cette section, différentes méthodes d'estimation de l'état latent ont été comparées. De manière générale, les scores obtenus par estimation régressive ou directe sont bons et la majeure partie de l'erreur dans l'espace physique provient de la méthode d'auto-encodage. Le sur-apprentissage est limité pour les trois premiers écoulements mais il est important pour l'écoulement autour de la tour. Pour une configuration si complexe, il faut revoir les exigences à la baisse et s'autoriser de ne pas reconstruire tout le champ de vitesse. Concernant le placement de capteurs, celui-ci a l'avantage d'être systématique. En revanche, il faudrait optimiser le nombre de partitions et ajouter une contrainte pour limiter les multi-colinéarités. Le résultat le plus intéressant concerne la robustesse des méthodes au bruit de mesures. Grâce à la validation croisée, chaque méthode régressive présente une robustesse similaire. Le choix d'une méthode de reconstruction ne dépend donc pas de ce critère. Avec l'expérience acquise, on recense quatre conseils. Le choix d'une méthode dépend :

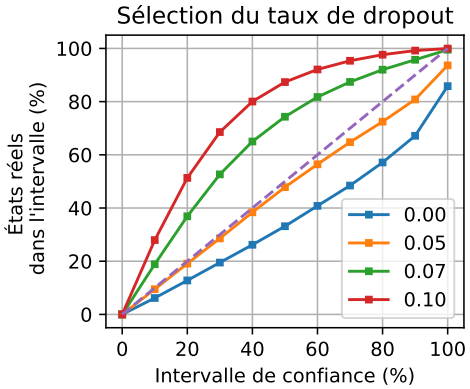
1. **De la qualité des données**. Ici, il s'agit d'avoir des exemples (mesures,



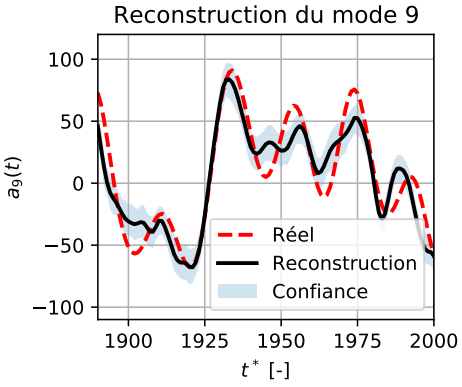
(a) Logarithme de la vraisemblance. Le point noir indique les paramètres optimaux pour construire la covariance du processus gaussien.



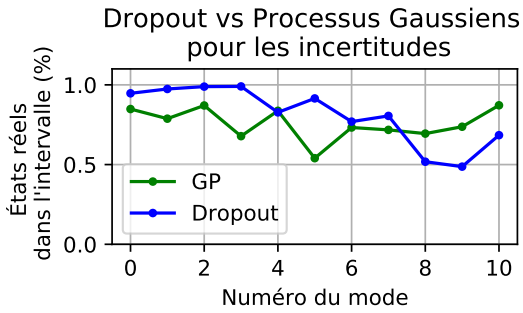
(b) Reconstruction du 10ème mode POD par réseau de neurones et intervalle de confiance (95%) par processus gaussien.



(c) Optimisation du taux d'oubli. On choisit celui qui contient c_i% des états réels pour une confiance de c_i%



(d) Reconstruction du 10ème mode POD par réseau de neurones et intervalle de confiance (95%) par dropout.



(e) Pourcentage d'états réels à l'intérieur des intervalles de confiance.

Figure *6.22* – *Estimation des incertitudes de reconstruction par processus gaussien ou par dropout. Illustré sur la couche de mélange réduite avec la POD,* $r_{0.8}$ *et* $p_{0.8}$.

états) qui soient représentatifs de l'écoulement. Le nombre d'échantillons peut jouer un rôle crucial dans l'apprentissage. Les réseaux de neurones sont connus pour nécessiter de très larges bases de données alors qu'une régression linéaire peut donner de bonnes estimations avec peu d'échantillons.

2. **De l'interprétabilité**. Les arbres de décisions sont très interprétables parce qu'on peut lire le chemin des décisions pour parvenir à l'estimation. Au contraire, les réseaux de neurones profonds sont des boîtes noires où la sortie est obtenue par un propagation complexe de l'entrée. Pour les régressions linéaires, la régularisation par LASSO est un pas vers l'interprétabilité, en sélectionnant de manière parcimonieuse les entrées les plus pertinentes.
3. **Du coût d'implémentation**. Les méthodes ne sont pas égales sur le plan mathématique. La régression linéaire multi-tâches est très simple à implémenter. Au contraire, les arbres de décision boostés et les réseaux de neurones nécessitent de bonnes compétences. Heureusement, avec l'essor des bibliothèques *open source*, le coût d'implémentation est moindre mais il ne faut pas tomber dans le piège du presse bouton.
4. **Du coût de calcul**. Chaque méthode n'est pas égale en termes de coût de calcul. Pour illustrer, le tableau 6.4 donne les temps CPU pour la reconstruction de l'écoulement turbulent autour du cylindre, avec $r_{0.8}$ modes et $p_{0.8}$ capteurs. Les phases d'optimisation des hyper-paramètres sont généralement les plus longues. Au total, pour obtenir les tableaux de reconstruction en annexe E, ce sont 323 heures de calcul CPU qui ont été nécessaires.

Méthode	À noter	Temps CPU
POD	Calcul de la SVD	1 minute
LAE **LVAE** **VAE**	Calcul itératif Pas d'optimisation des hyper-paramètres	3 minutes 6 minutes 6 minutes
Reconstruction directe	Reconstructions de toutes les données Une optimisation par estimation	[5, 5, 5, -] minutes
Réseau de neurones	Optimisation génétique	[15, 17, 30, 21] minutes
Linéaire mono-tâches	Une régression par modes LASSO avec norme l_1	[3, 3, 3, 3] minutes
Linéaire multi-tâches	Une régression pour tous les modes LASSO avec norme l_{21}	[43, 39, 36, 23] minutes
Vecteurs supports **Arbres boostés**	Recherche aléatoire Une optimisation par mode	[6, 6, 6, 6] minutes [26, 25, 26, 24] minutes

Tableau 6.4 *– Temps de calcul pour effectuer les réductions $r_{0.8}$ et les reconstructions $p_{0.8}$ pour l'écoulement turbulent autour du cylindre. Les résultats sont présentés sous la forme [POD, LAE, LVAE, VAE].*

Sur un nouvel écoulement, le conseil serait de commencer par une réduction

POD. Si l'écoulement est simple, la réduction par machines à vecteurs supports est un bon choix : elle permet des régressions non linéaires *simples* par l'astuce du noyau. Sur des écoulements plus complexes, la méthode risque de sur-apprendre comme ce fut le cas pour l'écoulement autour du bâtiment. On se dirige alors vers la régression linéaire multi-tâche. Le réseau de neurones est à envisager ensuite si la base de données est extrêmement riche [20] et que l'interprétabilité n'est pas importante.

Finalement, on peut estimer l'état latent à partir de mesures du champ de vitesse. On s'intéresse maintenant à la prédiction du futur de l'état latent et à l'apprentissage d'un modèle dynamique.

6.3 | Apprentissage d'un modèle dynamique

Cette section s'intéresse à l'apprentissage d'un modèle dynamique pour l'état latent. Différentes méthodes d'approximation finie de l'opérateur de Koopman sont illustrées et comparées selon l'ordre d'introduction dans le chapitre 4.

6.3.1 | Avant de commencer

Les méthodes d'approximation sont appliquées sur les trois premiers écoulements. Pour le sillage laminaire du cylindre, l'état latent est de dimension $r_{0.99} = 5$. Pour la couche de mélange, la dimension est $r_{0.8} = 11$. Pour le sillage turbulent, la dimension est $r_{0.8} = 21$. Contrairement à la reconstruction, on ne s'intéresse pas au passage dans l'espace physique et les erreurs calculées ne concernent que la prédiction de l'état latent ou de son observation. Les méthodes ne sont pas implémentées pour l'écoulement urbain [21]. Les figures de l'annexe F donnent les résultats quantitatifs pour toutes les approximations étudiées. Dans la suite, on propose une lecture de ces résultats en discutant spécifiquement de chaque méthode. Sans mention particulière, l'erreur considérée est l'erreur de prédiction [22] des **états** de test, avec un horizon maximal de $H = 16$ pas de temps sans dimension.

6.3.2 | Décomposition Modale Dynamique

Dans la décomposition modale dynamique (DMD), les observables sont les composantes de l'état latent. L'opérateur de Koopman est approximé par le meilleur modèle dynamique linéaire (au sens des moindres carrés) des clichés de vitesse fluctuante réduits.

20. Ce qui n'était pas vraiment le cas ici...

21. Les résultats sur la réduction et la reconstruction étaient assez décevants. Pour la prédiction, on préfère donc se concentrer sur les configurations canoniques.

22. Rappel : erreur normalisée moyenne entre les séquences réelles de longueur H et les séquences prédites. Cette erreur fait une moyenne sur le nombre de séquences disponibles, toutes les composantes de l'état et la taille de l'horizon.

Exemple de résultats

La figure 6.23 illustre quelques résultats pour l'écoulement turbulent autour du cylindre. La décomposition spectrale de la matrice DMD (réduction POD) fait apparaître des valeurs propres complexes de module inférieur ou égal à un. Le modèle restitue une dynamique oscillatoire amortie, ce que l'on vérifie avec la prédiction récursive du premier coefficient POD. D'un point de vue quantitatif, l'erreur normalisée moyenne augmente avec l'horizon h (barres bleues) tandis que la qualité des fonctions propres identifiées diminue (courbes vertes). Cela traduit que les fonctions propres construites avec les vecteurs propres sont fallacieuses et que la *vraie* dynamique est non linéaire pour ce choix d'observables. En changeant la réduction, les erreurs de prédiction sont différentes. En particulier, la dynamique latente de la réduction VAE semble plus adaptée à une modélisation linéaire. Avec les résultats de l'annexe F, ce n'est pas la tendance observée pour la couche de mélange où la réduction POD est plus adaptée.

Focus sur l'écoulement laminaire autour du cylindre

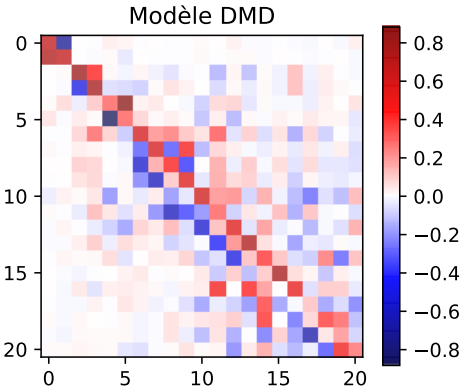
La DMD est facilement interprétable pour l'écoulement laminaire autour du cylindre. On peut complétement visualiser la différence entre une trajectoire réelle et une trajectoire récursive car l'état latent n'a que cinq composantes. Les figures 6.24 et 6.25 montrent les trajectoires pour chaque réduction. Avec la décomposition orthogonale propre, le modèle linéaire ne permet pas de prédire la contribution du cinquième mode [23]. Pour la réduction non linéaire variationnelle, la dynamique apprise est amortie et ne permet pas de rester sur un cycle limite. Pour les réductions linéaires non orthogonales, la dynamique de chaque composante est bien capturée par un modèle linéaire. En regardant les valeurs propres pour chacun des cas (figure 6.26), on s'aperçoit que les spectres des réductions linéaires sont quasi-identiques. Pourtant, la contribution d'amortissement (valeur propre réelle de modèle inférieur à 1) a un fort impact pour la réduction POD mais pas les réductions LAE et LVAE. La solution réside dans la non orthogonalité des modes, ce que l'on retrouve en écrivant les équations ci-dessous.

→ Dynamique POD

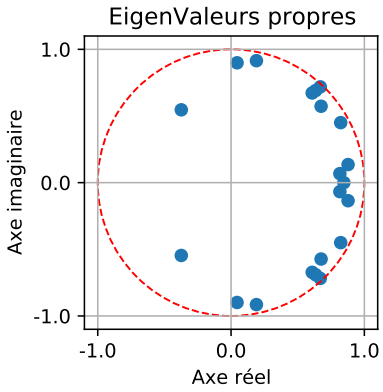
Pour la dynamique d'évolution POD, les quatre premiers coefficients ont une dynamique linéaire : les deux paires de modes correspondent à des fluctuations de vitesse périodiques autour du champ moyen. La dynamique du dernier mode a_4^{POD} est non linéaire. Le meilleur modèle linéaire pour la dynamique s'écrit :

$$a^{\mathrm{POD}}(k+1) = K^{\mathrm{POD}} a^{\mathrm{POD}}(k) = \begin{bmatrix} \alpha & \beta & 0 & 0 & 0 \\ -\beta & \alpha & 0 & 0 & 0 \\ 0 & 0 & \delta & \gamma & 0 \\ 0 & 0 & -\gamma & \delta & 0 \\ 0 & 0 & 0 & 0 & \Delta \end{bmatrix} a^{\mathrm{POD}}(k)$$

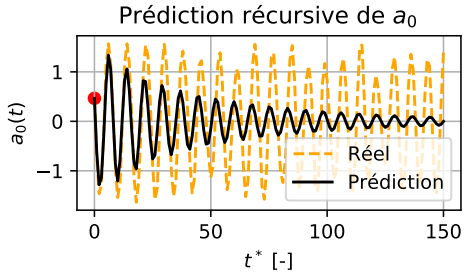
23. La littérature montre qu'il faut considérer des observables non linéaires pour récupérer l'énergie perdue, ce que l'on observera avec la DMD étendue. Voir Loiseau et Brunton (2018a).



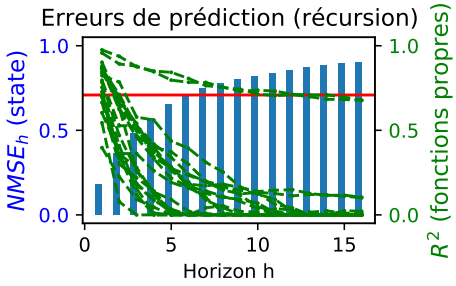
(a) Modèle linéaire pour la dynamique POD. Approximation finie de l'opérateur de Koopman.



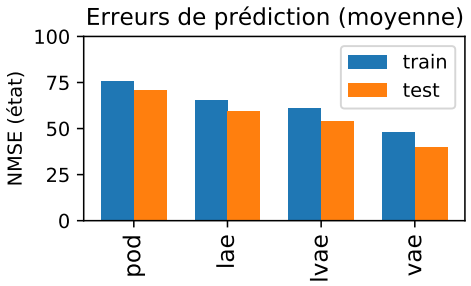
(b) Valeurs propres complexes de la meilleure dynamique linéaire POD.



(c) Prédiction récursive du premier coefficient POD à partir d'une condition initiale.



(d) Erreurs récursive et qualité des fonctions propres (données de test) pour la réduction POD.



(e) Erreurs de prédiction (moyenne sur tout l'horizon) pour toutes les réductions.

Figure *6.23* – *Quelques résultats de la décomposition modale dynamique pour l'écoulement turbulent autour du cylindre.*

avec les valeurs $\alpha, \beta, \delta, \gamma$ données sur la figure 6.27. Les deux matrices de rotation sont des représentations polaires des deux valeurs propres complexes placées sur le cercle unité. La valeur propre réelle Δ est de module inférieur à l'unité donc la dynamique non linéaire de a_4^{POD} est modélisée par un amortissement.

→ Dynamique LAE

Pour la dynamique d'évolution LAE, l'approximation de l'opérateur de Koopman est une matrice pleine que l'on diagonalise :

$$\begin{aligned} a^{\mathrm{LAE}}(k+1) = K^{\mathrm{LAE}} a^{\mathrm{LAE}}(k) = \Psi \Lambda \Psi^{-1} a^{\mathrm{LAE}}(k) \\ \implies b^{\mathrm{LAE}}(k+1) = \Psi^{-1} a^{\mathrm{LAE}}(k+1) = \Lambda b^{\mathrm{LAE}}(k) \end{aligned}$$

À quelques transformations unitaires près, la matrice Λ est identique à K^{POD}. Cela signifie que dans la base des vecteurs propres de K^{LAE}, la dynamique est identique à la dynamique POD. Pour la i-ème composante de l'état LAE, on peut écrire :

$$\begin{aligned} a_i^{\mathrm{LAE}}(k+1) = \Psi_{i,0} \left[\alpha b_0^{\mathrm{LAE}}(k) + \beta b_1^{\mathrm{LAE}}(k) \right] + \Psi_{i,1} \left[-\beta b_0^{\mathrm{LAE}}(k) + \alpha b_1^{\mathrm{LAE}}(k) \right] \\ + \Psi_{i,2} \left[\delta b_2^{\mathrm{LAE}}(k) - \gamma b_3^{\mathrm{LAE}}(k) \right] + \Psi_{i,3} \left[\gamma b_2^{\mathrm{LAE}}(k) + \delta b_3^{\mathrm{LAE}}(k) \right] \\ + \Psi_{i,4} \Delta b_4^{\mathrm{LAE}}(k) \end{aligned}$$

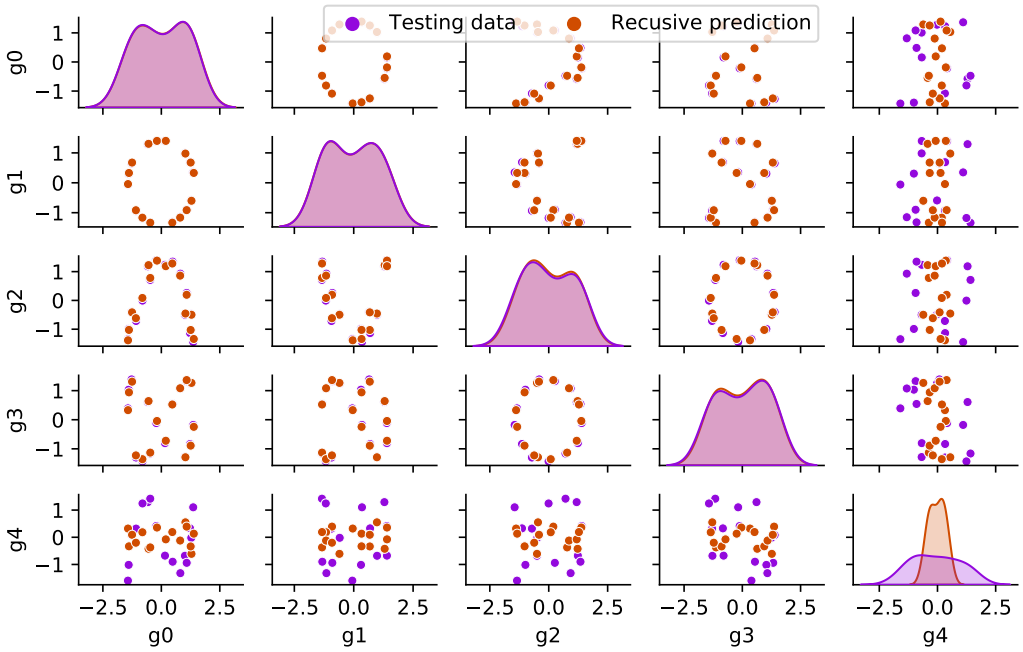
où la matrice Ψ peut être lue sur la figure 6.27. Avec cette écriture, on comprend que la dynamique amortie est présente dans chaque composante (valeur propre Δ) mais absorbée par la dynamique du cycle limite (matrices par blocs).

→ Cycle limite

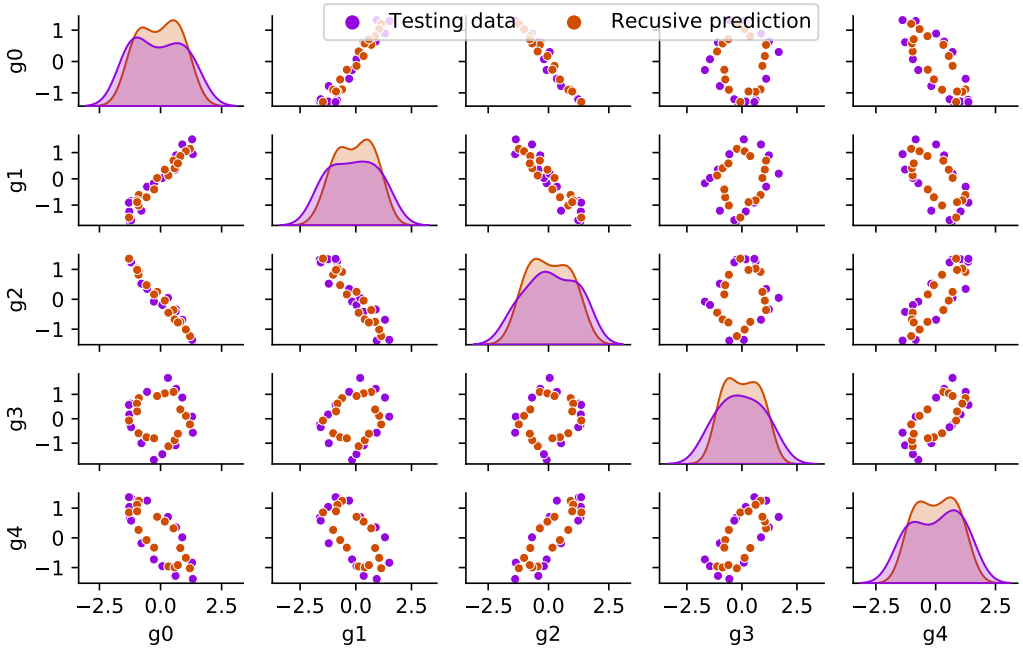
La matrice de rotation α, β est construite sur la valeur propre complexe $\lambda = 0.359 \pm i0.933$. L'angle est d'environ 69° donc une période correspond à 5.22 pas de temps adimensionnés. Cela correspond à un strouhal $\mathrm{St}_1 = 0.191$ soit une fréquence $f_1 = 0.955$ Hz.

La matrice de rotation δ, γ est construite sur la valeur propre complexe $\lambda = -0.743 \pm i0.669$. L'angle est d'environ 138° donc une période correspond à 2.61 pas de temps adimensionnés. Cela correspond à un strouhal $\mathrm{St}_2 = 0.383$ soit une fréquence $f_2 \approx 2f_1$.

La fréquence f_1 est associée à la dynamique périodique du coefficient de portance tandis que la fréquence f_2 est associée à la dynamique périodique du coefficient de traînée. La décomposition modale dynamique permet donc d'identifier les fréquences de l'écoulement.

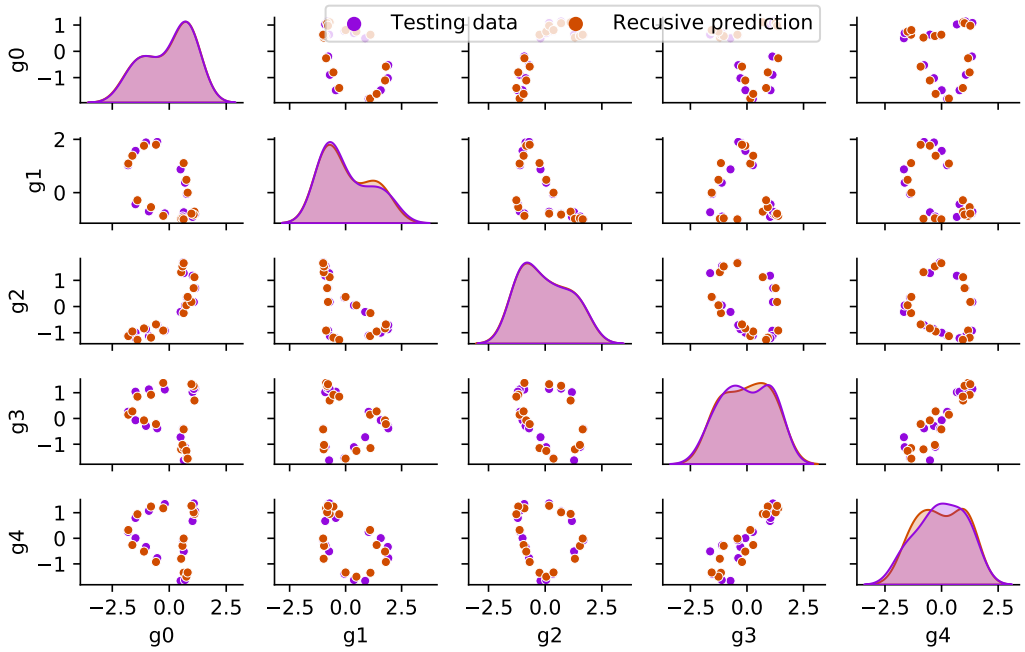


(a) Réduction POD

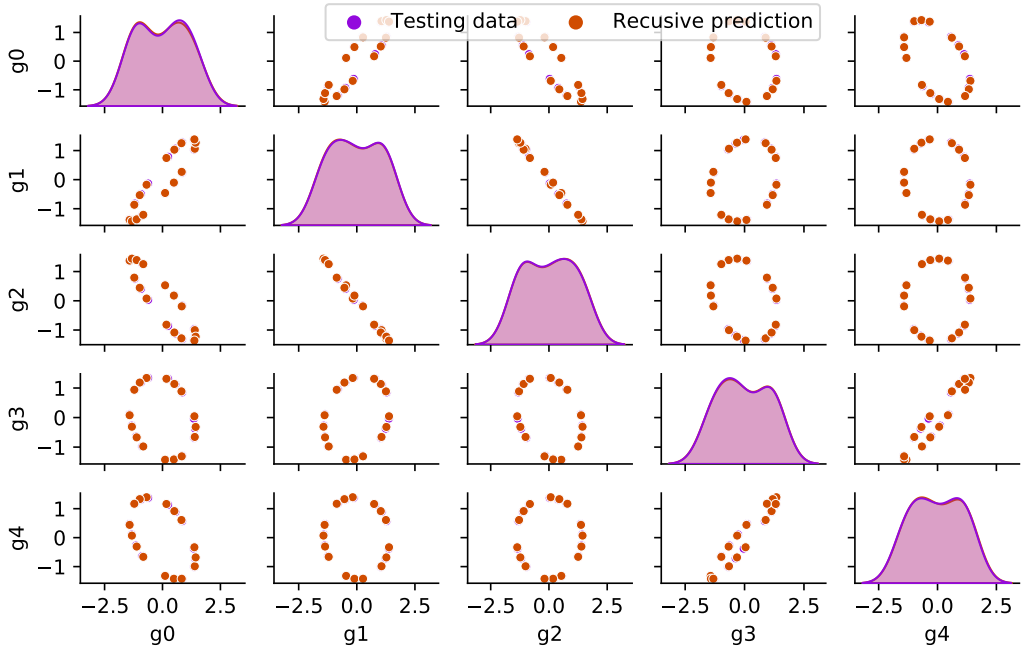


(b) Réduction VAE

Figure *6.24* – *Prédiction d'une trajectoire POD et d'une trajectoire VAE pour l'écoulement laminaire autour du cylindre. Le modèle linéaire ne parvient pas à capturer l'intégralité de la dynamique.*



(a) Réduction LAE



(b) Réduction LVAE

Figure *6.25* – *Prédiction d'une trajectoire LAE et d'une trajectoire LVAE pour l'écoulement laminaire autour du cylindre. Pour ces réductions, une approximation linéaire de la dynamique est pertinente.*

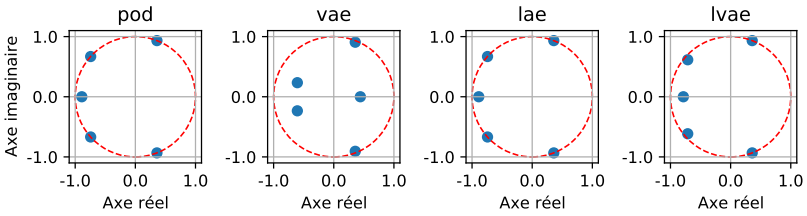
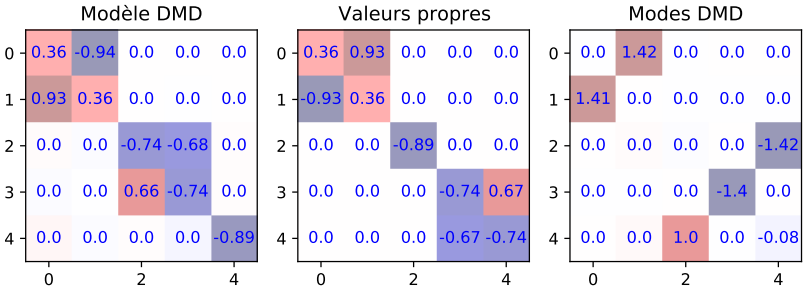
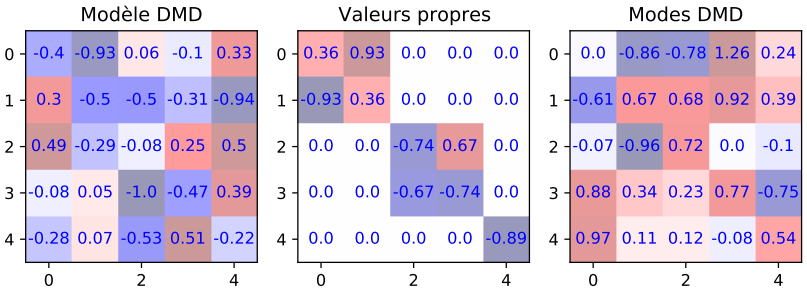


Figure *6.26* – *Spectre de l'approximation finie de l'opérateur de Koopman par décomposition modale dynamique de l'écoulement laminaire autour du cylindre.*



(a) Décomposition spectrale de la matrice K^{POD}. La matrice étant déjà diagonale par blocs, les modes dynamiques sont les directions propres.



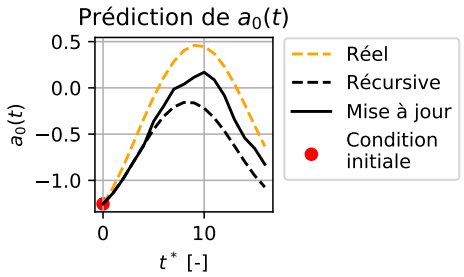
(b) Décomposition spectrale de la matrice K^{LAE}. La matrice est pleine et sa diagonalisation fait apparaître K^{POD}.

Figure *6.27* – *Matrices en jeu dans la décomposition modale dynamique de l'écoulement laminaire autour du cylindre.*

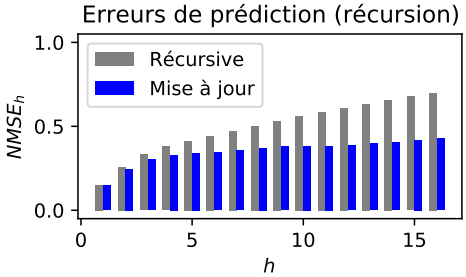
6.3.3 | Corrections par modèles directs

Pour la majeure partie des configurations, les composantes de l'état latent ne forment pas un espace invariant par l'opérateur de Koopman. Dans ce cas,

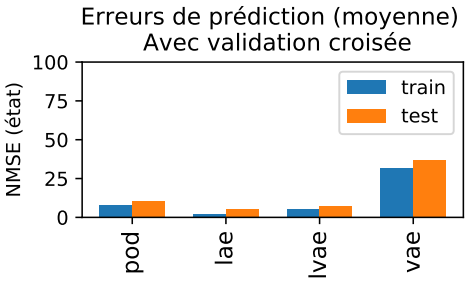
utiliser un modèle dynamique linéaire pour avancer l'état latent est erroné. Pour améliorer les prédictions, on peut utiliser des modèles directs qui corrigent la prédiction récursive à un horizon donné. Ici, seize modèles directs (arbres de décisions boostés) sont appris pour corriger les seize premiers pas de temps d'une prédiction récursive. La figure 6.28 illustre quelques résultats pour la couche de mélange. Sur la prédiction d'un coefficient LAE, l'utilisation des modèles directs permet une meilleure estimation. Sur la prédiction des coefficients VAE, l'erreur est en moyenne plus faible en utilisant les modèles directs. Concernant le sur-apprentissage, la validation croisée des arbres de décision permet de diminuer les écarts entre l'erreur de prédiction d'apprentissage et l'erreur de prédiction de test. Ces commentaires sont valables pour tous les écoulements, les résultats quantitatifs complets étant en annexe F.



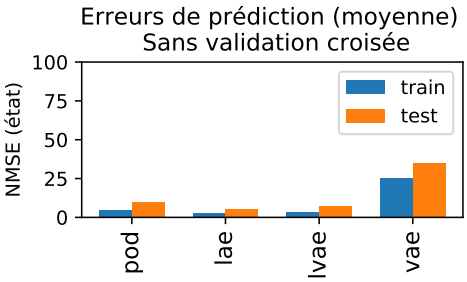
(a) Prédiction d'un coefficient LAE avec et sans modèles directs



(b) Erreur récursive avec et sans modèles directs pour la réduction VAE



(c) Erreurs de prédiction (moyennée sur tout l'horizon) avec modèles directs validés



(d) Erreurs de prédiction (moyennée sur tout l'horizon) avec modèles directs non validés

Figure *6.28* – *Influence des modèles directs pour la prédiction de la couche de mélange.*

6.3.4 | Décomposition d'ordre élevé

Dans la décomposition modale dynamique d'ordre élevé (HODMD), les composantes de l'état latent et leur histoire sont utilisées comme observables. La logique de construction du modèle est illustrée avec la figure 6.29 pour le sillage du cylindre rectangulaire (réduction POD). L'intérêt d'ajouter des retards se détecte en traçant la fonction d'auto-corrélation. Pour une composante de l'état latent, celle-ci indique les corrélations entre le signal et le signal décalé. Ici, la corrélation est significative au rang un et plus et on observe un motif sinusoïdal. Un modèle auto-régressif est donc pertinent *pour cette composante* et le nombre de retards est déterminé par l'auto-corrélation partielle. Pour un travail sur toutes les composantes, on utilise un critère d'information. Le nombre de retards optimal est celui qui minimise le critère Bayésien (BIC) ou d'Akaike (AIC). Pour l'écoulement turbulent autour du cylindre, le critère BIC indique d'ajouter trois retards, ce qui mène à $(3 + 1) \times r_{0.8} = 84$ observables. L'approximation de l'opérateur de Koopman fait apparaître une bande $r_{0.8} \times 84$ pour expliquer l'état $a(k + 1)$ par les composantes $[a(k) \mid a(k - 1) \mid a(k - 2) \mid a(k - 3) \mid a(k - 4)]$, ce qui est cohérent avec la structure présentée dans la partie méthodologique du manuscrit.

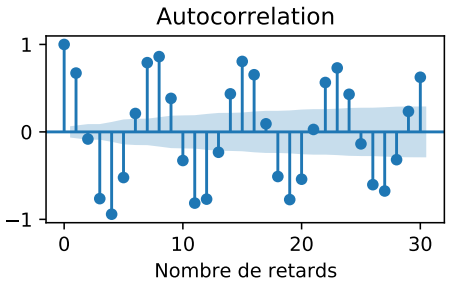
Le tableau 6.5 indique les retards déterminés par les critères d'information. De manière générale, le critère BIC semble plus adapté car le nombre de retards déterminé par le critère AIC est artificiellement élevé [24]. Pour l'écoulement laminaire autour du cylindre, le nombre de retards est important pour les deux critères, ce qui n'est pas totalement expliqué [25]. D'un point de vue résultats, la figure 6.30 montre les erreurs de prédictions de l'état pour la DMD et la HODMD. On observe que l'ajout des retards diminue légèrement l'erreur de prédiction pour les cas 1 et 2. Pour le sillage laminaire, les états retardés forment un espace invariant par l'opérateur de Koopman quelque soit la méthode de réduction. On dispose donc d'un modèle linéaire pour cet écoulement mais il est trop complexe car il considère $(12 + 1) \times r_{0.99} = 65$ modes dynamiques, un nombre élevé pour une dynamique périodique aussi simple.

Écoulement	POD	LAE	LVAE	VAE
Cylindre laminaire	[12, 12]	[12, 12]	[12, 12]	[12, 12]
Couche de mélange	[13, 8]	[23, 6]	[25, 8]	[7, 1]
Cylindre turbulent	[5, 3]	[5, 2]	[5, 2]	[4, 1]

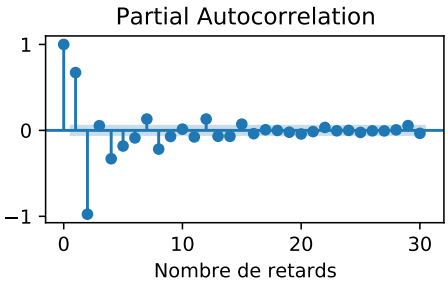
Tableau 6.5 *– Nombres de retards pour chaque configuration. Les résultats sont donnés sous la forme [retards AIC, retards BIC].*

24. Comme on laisse la machine faire le travail, il suffit d'un critère légèrement plus faible à 23 retards qu'à 6 retards pour sélectionner 23 retards. Mais d'un point de vue pratique, on préférerait travailler avec les 6 retards.

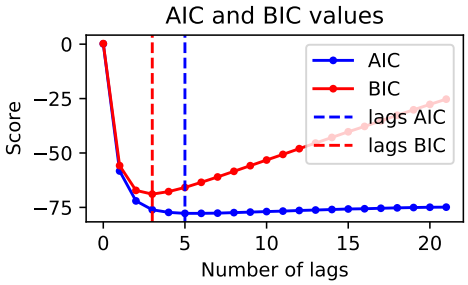
25. Sans doute lié au faible nombre d'échantillons disponibles, pas suffisant pour faire des statistiques.



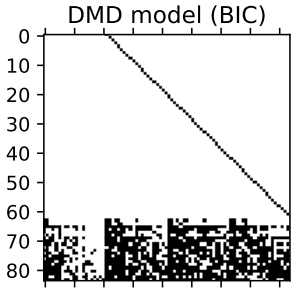
(a) Autocorrélation du premier coefficient POD et intervalle de significativité



(b) Autocorrélation partielle du premier coefficient POD et intervalle de significativité



(c) Critère d'information



(d) Approximation de l'opérateur de Koopman

Figure *6.29* – *Quelques résultats de la décomposition modale dynamique d'ordre élevé pour le sillage turbulent.*

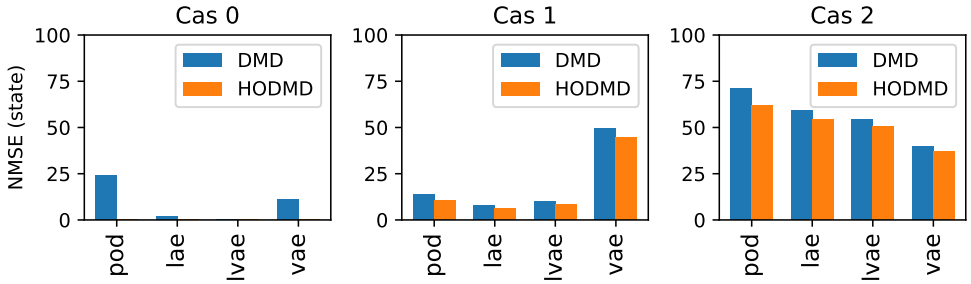


Figure *6.30* – *Erreurs de prédiction (moyenne sur l'horizon) des états de test calculées avec le modèle DMD et le modèle HODMD (sélection BIC).*

6.3.5 | Décomposition étendue

La décomposition modale dynamique étendue est le cadre le plus général pour l'approximation de l'opérateur de Koopman. Ici, on choisit comme observables des

monômes au plus quadratiques (EDMD) ou des fonctions à base radiales (EDMD RBF).

EDMD

La DMD étendue avec observables quadratiques est un choix raisonnable quand on sait que les équations de Navier Stokes sont quadratiques [26]. Le nombre de monômes possibles augmente de manière combinatoire avec la dimension de l'état latent mais la plupart de ces observables est vraisemblablement fallacieuse. Pour éviter les monômes parasites dans la construction du modèle linéaire, une sélection par LASSO est effectuée. Le niveau de pénalité est optimisée par validation croisée de type *hold-out*, en considérant l'erreur un pas de temps et l'erreur de prédiction moyenne. La figure 6.31 illustre les résultats obtenus pour l'écoulement laminaire autour du cylindre (réduction POD). La matrice qui est montrée est l'approximation moindres carrés du passage entre l'espace d'état et l'espace des observables. Sans surprise, la matrice contient une diagonale car les composantes de l'état latent sont présents dans le dictionnaire des monômes. Sur le graphique de l'erreur moyenne normalisée en fonction de l'horizon, il n'y a pas de barres bleues : l'erreur récursive est nulle car les observables décrivent *presque* un espace invariant par Koopman. On dit *presque* car malgré des fonctions propres de bonne qualité (courbes vertes), certaines sont encore fallacieuses. En calculant le spectre de la matrice de la dynamique, toutes les valeurs propres ne sont pas sur le cercle unité ; la combinaison des modes dynamiques permet toutefois une bonne qualité prédictive pour le cinquième mode qui était complètement amorti avec la DMD classique. Enfin, l'erreur de validation croisée est minimale pour une pénalité nulle, synonyme que le LASSO ne sélectionne pas de variables pour ce cas précis.

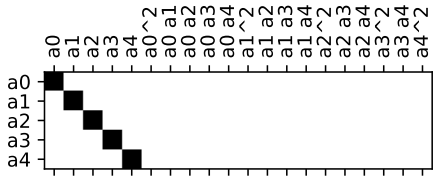
Les résultats sont moins probants pour la réduction non linéaire variationnelle. L'encodage par tangente hyperbolique rend difficile [27] la dérivation d'un modèle réduit *analytique* pour l'état latent VAE et les observations quadratiques de l'état latent ne sont peut-être pas adaptées. La figure 6.32 compare les erreurs de prédiction pour la DMD et l'EDMD. Si une petite baisse de l'erreur est observée avec l'écoulement turbulent autour du cylindre, les résultats sont mauvais pour la couche de mélange. Avec les résultats de l'annexe F, le gain semble un peu plus substantiel sur l'erreur un pas de temps. De manière générale, on retiendra que la DMD étendue aux observations quadratiques est plus adaptée aux écoulements de sillage réduits linéairement.

EDMD RBF

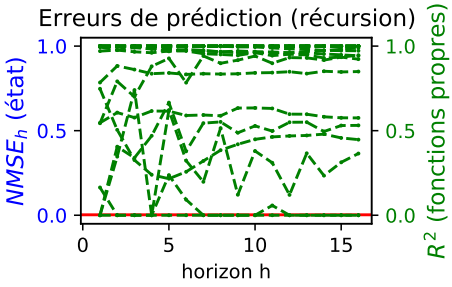
Utiliser des observations quadratiques de l'état n'est pas toujours optimal. Sans une connaissance précise du système, il est judicieux d'utiliser des fonctions à base radiale dont les centres sont déterminés par partitionnement. Le nombre de clusters est choisi par le critère du coude, qui détermine le point à partir duquel

26. Avec une réduction orthogonale propre, on peut d'ailleurs projeter facilement les équations et faire apparaître explicitement les interactions quadratiques entre les coefficients POD

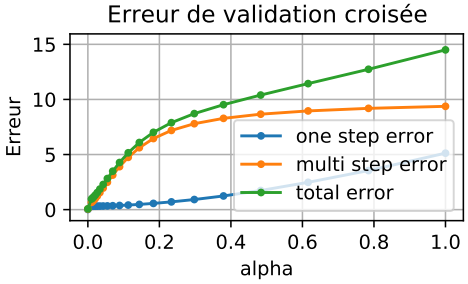
27. Pour pas dire impossible...



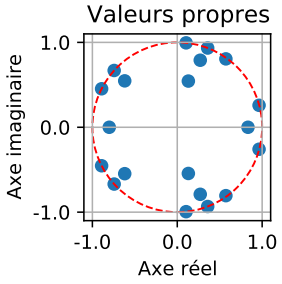
(a) Poids de Koopman (matrice B). Les composantes de l'état latent sont présents dans le dictionnaire d'observables, d'où la diagonale.



(b) Erreurs et qualité des fonctions propres en fonction de l'horizon. Quelques fonctions de bonne qualité, d'autres fallacieuses.



(c) Chemin du LASSO. L'erreur de prédiction (orange) est celle qui contribue le plus à l'erreur de validation.



(d) Spectre. Plusieurs valeurs propres sur le cercle unité, quelques unes à l'origine d'oscillations amorties.

Figure *6.31* – *Résultats de la décomposition modale étendue pour l'écoulement laminaire autour du cylindre (réduction POD).*

l'ajout de clusters ne vaut plus le gain en inertie. Le coefficient de silhouette permet ensuite de vérifier que les partitions sont suffisamment distantes. Sur la figure 6.33, on présente quelques résultats pour le sillage laminaire réduit par VAE. L'inertie en fonction du nombre de partitions montre que six clusters semblent raisonnables pour partitionner les états latents. L'analyse des coefficients de silhouette montre que ces clusters sont denses et plutôt bien séparés (score moyen proche de 1). Concernant les valeurs propres, celles-ci se rapprochent du cercle unité comparé à la décomposition modale dynamique classique (figure 6.26). Enfin, la matrice des poids de Koopman est pleine ; c'est un signe que le passage de l'espace des observables vers l'espace d'état n'est pas trivial.

La figure 6.34 montre les erreurs de prédiction de test dans l'espace d'état et

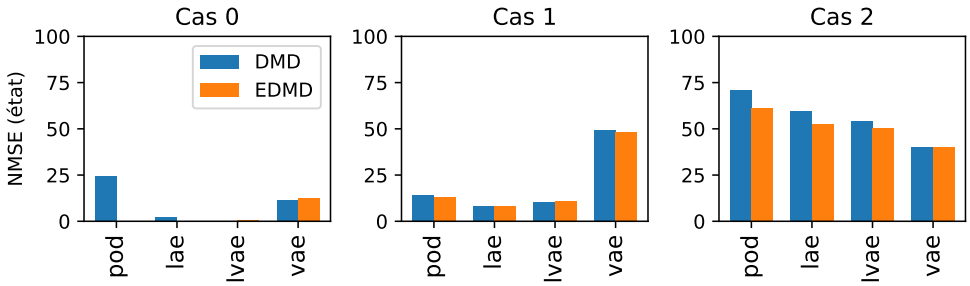


Figure *6.32* – *Erreurs de prédiction dans l'espace d'état (moyenne sur tout l'horizon) pour une approximation de l'opérateur de Koopman par DMD ou EDMD. Utiliser des observations quadratiques permet uniquement d'améliorer la prédiction des réductions linéaires pour les écoulements de sillage*

dans l'espace des observables. Comparé aux mesures quadratiques de l'état, les mesures radiales semblent plus adaptées pour construire un modèle dynamique linéaire. C'est notamment le cas pour la réduction non linéaire variationnelle du sillage turbulent. Ce gain n'est toutefois pas observé dans l'espace d'état. Cela est dû à une mauvaise inversion des observables gaussiens : une approximation linéaire par moindres carrés n'est pas suffisante. Il faudrait envisager une inversion plus flexible, par exemple par réseau de neurones.

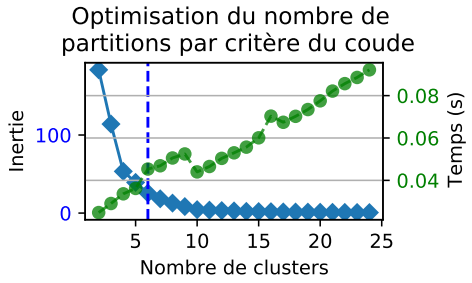
6.3.6 | Apprentissage des observables

Dans les méthodes précédentes, les observations de l'état latent étaient prédéfinies. Maintenant, on souhaite construire le dictionnaire en apprenant des observables qui évoluent linéairement. Pour ce faire, les observables sont les sorties d'un réseau de neurones dont les paramètres sont optimisés par une descente alternée. Une itération contient deux étapes :

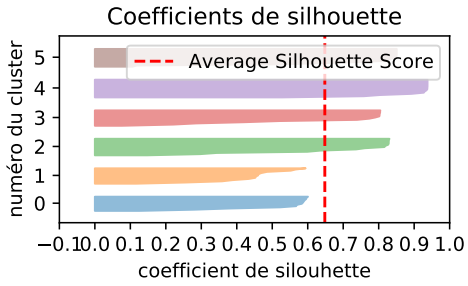
1. Utilisation du réseau actuel pour construire l'approximation de l'opérateur de Koopman. Les observables sont fixés et on effectue une régression d'arête. La pénalité est optimisée par validation croisée de type *hold out*.
2. L'opérateur de Koopman est fixé. On optimise les paramètres du réseau par descente de gradient. La fonction coût contient trois termes d'erreurs : l'erreur un pas de temps, l'erreur de prédiction et l'erreur d'inversion des observables par régression linéaire.

Pour ne pas apprendre des fonctions d'observations nulles, deux observables fixes (fonctions radiales) sont ajoutées dans le dictionnaire. Les hyper-paramètres du réseau sont optimisés par ASHA [28]. La figure 6.35 illustre un exemple d'optimisation pour le sillage laminaire réduit avec la POD. Il s'agit d'un

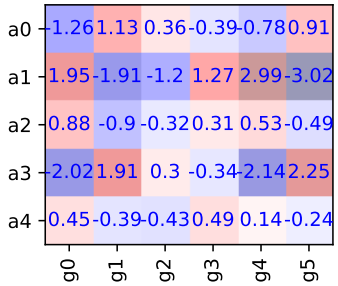
28. *Asynchronous Successive Halving Algorithm*



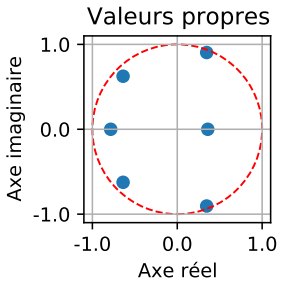
(a) Inertie (somme des distances intra-clusters) en fonction du nombre de clusters. Le coude indique le nombre optimal de partitions.



(b) Analyse de la silhouette. Pour chaque échantillon d'un cluster, le coefficient de silhouette indique s'il est loin (+1) ou proche (-1) d'autres clusters.



(c) Matrice des poids de Koopman. La matrice est pleine comparé aux observations quadratiques.

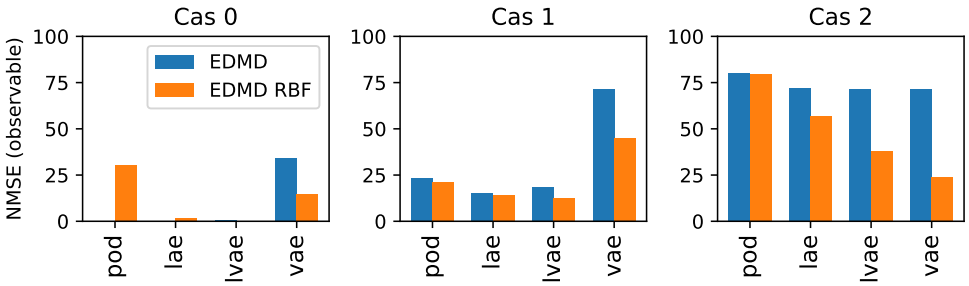


(d) Valeurs propres de l'approximation de l'opérateur de Koopman.

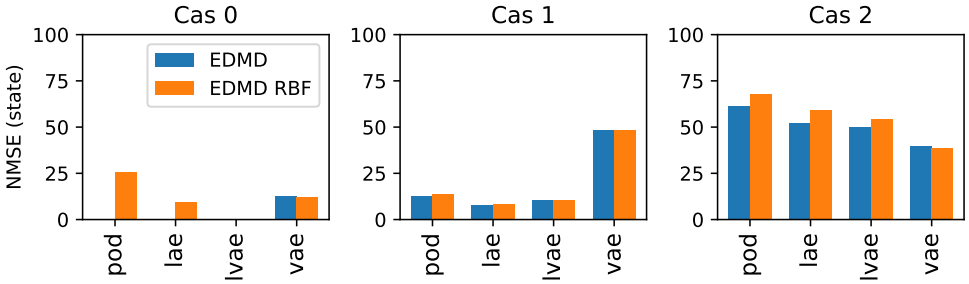
Figure *6.33* – *Résultats de la décomposition modale dynamique étendue aux observations radiales. Illustré sur le sillage laminaire réduit par auto-encodeur non linéaire variationnel.*

graphique de coordonnées parallèles qui présente toutes les configurations étudiées. Un réseau est construit sur la base d'un taux d'oubli, de fonctions d'activations, d'un taux d'apprentissage (lr), d'un nombre de couches et d'un nombre de neurones par couches (nh). Les premières combinaisons d'hyper-paramètres ne sont testées que pour une seule itération. Les réseaux les plus performants (coût de validation le plus faible, "loss") sont promus au rang suivant ("rung") et entraînés sur quatre fois plus d'itérations. On arrête l'optimisation lorsque trois réseaux ont atteint le dernier palier (rung 3) et ont été entraînés sur au plus 64 itérations [29]. On sélectionne le réseau à l'erreur de validation plus faible et on le ré-entraîne sur l'intégralité des données d'entraînement. Pour cette optimisation, on montre

29. Donc 64 descentes de gradient complètes pour la deuxième étape d'une itération. Heureusement que la descente est arrêtée prématurément, sinon l'apprentissage serait incommensurablement long.



(a) Erreurs dans l'espace des observables. Bonnes performances pour EDMD RBF, notamment pour la réduction non linéaire variationnelle.

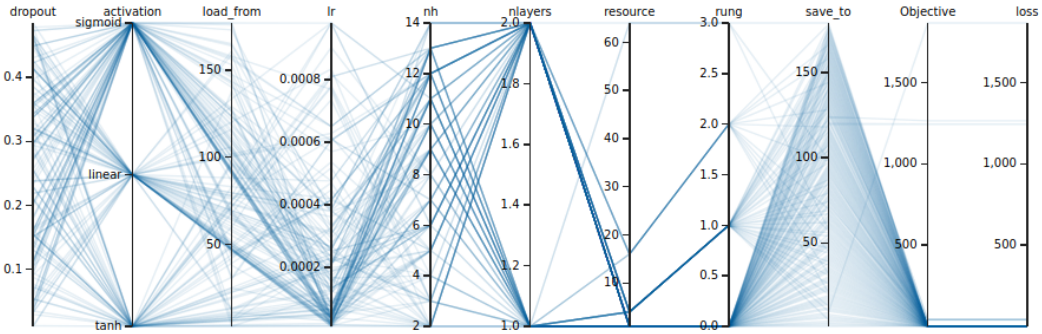


(b) Erreurs dans l'espace d'état. Mauvaises performances pour EDMD RBF car l'inverse des fonctions d'observations est mal approximée par régression linéaire.

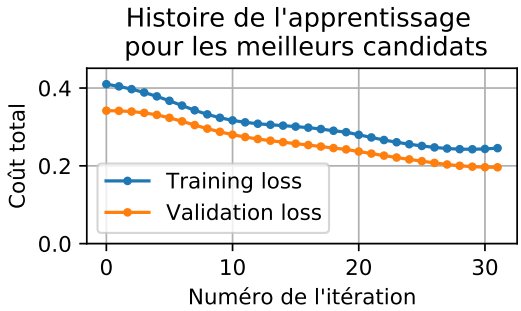
Figure *6.34* – *Comparaison des erreurs de prédictions (moyenne sur l'horizon) pour les méthodes EDMD et EDMD RBF.*

l'histoire de la fonction coût totale en fonction du nombre d'itérations. Seulement 32 itérations sont montrées car il s'agit du meilleur état possible pour les paramètres du réseau ; au delà, l'erreur de validation ré-augmente. À titre d'illustration, on montre également l'architecture générale d'un réseau de neurones sur la figure 6.36.

Sur la figure 6.37, on montre les résultats obtenus pour les trois écoulements. Dans l'espace des observables, on obtient de bonnes performances pour les écoulements de sillage. Le gain est surtout notable sur le sillage turbulent, quelque soit la méthode de réduction. Pour le sillage laminaire, les résultats sont corrects pour les réductions linéaires mais mitigés pour la réduction non linéaire. La figure 6.38 compare les spectres obtenus pour le sillage laminaire réduit avec la POD. L'apprentissage automatique du dictionnaire permet de retrouver les fréquences dominantes et une partie des valeurs propres de la matrice EDMD. Toutefois, quelques valeurs propres restent défectueuses, symptomatique d'observables qui ne sont pas invariants par l'opérateur de Koopman. Dans l'espace d'état, on est relativement déçu du pouvoir prédictif : mise à part le sillage laminaire, les performances sont très mauvaises pour la couche de mélange et le sillage turbulent.



(a) Coordonnées parallèles pour l'optimisation des hyper-paramètres. Une ligne correspond à une configuration d'hyper-paramètres, promue jusqu'à un certain palier et ayant un coût de validation.



(b) Histoire de la fonction coût pour les meilleurs hyper-paramètres. Un point correspond à une approximation de l'opérateur de Koopman (étape 1) et une descente de gradient (étape 2).

Figure *6.35* – *Résultats d'optimisation pour l'apprentissage automatique du dictionnaire. Illustré sur le sillage laminaire réduit avec la POD.*

Cela est dû aux observables appris qui sont difficilement inversibles par régression linéaire : les scores sont de 100% pour le sillage laminaire (inversion facile) mais ils sont mitigés pour les autres cas [30]. En annexe F, on compare EDMD_DL aux autres méthodes d'approximation de l'opérateur de Koopman et en particulier, avec EDMD_RBF_two. Cette dernière méthode correspond à EDMD RBF avec seulement deux clusters, afin de vérifier que EDMD DL apprend effectivement des observables et ne se contente pas des deux observables fixes.

Finalement, l'approche proposée est une direction intéressante pour apprendre

30. Rappel : le score compare les états réels et les états obtenus par l'inversion moindres carrés. Si la variance des états réels est complètement retrouvée dans l'inversion moindres carrés, le score vaut 1.

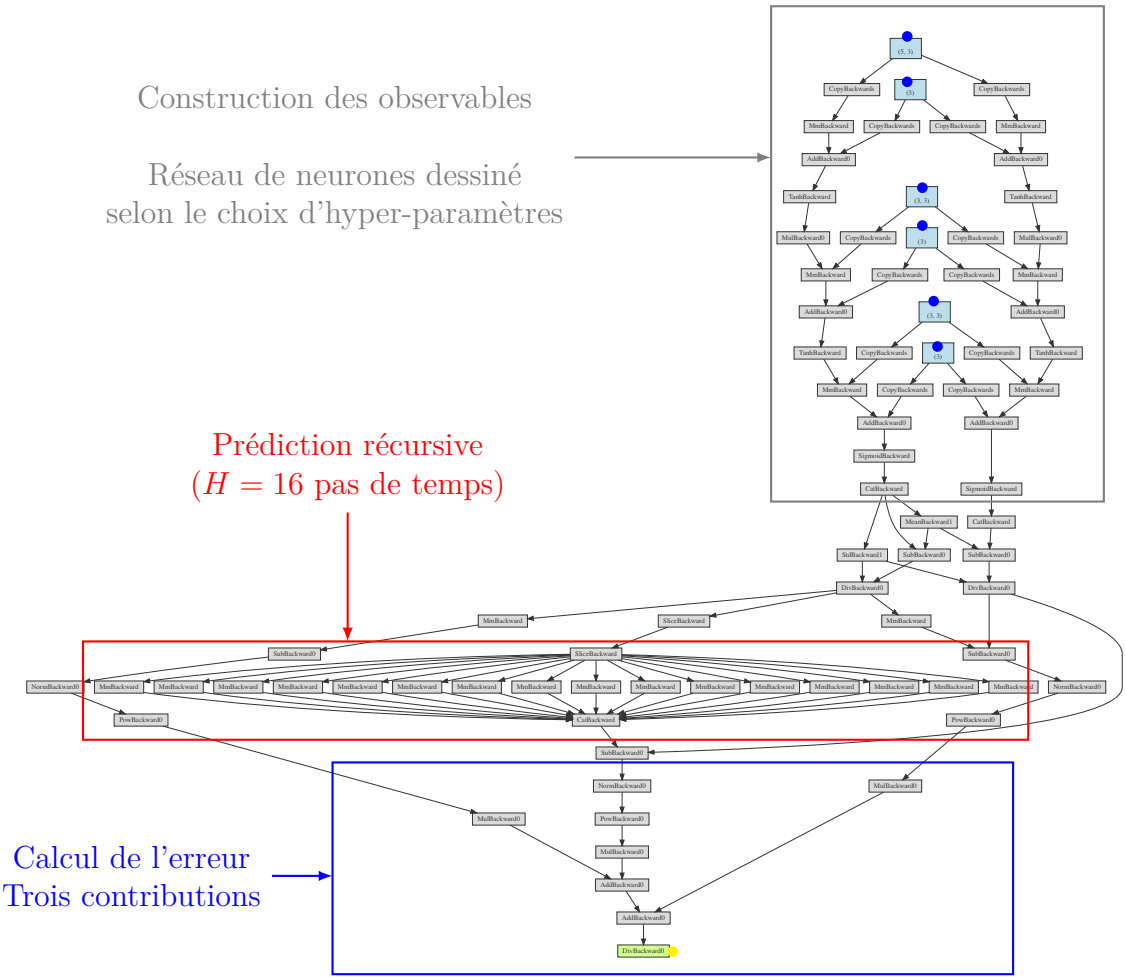
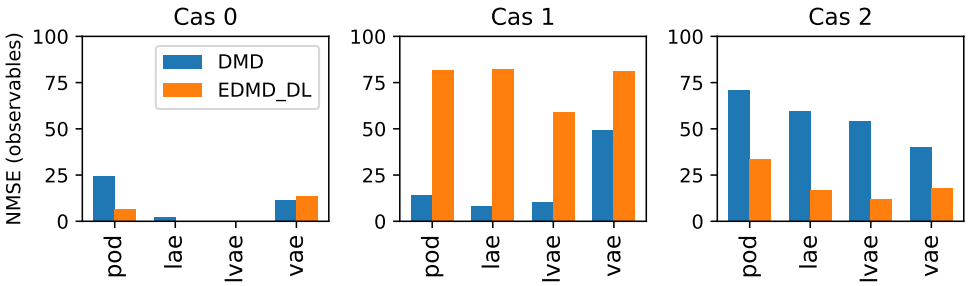
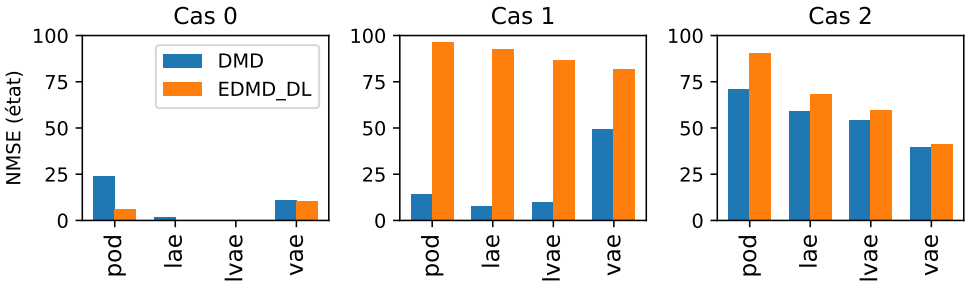


Figure *6.36* – *Exemple de schéma dynamique pour l'apprentissage du dictionnaire. Le plus difficile consiste à créer la fonction coût (point jaune), qui comprend trois termes : l'erreur un pas de temps, l'erreur de prédiction récursive (branche du milieu) et l'erreur d'inversion. Ce schéma est créé par* pyTorch *pour ensuite faire la rétro-propagation et optimiser les paramètres (points bleus). Ce schéma est celui obtenu pour le sillage laminaire, réduit avec la POD.*

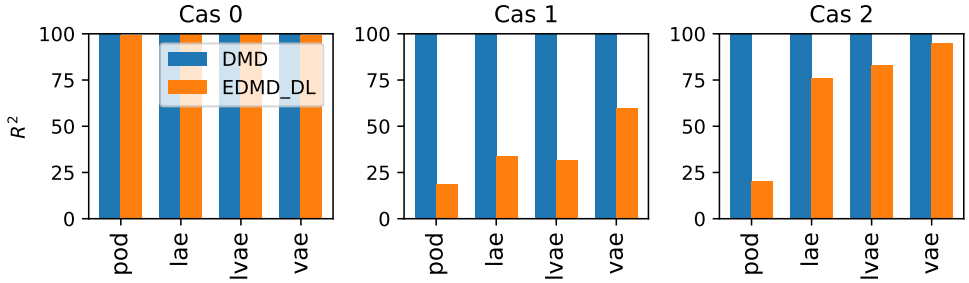
les observables. De nombreuses améliorations doivent toutefois être apportées. Tout d'abord, le premier brassage des hyper-paramètres s'est basé sur une seule itération. C'est un critère beaucoup trop brutal et il faudrait encourager l'exploration. Deuxièmement, l'inversion par régression linéaire n'est pas optimale. Il faudrait adjoindre un réseau de neurones pour cette tâche et ainsi améliorer le pouvoir prédictif dans l'espace d'état. Troisièmement, l'architecture proposée est simpliste : même nombre de neurones par couche et fonctions d'activations



(a) Erreurs de prédiction dans l'espace des observables. Pour le sillage turbulent, bonnes performances avec les fonctions d'observations apprises comparé aux observations quadratiques.



(b) Erreurs de prédiction dans l'espace d'état. Mauvaises performances avec les fonctions d'observations apprises



(c) Qualité de l'approximation linéaire de l'inverse des observables. Pour la DMD, la qualité est parfaite (score de 100%) car les composantes de l'état sont dans le dictionnaire d'observables. Pour EDMD DL, mauvaise qualité.

Figure *6.37* – *Comparaison quantitative entre les approximations de l'opérateur de Koopman par DMD et par EDMD avec apprentissage du dictionnaire.*

identiques. Il faudrait augmenter la flexibilité pour exploiter la puissance des réseaux de neurones. Un conseil serait de reprendre la stratégie sur des systèmes

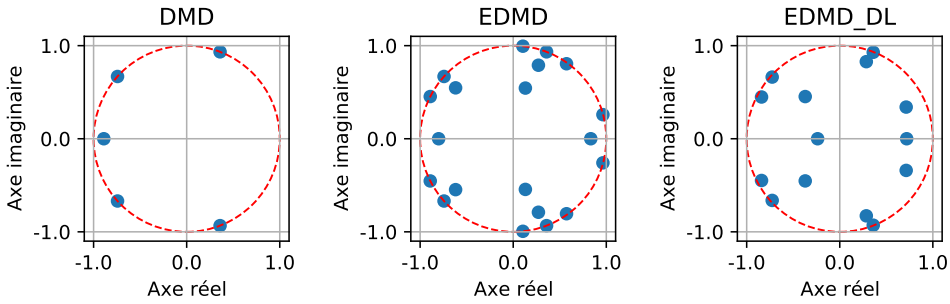


Figure *6.38* – *Comparaison des valeurs propres du sillage laminaire (réduction POD) déterminées par DMD, EDMD et EDMD DL.*

beaucoup plus simples tels que des oscillateurs[31] et faire des développements académiques.

6.3.7 | Conclusion

Dans cette section, différentes approximations de l'opérateur de Koopman ont été comparées. La décomposition modale dynamique est la plus simple à implémenter car elle consiste en une régression linéaire entre les états latents et les états latents décalés dans le temps. Or pour des systèmes non linéaires, les composantes de l'état latent ne forment pas un espace invariant par Koopman. Les prédictions récursives de l'état à partir d'une condition initiale sont donc erronées. Différentes extensions sont proposées :

- **L'ajout de modèles directs**. Des arbres de décisions boostés permettent de corriger les prédictions récursives. La validation croisée des arbres diminue l'écart entre les erreurs d'entraînement et de test.
- **La décomposition d'ordre plus élevé**. L'espace des observables est enrichi de l'histoire de l'état latent. Le nombre de retards est optimisé par critère d'information. Pour l'écoulement laminaire autour du cylindre, ce nombre de retards est artificiellement élevé.
- **La décomposition étendue**. Les observations sont des monômes au plus quadratiques ou des mesures gaussiennes de l'état latent. Dans le deuxième cas, les prédictions dans l'espace d'état sont mauvaises à cause d'une inversion difficile des observables.
- **La décomposition avec apprentissage automatique du dictionnaire**. Les observations sont les sorties d'un réseau de neurones. La méthode est complexe à implémenter, avec des résultats assez décevants. Cela mérite de l'investissement mais sur des systèmes plus académiques.

Pour le sillage laminaire et la couche de mélange, les méthodes ont montré de bonnes performances pour les réductions linéaires. Pour le sillage turbulent, les

31. Ou alors le sillage laminaire, mais avec beaucoup plus d'échantillons.

erreurs de prédiction sont les plus faibles pour la réduction non linéaire variationnelle. Sur un nouvel écoulement à prédire, le conseil serait de commencer par une réduction POD et un modèle DMD. Si le modèle dynamique est *trop* erroné, on peut envisager la décomposition étendue avec observables quadratiques. C'est une méthode de choix car la projection des équations de Navier Stokes sur le sous espace POD fait apparaître des termes quadratiques. Il faut toutefois faire attention au sur-apprentissage, potentiellement important si la dimension de l'état latent est importante et que tous les monômes possibles sont considérés.

Remarque : un travail préliminaire sur l'assimilation de données est présenté en annexe H. Il s'agit d'un article de conférence où des modèles purement orientés données sont combinés pour prédire l'écoulement sur le long terme. La procédure est testée avec des machines à vecteurs support pour la reconstruction et la décomposition modale dynamique pour la prédiction.

6.4 | Synthèse du chapitre

Dans ce chapitre, différents outils de l'apprentissage automatique ont été utilisés pour estimer des écoulements de complexité croissante. L'objectif était de tester la scalabilité de méthodes orientées données pour la réduction, la reconstruction et la prédiction du champ de vitesse d'un écoulement.

Pour la réduction, l'objectif était de définir une transformation d'auto-encodage du champ de vitesse. Différentes méthodes ont été implémentées, chacune avec leur avantages et leur inconvénients :

→ **La décomposition en modes propres orthogonaux**. En diagonalisant la matrice de covariance des clichés du champ de vitesse fluctuante, on en détermine les directions principales. Le film de l'écoulement est réécrit grâce à la superposition orthogonale de ces modes, ce qui fournit une approximation de faible rang optimale (dans le sens moindres carrés) de l'écoulement.

→ **L'auto-encodeur linéaire**. La POD est reformulée grâce à un réseau de neurones avec une seule couche cachée et des activations linéaires. La contrainte d'orthogonalité n'est plus imposée, ce qui a pour effet de faciliter l'interprétabilité des modes. Les structures sont déterminées par un critère énergétique (moindres carrés) mais aucune hiérarchie n'est imposée. Pour les écoulements absolument instables, les directions oscillent à la fréquence dominante.

→ **L'auto-encodeur linéaire variationnel**. Pour régulariser l'espace latent, les champs de vitesse sont encodés en distributions plutôt qu'en points. En gardant des activations linéaires, le sous espace généré par les modes est le même que les deux sous espaces POD et LAE. Pour cet auto-encodeur, la contrainte variationnelle a peu d'impact.

→ **L'auto-encodeur variationnel**. Les activations du réseau précédant sont changées en tangente hyperbolique. L'encodage et le décodage deviennent

difficilement interprétables (variété latente) et la non optimisation des hyper-paramètres se reflète dans une erreur d'auto-encodage plus élevée que les réductions linéaires. Toutefois, la régularisation de l'espace latent est visible : la robustesse du décodeur à des états latents perturbés est attestée.

En comparant les spectres des champs de vitesse auto-encodés, on s'aperçoit que les réductions linéaires filtrent l'information haute fréquence. Au contraire, la réduction non linéaire essaye de restituer l'intégralité du spectre mais souvent au détriment de l'énergie des premiers modes. Concernant la généralisation des modes appris, seul du sur-apprentissage est notable sur l'écoulement de la tour. On attribue cette faiblesse au peu d'échantillons disponibles et aux architectures non optimales des réseaux de neurones.

Pour la reconstruction, l'objectif était d'utiliser l'algèbre linéaire (méthode directe) ou l'apprentissage supervisé (méthode régressive) pour inférer l'état latent à partir d'une mesure du champ de vitesse fluctuante. La grille de capteurs a été déterminée par partitionnement amélioré, une méthode semi-supervisée qui permet un placement systématique. Les hyper-paramètres n'ayant pas été optimisé, le nombre de capteurs semble élevé pour les configurations les plus simples. De manière générale, les scores de reconstruction de l'état latent sont *bons*, ce que l'on attribue à la validation croisée des hyper-paramètres. Celle-ci est étudiée en détails pour chaque méthode régressive :

→ Pour la **régression linéaire** seul le paramètre de régularisation LASSO est validé. Une simple recherche par grille suffit. Le paramètre est unique pour l'approche multi-tâche mais il est spécifique pour chaque mode dans l'approche mono-tâche.

→ Pour la **régression par vecteurs supports** et les **arbres de décisions boostés**, plusieurs paramètres sont à valider. Les candidats sont sélectionnés aléatoirement dans une grille, ce qui limite la probabilité de sélectionner la combinaison optimale d'hyper-paramètres.

→ Pour la régression par **réseau de neurones**, les hyper-paramètres sont validés par optimisation génétique.

Deux tendances principales ressortent. D'abord, augmenter la densité de capteurs améliore le score de reconstruction. Ensuite, l'écoulement urbain est plutôt difficile à reconstruire, avec des scores d'estimation plutôt faibles. Concernant la robustesse des méthodes au bruit de mesures, une fonction d'estimation ne semble pas l'emporter sur l'autre. Le choix d'une méthode régressive dépend donc de critères plus pratiques : la disponibilité des données, l'interprétabilité souhaitée, le coût d'implémentation et le coût de calcul.

Pour la prédiction, l'objectif était d'apprendre un modèle dynamique pour l'état latent. Le travail s'est concentré sur la recherche d'une approximation finie de l'opérateur de Koopman. Différents choix d'observables ont été considérés :

→ **Décomposition modale dynamique**. Les observables sont les composantes de l'état latent. L'approximation de l'opérateur de Koopman est donc le meilleur modèle dynamique linéaire de l'état latent. Sans

surprise, la majeure partie des fonctions propres identifiées sont fallacieuses car les composantes de l'état latent ne forment pas un sous espace invariant par Koopman. Adjoindre des modèles de prédiction direct (arbres de décisions boostés) permet d'améliorer l'estimation.

→ **Décomposition modale dynamique d'ordre élevé**. Les observables contiennent l'état latent et son histoire. Les modèles prédictifs sont améliorés mais le nombre de retards est artificiellement élevé dans le cas du cylindre 2D.

→ **Décomposition modale dynamique étendue**. Les observables sont des fonctions non linéaires de l'état, ici des observations quadratiques ou gaussiennes. Pour ce second choix, les prédictions ne sont pas améliorées dans l'espace d'état car l'approximation moindres carrés de l'inverse des observables est mauvaise.

→ **Décomposition modale dynamique avec apprentissage automatique du dictionnaire**. Les observables sont les sorties d'un réseau de neurones que l'on optimise pour avoir une représentation linéaire de la dynamique. Les résultats sont assez décevants pour cette première étude ; la méthode mériterait toutefois un investissement supplémentaire.

Enfin, un travail préliminaire en annexe H montre dans quelle mesure des modèles de reconstruction et de prédiction peuvent être combinées pour de l'assimilation de données. Ce cadre constitue une direction prometteuse pour la prédiction long terme d'un écoulement.

Le mot de la fin

Ce chapitre met en évidence qu'il existe pléthore de méthodes pour estimer un écoulement. Pour que les méthodes d'apprentissage automatique donnent des résultats pertinents, il est nécessaire de passer par une étape de validation croisée. Sur la partie *prédiction*, les résultats sont assez mitigés et tous les gains observés pour les raffinements étudiés ne valent pas la complexité de la méthode.

7. Conclusion et perspectives

L'objectif principal de cette thèse était de comprendre comment l'apprentissage automatique pouvait se mettre au service de la mécanique des fluides. On a ainsi proposé une méthodologie pour pouvoir réduire, reconstruire et prédire un écoulement à partir de données. Dans cette conclusion, on présente la synthèse générale des travaux ainsi que les perspectives.

7.1 | Synthèse générale

La **première** étape de cette thèse a été de comprendre l'intérêt des méthodes orientées données pour la mécanique des fluides (chapitre 1). On a commencé par présenter la physique multi-échelles d'un écoulement qui rend les missions de modélisation, de réduction et de contrôle complexes. En effet, lorsque les effets non linéaires de l'inertie dominent les effets linéaires de la viscosité, l'écoulement devient multi-échelles et la résolution des problèmes d'optimisation devient impossible analytiquement ou numériquement. La mécanique des fluides n'échappant pas au *Big Data* (disponibilité croissante de données expérimentales et numériques) une transition est en train de s'effectuer pour résoudre les problèmes d'optimisation à partir de données. Dans ce contexte, on a présenté un historique du *machine learning* ainsi que les applications majeures en mécanique des fluides. On a notamment précisé les outils pour la réduction et la modélisation :

- → La **réduction** consiste à extraire des données des structures cohérentes pour réécrire le film de l'écoulement avec une approximation de faible rang. La méthode la plus classique est la décomposition en modes propres orthogonaux, qui permet une réduction linéaire et orthogonale des bases de données. Avec l'essor des réseaux de neurones, les réductions peuvent maintenant être non linéaires et non orthogonales.
- → La **modélisation** consiste à identifier des modèles interprétables (peu de termes) et généralisables (robustes à de nouvelles configurations) pour notamment décrire la dynamique d'un écoulement. De nombreuses approches basées sur des régressions linéaires (décomposition modale dynamique, identification parcimonieuse de la dynamique) font le lien avec la théorie de Koopman et permettent aujourd'hui de déterminer des modèles réduits que l'on ne saurait pas dériver analytiquement.

Après cet état de l'art, on s'est posé une question pratique : est-il envisageable d'utiliser de l'information ponctuelle pour estimer un écoulement urbain et ainsi aider un drone autonome qui assurerait des missions de livraisons ou d'inspection ? Pour apporter des éléments de réponse, on s'est intéressé aux méthodes d'estimation d'écoulements présentés dans la littérature. On a repéré trois classes

de méthodes : l'estimation directe, l'estimation régressive et l'assimilation de données. Ce travail a montré qu'il existait pléthore d'approches pour estimer un écoulement mais qu'elles étaient peu comparées entre elles. De plus, les applications sont restées académiques et on ne peut pas statuer sur la réussite (ou non) des méthodes pour un écoulement urbain. On a donc défini les deux objectifs principaux de la thèse : 1) unifier les outils mathématiques pour réduire et estimer un écoulement ; 2) tester et comparer les méthodes sur des écoulements de complexité croissante.

La **deuxième** étape de la thèse a donc été d'unifier les outils de l'apprentissage automatique pour l'estimation d'un système dynamique. Ce travail méthodologique a été décomposé en trois parties. **Tout d'abord**, on s'est concentré sur la réduction (chapitre 2). On a formalisé l'auto-encodage d'un état de haute dimension pour déterminer un état *latent* de dimension plus faible. On s'est intéressé à différents auto-encodeurs neuronaux, pensés comme de simples extensions de l'analyse en composantes principales. L'innovation majeure repose sur l'étude de l'auto-encodeur variationnel qui utilise des transformations non linéaires et structure son espace latent pour gagner en robustesse lors du décodage. **Ensuite**, on s'est focalisé sur le problème de reconstruction (chapitre 3) qui consiste à estimer l'état latent du système à partir de mesures de l'état de haute dimension. Dans l'*estimation directe*, les mesures sont utilisées pour trouver la combinaison linéaire optimale des modes de décodage. Cette approche requérant une optimisation pour chaque nouvelle estimation, elle est surtout adaptée pour des tâches *offline*. Dans l'*estimation par régression*, on apprend complètement l'opération de passage entre l'espace des mesures et l'espace latent. Pour s'assurer que le modèle respecte le compromis entre son biais et sa variance, on a compris qu'il était nécessaire de faire une validation croisée de ses hyper-paramètres. On a conclu le chapitre en fournissant la carte d'identité de chacune des méthodes régressives. **Enfin**, on s'est intéressé au problème de prédiction (chapitre 4) pour estimer le futur de l'état latent à partir de l'état latent courant. Différentes approximations finies de l'opérateur de Koopman ont été présentées, l'objectif étant de faire une prédiction linéaire dans un espace d'observations plutôt qu'une prédiction non linéaire dans l'espace d'état. Toutes les approximations sont pensées comme des raffinements de la décomposition modale dynamique et on conclut le chapitre en proposant un algorithme pour apprendre automatiquement les observations de l'état à utiliser.

La **troisième** et dernière étape de la thèse a été d'implémenter et de tester les outils mentionnés ci-dessus pour estimer quatre écoulements de complexité croissante. L'obtention des différentes bases de données a été présentée succinctement (chapitre 5). On a retenu les configurations suivantes : l'écoulement laminaire autour d'un cylindre, un écoulement de couche de mélange, l'écoulement turbulent autour d'un cylindre rectangulaire et l'écoulement dans différents plans d'un écoulement urbain. La partie des résultats (chapitre 6) a été structurée en trois sections, en lien avec la partie méthodologique.

→ Sur la **réduction**, on a testé la décomposition en modes propres orthogonaux et trois auto-encodeurs neuronaux. En comparant les spectres des champs

de vitesse auto-encodés, on s'est aperçu que les réductions linéaires filtraient l'information haute fréquence, ce qui est cohérent étant donné que les modes sont déterminés par critère énergétique. Au contraire, la réduction non linéaire essaye de restituer l'intégralité du spectre mais souvent au détriment des premiers modes. Concernant la généralisation des modes appris à des données de test, du sur-apprentissage est notable sur l'écoulement autour de la tour. En étudiant l'impact du bruit sur l'état latent lors du décodage, on s'est aussi aperçu que la contrainte variationnelle permettait une meilleure robustesse.

→ Sur la **reconstruction**, on a testé l'estimation directe, l'estimation par régression linéaire en version mono-tâches et multi-tâches, par vecteurs supports, par réseau de neurones superficiel et par arbres de décisions boostés. Il est ressorti qu'une validation croisée des hyper-paramètres permettait des performances similaires pour chacune des méthodes régressives. Le sur-apprentissage était surtout notable pour l'écoulement autour de la tour, avec une différence entre les scores d'entraînement et de test qui pouvait atteindre 30%.

→ Sur la **prédiction**, on a testé la décomposition en modes dynamiques et sa version étendue à des observations quadratiques, des observations radiales, des observations retardées ou des observations automatiquement apprises. Tous les raffinements de la méthode de base ne semblent pas valoir le gain en complexité : dans chacun des cas, il est très difficile d'identifier des fonctions propres non fallacieuses de l'opérateur.

Tous ces résultats mettent en exergue que l'apprentissage automatique n'est pas encore mûr pour des configurations très complexes. De bonnes estimations ont été effectuées sur les trois premiers écoulements car les données disponibles permettaient de caractériser la dynamique avec peu de modes. Pour l'écoulement autour de la tour qui était l'application initialement visée, il y a deux points critiques sur lesquels il faut réfléchir plus en détail :

1. La dynamique est très instationnaire du fait des niveaux de turbulence. Pour espérer apprendre un modèle d'estimation généralisable, il est nécessaire d'augmenter drastiquement la base de données et d'utiliser des méthodes de régularisation élaborées du type génératives. Dans tous les cas, il faut définir un cahier des charges pour figer la fidélité nécessaire que le modèle doit restituer. À l'issu de cette thèse, le travail préliminaire sur la tour isolée montre qu'il est difficilement envisageable d'estimer son champ de vitesse (même à 80% d'énergie) en temps réel.
2. Dans une configuration pratique, le nombre de capteurs sera très limité. Les scores de reconstruction obtenus, pourtant mitigés, ont été obtenus avec un nombre déjà trop important de capteurs.

Ces deux points démotivent une modélisation orientée données de l'écoulement. Pour concevoir un drone autonome, il serait plus sage de chercher à renforcer son comportement à partir d'expériences.

7.2 | Perspectives

De nombreuses perspectives émergent de ce travail de thèse. On liste les principales ci-dessous.

→ Les modèles dynamiques qui ont été appris sont purement déterministes. On pourrait apprendre des modèles probabilistes, basés sur une approximation finie de l'opérateur de Perron Frobenius. Il s'agit de l'opérateur dual de l'opérateur de Koopman, qui fait avancer dans le temps *toutes* les densités de probabilités de l'état. Cette approche est développée dans l'algorithme *Cluster-Reduced Order Model* de Kaiser *et al.* (2014).

→ Les modèles ont été appris pour des paramètres fixés. Il serait plus pertinent d'apprendre des modèles couvrant un espace de paramètres. Pour ce faire, on peut utiliser des approches génératives comme décrit dans Morton *et al.* (2021). Les auteurs apprennent un modèle pour l'écoulement autour d'un cylindre en rotation et ce modèle est robuste au nombre de Reynolds et à la vitesse de rotation du cylindre. Un écoulement de couche de mélange forcée serait un bon cas test : selon les perturbations harmoniques en entrée du domaine, on chercherait à prédire l'appariement des vortex (Ko *et al.*, 2008).

→ Nos méthodes de réduction sont *globales*. Le champ de vitesse complet est caractérisé par un seul ensemble de modes. On pourrait envisager des réductions *locales* avec des raffinements dans les zones où l'écoulement est complexe. À titre d'exemple, Xiao *et al.* (2019) utilise une décomposition modale non intrusive locale pour réduire un écoulement urbain. On pense que cette méthode pourrait être étendue à des réseaux de neurones.

→ Le placement de capteurs a été purement heuristique. Dans une approche plus physique, il aurait fallu déterminer des capteurs à la fois observables (pour bien caractériser la dynamique) et contrôlables. Ce travail est formalisé par Manohar *et al.* (2018b). Pour appliquer cette méthode, il est nécessaire de disposer des clichés de la simulation adjointe.

→ Les reconstructions ont été effectuées avec des mesures homogènes. On pourrait essayer de faire des reconstructions avec des signaux non homogènes au champ à reconstruire, par exemple des signaux de pression. À titre d'exemple, Erichson *et al.* (2020) ont utilisé des signaux de pression pariétale pour reconstruire l'écoulement laminaire en aval d'un cylindre.

→ La réduction non linéaire variationnelle n'a pas de régularité temporelle. Il faudrait adjoindre une contrainte temporelle pour déterminer une variété latente sur laquelle l'état latent a une dynamique lisse. On peut envisager l'utilisation de structures récurrentes de type cellules *Long-Short Term Memory* (Gonzalez et Balajewicz, 2018).

→ Les reconstructions n'utilisaient que les mesures courantes. À la manière des extensions proposées pour l'estimation stochastique, on pourrait étendre nos modèles de reconstruction par régression à des fenêtres d'observations. Cette approche serait utile pour reconstruire des clichés PIV *a posteriori* à partir de signaux instationnaires (Deng *et al.*, 2019).

→ Les capteurs sont tous supposés fonctionnels. Pour une application pratique, il faudrait s'intéresser à la possibilité de capteurs faisant défaut (de Silva *et al.*, 2020).

→ Dans nos méthodes de réduction, l'ordre des composantes du vecteur vitesse n'importe pas. En imposant une structure spatiale i.e. en travaillant avec des images, on pourrait utiliser la puissance des auto-encodeurs convolutifs (Fukami *et al.*, 2020). Par ailleurs, on pourrait voir le problème de reconstruction comme un problème de *super-résolution*.

→ Dans ce manuscrit, la physique n'était pas la préoccupation majeure ; on s'est surtout focalisé sur les outils mathématiques. La prochaine étape est la prise en compte de lois physiques pour faire de l'intelligence artificielle de confiance. Une des pistes est de régulariser la fonction coût en imposant que la relation entrée - sortie apprise par le modèle respecte plus globalement les équations de Navier Stokes (Sun et Wang, 2020). On parle de *Physics Informed Neural Networks* (Raissi *et al.*, 2019).

→ Avec un auto-encodeur profond, on peut (potentiellement) réduire des données de grande dimension. Cette réduction s'obtient au prix d'une perte d'interprétabilité : un réseau de neurones avec beaucoup de couches cachés n'a pas d'expression analytique simple. Pour améliorer l'interprétabilité du réseau, on peut chercher à le *distiller* en effectuant une régression symbolique. Cranmer *et al.* (2020) suivent cette philosophie en combinant des *Graph Neural Network* et des arbres binaires évolués par algorithme génétique. On pourrait reprendre l'idée à nos cas mécanique des fluides, dans l'objectif d'apprendre un modèle physique interprétable pour la variable latente.

→ Les modèles d'estimation déployés reflètent uniquement la *fin* d'un apprentissage. Chaque nouveau modèle est ré-entraîné depuis le début sans tirer parti de *l'histoire* d'apprentissage d'autres modèles. Pour accélérer l'apprentissage de nouveaux modèles (par exemple pour une nouvelle configuration i.e. un nouveau nombre de Reynolds), on pourrait *transférer* les connaissances d'anciens apprentissages. On parle de *transfer learning* et une application en mécanique des fluides pourrait se révéler pertinente (Lee *et al.*, 2021).

Quatrième partie

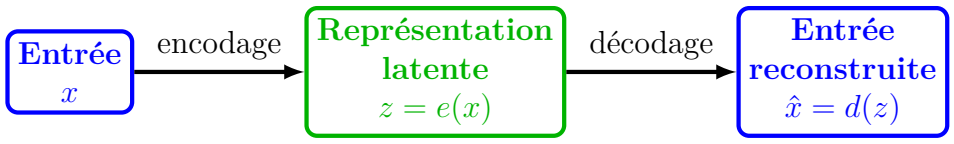
Annexes et bibliographie

A. Auto-encodeur variationnel

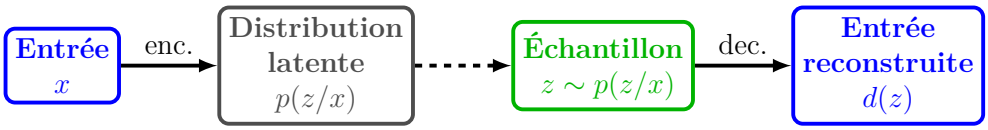
Dans cette annexe, on présente les détails mathématiques de l'auto-encodeur variationnel. Pour être consistant avec les notations de la littérature, on désigne par x l'état de haute dimension et z l'état latent (Odaibo, 2019).

Modèles des auto-encodeurs

Pour un auto-encodeur classique, l'encodeur e et le décodeur d sont des objets déterministes. Pour un auto-encodeur variationnel, l'encodeur et le décodeur sont des objets probabilistes (voir figure A.1)



(a) Modèle d'auto-encodeur. Les objets e et d sont déterministes i.e. un point donne un point.



(b) Modèle d'auto-encodeur variationnel. L'état encodé suit la distribution $p(z/x)$ (état latent sachant l'état complet). L'état décodé suit la distribution $p(x/z)$ (état complet sachant l'état latent).

Figure *A.1* – *Deux modèles d'auto-encodeurs.*

Si l'on connaît la distribution *a priori* de l'état latent $p(z)$, la distribution *a priori* de l'état complet $p(x)$ et l'encodeur $p(z/x)$, on peut calculer le décodeur avec la formule de Bayes :

$$p(z/x) = \frac{p(x/z)p(z)}{p(x)} = \frac{p(x/z)p(z)}{\int p(x/u)p(u)du}$$

Hypothèses

On suppose que l'état latent suit une gaussienne standard i.e. $p(z) \equiv \mathcal{N}(0, I_{r\times r})$. On suppose aussi que l'état décodé suit une gaussienne de moyenne $d(z)$ et d'écart type c i.e. $p(x/z) \equiv \mathcal{N}(d(z), c^2 I_{n\times n})$. La fonction d est déterministe et définit le décodeur.

Inférence variationnelle

En inférence *bayésienne*, on aurait calculé $p(z/x)$ à partir des distributions $p(z)$ et $p(x/z)$. Ce calcul étant difficile (principalement à cause de l'intégration), on approche plutôt la distribution $p(z/x)$ par inférence *variationnelle* :

1. On choisit une famille de distributions, par exemple les gaussiennes.
2. On détermine la *meilleure* approximation de la distribution cible dans la famille sélectionnée. Les paramètres optimaux minimisent l'erreur entre la distribution réelle et la distribution approximée. Cette erreur est par exemple la divergence de Kullback Leibler.

On approxime donc $p(z/x)$ par une gaussienne $q_x(z)$ avec $q_x(z) \equiv \mathcal{N}\left(g(x), h^2(x)\right)$. Les fonctions g et h appartiennent respectivement aux ensembles G et H (formes de fonction) et définissent la moyenne et l'écart type de l'état encodé. Ces fonctions sont solutions du problème :

$$\boxed{(g,h) = \underset{(g,h)\times G\times H}{\arg\min}\ D_{\mathrm{KL}}\left(q_x(z)\ ||\ p(z/x)\right)}$$

où la divergence de Kullback Leibler est définie en théorie de l'information et se calcule, pour les distributions P et Q, selon :

$$D_{\mathbf{KL}}\left(P||Q\right) = \int_{\infty}^{\infty} P(u)\log\left(\frac{P(u)}{Q(u)}\right)du = \mathbb{E}\left[\log\left(\frac{P(u)}{Q(u)}\right)\right]$$

Lorsque la divergence est nulle, les distributions P et Q sont parfaitement identiques. Lorsque la divergence tend vers ∞, les distributions sont très différentes. Si l'on revient à notre problème, l'objectif est de minimiser la divergence KL entre la distribution réelle $p(z/x)$ et la gaussienne qui l'approxime $q_x(z)$. En utilisant la formule de Bayes et les propriétés du logarithme, on parvient à :

$$(g,h) = \underset{(g,h)\in G\times H}{\arg\min}\left\{\mathbb{E}_{z\sim q_x}\left(\log q_x[z]\right) - \mathbb{E}_{z\sim q_x}\left(\log\left[\frac{p(x/z)p(z)}{p(x)}\right]\right)\right\}$$

Avec quelques manipulations, on peut faire apparaître la divergence KL entre la distribution approchée et la distribution *a priori* de l'état latent, modélisée par une gaussienne standard :

$$\boxed{(g,h) = \underset{(g,h)\in G\times H}{\arg\max}\left\{\mathbb{E}_{z\sim q_x}\left(\log p[x/z]\right) - D_{\mathrm{KL}}\left(q_x[z]\ ||\ p[z]\right)\right\}}$$

L'approximation optimale de la distribution *a posteriori* de l'état latent $p(z/x)$ nécessite donc de faire en compromis :

— Les paramètres g et h doivent maximiser la vraisemblance [1] des données (termes de gauche).

1. À ne pas confondre avec la densité. Supposons que la variable aléatoire A suive une loi gaussienne de moyenne μ et de variance σ^2. Dans ce cas, la densité de probabilité est $p(a/\mu,\sigma) =$

— Les paramètres g et h doivent assurer un encodage proche de la distribution *a priori* de l'état latent (termes de droite).

Optimisation du décodeur

La distribution $p(z/x)$ est approximée par inférence variationnelle depuis $p(z)$ et $p(x/z)$. Les leviers restants pour optimiser le schéma encodeur/décodeur sont les paramètres e et c du décodeur. Pour une entrée x, on souhaite maximiser la probabilité d'avoir $\hat{x} = x$ en échantillonnant z de la distribution $q_x^*(z)$ puis en échantillonnant $\hat{x}$ de la distribution $p(x/z)$. Cela se traduit par :

$$(d, c) = \arg\max_{d \in D} \mathbb{E}_{z \sim q_x^*} \left(\log p[x/z] \right)$$

En ayant choisi de modéliser la distribution de x sachant z par une gaussienne, on peut expliciter :

$$d = \arg\max_{d \in D} \mathbb{E}_{z \sim q_x^*} \left(-\frac{||x - d(z)||^2}{2c} \right)$$

Toutes pièces ensemble

Finalement, l'auto-encodeur variationnel est construit en résolvant le problème suivant :

$$(d, g, h) = \arg\max_{d,g,h \in D,G,H} \left\{ \mathbb{E}_{z \sim q_x} \left(-\frac{||x - d(z)||^2}{2c} \right) - D_{\mathrm{KL}} \left(q_x[z] \;||\; p[z] \right) \right\}$$

Cette formulation fait clairement apparaître l'erreur d'auto-encodage (terme de gauche) et la régularisation de l'espace latent. Pour le travail du manuscrit, il reste à dire que les fonctions d, g et h sont des réseaux de neurones. La fonction coût précisée dans le chapitre 2 est obtenue en explicitant la divergence à partir de manipulations sur les gaussiennes.

$\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2} \left(\frac{a - \mu}{\sigma} \right)^2 \right]$. Ici, les paramètres sont fixés et on peut calculer la probabilité que a soit dans un intervalle donné. Supposons maintenant que l'on dispose de données d'observations $\{a\}_i$. On sait que ces données sont échantillonnées d'une gaussienne mais on ne connaît pas les paramètres. La vraisemblance pour a_i est définie par $L(\mu, \sigma / a_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2} \left(\frac{a_i - \mu}{\sigma} \right)^2 \right]$. Ici, les paramètres sont les variables. On cherche les paramètres qui maximisent la vraisemblance des données. Dans le cas d'une gaussienne, la solution est simple : la moyenne μ et l'écart type σ sont les moyennes et écart type empiriques. De cette manière, si on refait des échantillons de loi appris, on retrouvera vraisemblablement nos données.

B. Éléments de mécanique des fluides numériques

Dans cette annexe, on donne quelques éléments de compréhension pour résoudre une équation de transport avec des *volumes finis*. On précise également les schémas numériques pour les quatre écoulements étudiés.

Principe des volumes finis

Soit une quantité scalaire ϕ transportée par un écoulement dont le champ de vitesse est $\boldsymbol{V}$. La forme locale de conservation de ϕ s'écrit :

$$\frac{\partial \rho\phi}{\partial t} + \boldsymbol{\nabla} \cdot [\rho\phi \boldsymbol{V}] - \boldsymbol{\nabla} \cdot [\rho\Gamma \boldsymbol{\nabla}\phi] = S(\phi)$$

où ρ est la masse volumique, Γ est le coefficient de diffusion et S est un terme source. Cette équation traduit que la variation instantanée de ϕ est équilibrée lors du transport par des échanges convectifs et diffusifs avec le milieu extérieur. Pour résoudre cette équation dans une approche *volumes finis*, on commence par discrétiser le domaine en mailles élémentaires de volume $\mathcal{V}$ et de frontière $\mathcal{S}$. En intégrant l'équation de conservation sur une maille et en appliquant le théorème de la divergence [1], on peut écrire :

$$\int_{\mathcal{V}} \frac{\partial \rho\phi}{\partial t} d\mathcal{V} + \int_{\mathcal{S}} \rho\phi \boldsymbol{V} \cdot \boldsymbol{n} d\mathcal{S} - \int_{\mathcal{S}} \rho\Gamma \boldsymbol{\nabla}\phi \cdot \boldsymbol{n} d\mathcal{S} = \int_{\mathcal{V}} S(\phi) d\mathcal{V}$$

avec $\boldsymbol{n}$ la normale à la surface, pointant vers l'extérieur. Une maille étant discrète (e.g. cubique ou polyédrique), les intégrales sont transformées en des sommes finies. En calculant les intégrales et en faisant le bilan des flux pour toutes les mailles, on parvient à un système linéaire $\mathcal{M}_t \Phi_{t+1} = \mathcal{B}_t$ où Φ_{t+1} contient les quantités au centre des mailles à l'instant $t+1$, $\mathcal{M}_t$ contient sur les flux calculés avec les grandeurs à l'instant t et $\mathcal{B}_t$ contient les termes sources.

Flux convectifs et diffusifs

La maille de centre P possède M voisins, définissant ainsi $\mathcal{S}_1, ..., \mathcal{S}_M$ surfaces de contact.

1. Intégrales dans le volume écrites en intégrales de surface. Intuitivement, on écrit que la variation totale au niveau de frontière est égale à la somme des petites variations au sein du volume. C'est le théorème de Stokes, expliqué de manière très intuitive à l'adresse suivante : `https://www.youtube.com/watch?v=1lGM5DEdMaw`.

Pour le terme **convectif**, l'intégrale se réécrit :

$$\int_{\mathcal{S}} \rho\phi \boldsymbol{V} \cdot \boldsymbol{n} d\mathcal{S} = \sum_{i=1}^{M} \int_{\mathcal{S}_i} \rho\phi_i \boldsymbol{V}_i \cdot \boldsymbol{n}_i d\mathcal{S}_i$$

où ϕ_i et $\boldsymbol{V}_i$ sont respectivement la quantité ϕ et la vitesse $\boldsymbol{V}$ *sur* la surface i de normale extérieure $\boldsymbol{n}_i$. Pour une méthode *volumes finies* du deuxième ordre, on peut montrer que l'intégrale se résume à l'information **au centre de la cellule** (Jasak, 1996). Ainsi, on a :

$$\sum_{i=1}^{M} \int_{\mathcal{S}_i} \rho\phi_i \boldsymbol{V}_i \cdot \boldsymbol{n}_i d\mathcal{S}_i \approx \sum_{i=1}^{M} \rho\phi_{f_i} \boldsymbol{V}_{f_i} \cdot \boldsymbol{n}_{f_i} \mathcal{S}_i$$

où l'indice f précise que la quantité est prise au centre de la surface. En notant F_{f_i} le débit massique ($kg.s^{-1}$) à travers la surface, on parvient donc à :

$$\boxed{\int_{\mathcal{S}} \rho\phi \boldsymbol{V} \cdot \boldsymbol{n} d\mathcal{S} \approx \sum_{i=1}^{M} \phi_{f_i} F_{f_i}}$$

Les valeurs aux centre des surfaces sont calculées à partir des informations aux centres des mailles voisines via des *schémas d'interpolation*. Les flux F_{f_i} sont calculés à partir des informations au temps t tandis que la quantité ϕ_{f_i} est l'inconnue au temps $t + 1$. Parmi les choix classiques, on peut citer les interpolations :

- **linear**. La quantité ϕ_f est la moyenne entre ϕ_P et ϕ_N où N est la maille suivant P.
- **upwind differencing**. La quantité ϕ_f prend la valeur ϕ_P *ou* ϕ_N selon la direction du flux.
- **linear upwind differencing**. La variation entre le centre d'une maille et le centre de la face est linéaire. L'interpolation s'appuie sur un calcul de gradient dont la contribution est contrôlée par des *limiteurs*.

Pour le terme **diffusif**, il faut discrétiser :

$$\boxed{\int_{\mathcal{S}} \rho\Gamma \boldsymbol{\nabla}\phi \cdot \boldsymbol{n} d\mathcal{S} = \sum_{i=1}^{M} \rho\Gamma_i \left[\boldsymbol{\nabla}\phi_i \cdot \boldsymbol{n}_i\right] \mathcal{S}_i}$$

On ne connaît pas le gradient de ϕ sur la face i donc le calcul n'est pas trivial. Deux configurations se présentent :

1. Si les cellules sont *orthogonales*, on peut écrire :

$$\boldsymbol{\nabla}\phi_i \cdot \boldsymbol{n}_i = \frac{\phi_P - \phi_N}{|\boldsymbol{d}|} |\boldsymbol{\Delta}_i|$$

où ϕ_P est l'information au centre de la cellule courante P, ϕ_N est l'information au centre de la cellule voisine N, $\boldsymbol{d}$ est le vecteur reliant les centres des

cellules P et N et $\boldsymbol{\Delta}$ est le vecteur reliant le centre de la face et le centre de la cellule N.

2. Si les cellules ne sont *pas orthogonales* alors la normale à la face n'est pas alignée avec $\boldsymbol{d}$. On introduit donc le vecteur complémentaire $\boldsymbol{k}$ tel que $\boldsymbol{n} = \boldsymbol{\Delta} + \boldsymbol{k}$. On peut écrire :

$$\sum_{i=1}^{M} \Gamma_i \left[\boldsymbol{\nabla}\phi_i \cdot \boldsymbol{n}_i\right] \mathcal{S}_i = \sum_{i=1}^{M} \Gamma_i \left[\boldsymbol{\nabla}\phi_i \cdot \boldsymbol{\Delta}_i\right] \mathcal{S}_i + \sum_{i=1}^{M} \Gamma_i \left[\boldsymbol{\nabla}\phi_i \cdot \boldsymbol{k}_i\right] \mathcal{S}_i$$

Le terme avec $\boldsymbol{\Delta}_i$ est la composante orthogonale, traitée comme le cas 1 de manière implicite (dans la matrice $\mathcal{M}_t$). Le terme avec $\boldsymbol{k}_i$ est la composante non orthogonale et son traitement est explicite (dans la matrice $\mathcal{B}_t$). Pour des questions de stabilité, la valeur de cette composante est limitée à une fraction du terme implicite.

Sur les équations de Navier Stokes

Pour un écoulement incompressible de fluide newtonien et sans forces de volume, les équations de Navier Stokes s'écrivent :

$$\begin{cases} \boldsymbol{\nabla} \cdot \boldsymbol{V} = 0 \\ \dfrac{\partial \boldsymbol{V}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{V} \otimes \boldsymbol{V}) = -\dfrac{1}{\rho}\boldsymbol{\nabla} p + \nu \Delta \boldsymbol{V} \end{cases}$$

où $\boldsymbol{V}$ est la vitesse, p est la pression, ν est la viscosité cinématique et Δ est le laplacien. Les inconnues sont les trois composantes de la vitesse et la pression **mais** on ne dispose pas d'équation explicite pour la pression. L'équation de continuité est surtout une restriction du champ de vitesse calculé par l'équation de quantité de mouvement. La boucle PISO pour *Pressure-Implicit with Splitting Operators* permet de résoudre le couplage pression - vitesse par un calcul itératif (Issa, 1986) :

1. **Résoudre l'équation de quantité de mouvement**. Il s'agit de résoudre $\mathcal{M}U = P$ où U contient les vitesses aux centres des cellules (inconnues), $\mathcal{M}$ contient les flux et P est la discrétisation de l'opposé du gradient de pression.
2. **Itérations jusqu'à ce que la pression converge**.
   (a) Calcul de $H = AU - \mathcal{M}U$ où $A = \text{diagonale}(\mathcal{M})$.
   (b) Résolution de $\boldsymbol{\nabla} \cdot (A^{-1}P) = \boldsymbol{\nabla} \cdot (A^{-1}H)$. C'est l'équation de Poisson pour la pression, dérivée à partir de la condition d'incompressibilité.
   (c) Correction de la vitesse via $U = A^{-1}H - A^{-1}P$

Pour un maillage avec des mailles non orthogonales, on rajoute des itérations supplémentaires sur l'équation de pression 2b). Si la boucle PISO a été pensée pour des écoulements incompressibles instationnaires, il existe d'autres stratégies de couplage :

— **SIMPLE** pour *Semi-Implicit Method for Pressure Linked Equations*. Contrairement à la boucle PISO, la résolution de l'équation de quantité de mouvement est inclus dans les itérations. Cet algorithme est utilisé pour des cas stationnaires afin d'améliorer la stabilité du calcul [2].
— **PIMPLE** qui mélange les procédures PISO et SIMPLE. Cela permet de faire des calculs transitoires.

Attention

On utilise l'algorithme PISO pour l'écoulement laminaire autour du cylindre et la couche de mélange spatiale. On utilise l'algorithme SIMPLE pour l'écoulement turbulent autour du cylindre et l'écoulement autour de la tour.

Exemple de calcul CFD

On souhaite résoudre les équations de Navier Stokes *moyennées* avec un modèle de turbulence par équations de transport. Le schéma classique de résolution est le suivant :

1. On précise les conditions initiales des champs moyens (vitesse et pression), les conditions initiales des champs des quantités turbulentes (*e.g* énergie cinétique turbulente et dissipation) et les conditions limites.
2. Itérations en temps :
   (a) Calcul de la viscosité turbulente à partir des champs turbulents de l'itération précédente.
   (b) Calcul des flux et formation de la matrice des coefficients $\mathcal{M}$.
   (c) Algorithme qui résout le couplage pression - vitesse pour avoir le champ de vitesse moyenne et le champ de pression moyenne.
   (d) Résolution des équations de transport des quantités turbulentes par volumes finis.

Schémas numériques OpenFOAM

Concernant le terme temporel, on utilise le schéma *backward* pour les trois premiers écoulements. Il s'agit d'un schéma d'ordre deux implicite. Pour l'écoulement de la tour, on utilise *euler* c'est-à-dire un schéma d'ordre un implicite. Pour les dérivées spatiales, on utilise pour tous les écoulements :
— Une discrétisation des divergences par *linear upwind*.
— Une discrétisation des gradients par *linear* avec limiteurs.
— Une discrétisation des laplaciens par *linear* avec limiteurs.
— Une interpolation *linear* des centres des cellules vers les centres des faces.

2. Avec le terme temporal absent de la matrice $\mathcal{M}$, la diagonale n'est plus dominante et la matrice $\mathcal{M}$ est moins bien conditionnée.

On renvoie directement à la documentation OpenFOAM pour l'expression détaillée des schémas et leur propriétés [3]. Cette thèse n'étant pas numérique, on a suivi les recommandations par défaut.

Calcul RANS $k-\omega$ SST

On résout les équations de Navier-Stokes moyennées et on modélise toute la turbulence par une viscosité turbulente ν_t. Pour la calculer, OpenFOAM résout des équations de transports pour k (énergie cinétique turbulente) et ω (taux de dissipation spécifique) précisées dans Menter *et al.* (2003). L'équation pour ω fait intervenir une fonction de mélange F_1 permettant de basculer entre un fonctionnement $k-\omega$ en proche paroi et un fonctionnement $k-\epsilon$ dans l'écoulement cisaillé libre [4].

L'équation pour l'énergie cinétique turbulente k fait intervenir la viscosité moléculaire dynamique μ, le champ de vitesse moyenne $\overline{\boldsymbol{V}}$, la viscosité dynamique turbulente μ_t, la production d'énergie cinétique turbulente P_k, la masse volumique ρ et le taux de dissipation de l'énergie ϵ.

$$\frac{\partial \rho k}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \overline{\boldsymbol{V}} k) = \boldsymbol{\nabla} \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \boldsymbol{\nabla} k \right) + P_k - \rho\epsilon$$

L'équation pour le taux de dissipation d'énergie spécifique ω fait intervenir les coefficients $(\sigma_{\omega,2}, \beta, \gamma)$ et la fonction de mélange F_1 tels que :

$$\begin{aligned}\frac{\partial \rho \omega}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \overline{\boldsymbol{V}} \omega) = \boldsymbol{\nabla} \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \boldsymbol{\nabla} \omega \right) \\ + \frac{\gamma}{\nu_t} P_k - \beta \rho \omega^2 + 2(1 - F_1) \frac{\rho \sigma_{\omega,2}}{\omega} \boldsymbol{\nabla} k : \boldsymbol{\nabla} \omega\end{aligned}$$

Pour des détails supplémentaires, on renvoie à l'article de Menter *et al.* (2003) et les références en bas de page [5].

Calcul LES - smagorinsky

On résout les équations de Navier-Stokes filtrées et on modélise l'effet des échelles non résolues sur les échelles résolues via un modèle sous maille. Ce modèle est construit sur une viscosité sous maille ν_{sgs} telle que :

$$\nu_{\text{sgs}} = C_k \Delta_m k^{0.5}$$

3. Voir `https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes.html`

4. Les fonctions d'amortissement du modèle $k-\epsilon$ ne sont pas pertinentes en proche paroi, menant à de mauvaises prédictions d'écoulements décollés. Au contraire, le modèle $k-\omega$ est adapté pour prédire de faibles décollements mais les résultats sont très sensibles à la turbulence *free stream*. Le modèle $k-\omega$ SST permet de tirer les bénéfices des deux modèles.

5. Voir la page *OpenFOAM* `https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html` et l'excellente vidéo `https://www.youtube.com/watch?v=myv-ityFnS4`.

où Δ_m est la taille de la maille et k est l'énergie cinétique turbulente solution de l'équation quadratique :

$$ak^2 + bk + c = 0$$

Les coefficients a, b et c sont construits sur le tenseur de déformation du champ de vitesse filtré et donnés par :

$$\begin{cases} a = \dfrac{C_e}{\Delta} \\ b = \dfrac{2}{3} \times \text{trace}(\boldsymbol{S}) \\ c = 2C_k\Delta \times \text{deviateur}(\boldsymbol{S}) : \boldsymbol{S} \end{cases}$$

avec $\boldsymbol{S} = \dfrac{1}{2}\left(\boldsymbol{\nabla}\tilde{V} + [\boldsymbol{\nabla}\tilde{\boldsymbol{V}}]^T\right)$ où $\tilde{.}$ est l'opération de filtrage. Les coefficients du modèle sont $C_k = 0.094$ et $C_e = 1.048$ (Smagorinsky, 1963).

C. Calcul numérique de l'écoulement urbain

La simulation des grandes échelles autour de la tour isolée est basée sur les travaux de thèse de Sheng (2017). Le calcul utilise des profils Eurocode pour générer numériquement la turbulence. Dans cette annexe, on résume le dispositif expérimental et quelques résultats de validation.

Détails de l'expérience

La maquette est un parallélépipède à base carrée de diamètre $D = 10$cm et de hauteur $H = 49$cm. Elle est plongée dans un écoulement de vitesse $U_\infty = 10\ m.s^{-1}$. On définit la hauteur de demi-veine $H_v = 50$ cm. Le nombre de Reynolds est 3×10^5. Des générateurs de turbulence triangulaires (voir figure C.1) permettent de produire un profil de vitesse caractéristique d'une couche limite atmosphérique. Avec une longueur de rugosité (échelle maquette) $z_0 = 0.02/300$ m et une hauteur minimale (échelle maquette) $z_{\min} = 0.004$ m, Sheng (2017) montre un bon accord avec le profil moyen de l'Eurocode :

$$\text{Profil moyen} \rightarrow \begin{cases} \bar{U}(z) = U_\infty \times 0.19 \left(\dfrac{z_0}{0.05}\right)^{0.07} \ln\left(\dfrac{z}{z_0}\right) \text{ pour } z \leq z_{\min} \\ \bar{U}(z) = \bar{U}(z_{\min}) \text{ pour } z < z_{\min} \end{cases}$$

et avec les profils d'intensité turbulente [1] pour les trois composantes :

$$\text{Intensité turbulente selon } x \rightarrow \begin{cases} I_u(z) = \dfrac{1}{\ln\left(\dfrac{z}{z_0}\right)} \text{ pour } z \leq z_{\min} \\ I_u(z) = I_u(z_{\min}) \text{ pour } z < z_{\min} \end{cases}$$

$$\text{Intensité turbulente selon } y \rightarrow \begin{cases} I_v(z) = \dfrac{0.88}{\ln\left(\dfrac{z}{2.5 \times 10^{-5}}\right)} \text{ pour } z \leq z_{\min} \\ I_v(z) = I_v(z_{\min}) \text{ pour } z < z_{\min} \end{cases}$$

$$\text{Intensité turbulente selon } z \rightarrow \quad I_w(z) = 0.08$$

Sheng (2017) détermine aussi les échelles de longueur de Von Karman suivantes :

1. Les fluctuations u suivent une loi normale centrée d'écart type $I_u \times \bar{U}$.

$$\text{Échelle pour } u, \text{ selon } x \rightarrow \begin{cases} L_u(z) = 0.7955 \times z^{0.3365} \text{ pour } 0 < z < H \\ L_u(z) = 0.7955 \times H^{0.3365} \text{ sinon} \end{cases}$$

$$\text{Échelle pour } v, \text{ selon } x \rightarrow \begin{cases} L_v(z) = 0.3204 \times z^{0.3694} \text{ pour } 0 < z < H \\ L_v(z) = 0.3204 \times H^{0.3694} \text{ sinon} \end{cases}$$

$$\text{Échelle pour } w, \text{ selon } x \rightarrow \begin{cases} L_w(z) = 0.2339 \times z^{0.8289} \text{ pour } 0 < z < H \\ L_w(z) = 0.2339 \times H^{0.8289} \text{ sinon} \end{cases}$$

ce qui permet de construire la densité spectrale de puissance pour des rafales (*gusts* en anglais) continues (Beal, 1993). Pour les fluctuations selon x, on a par exemple :

$$S_{u_g}(\Omega) = \sigma_u^2 \frac{2L_u}{\pi} \frac{1}{(1 + (1.339 L_u \Omega)^2)^{5/6}}$$

où σ_u est l'écart type des fluctuations longitudinales et Ω est la fréquence spatiale. À noter que l'on retrouve de façon déguisée $\Omega^{-5/3}$.

(a) Maquette de la tour avec prises de pression.

(b) Générateurs de vortex en entrée de veine.

Figure *C.1* – *Photos de la campagne expérimentale à la soufflerie NSA du CSTB de Nantes (Sheng* et al.*, 2018).*

Vortex method

Les profils identifiés expérimentalement sont utilisés pour générer la turbulence en entrée du domaine de calcul avec une *vortex method* (Mathey *et al.*, 2006). En *amont* du calcul, on prépare l'évolution de la condition d'entrée sur l'intervalle du temps de simulation. La méthode commence par créer N_v tourbillons situés aléatoirement dans le plan d'entrée. À chaque tourbillon est associée une taille,

une circulation, une durée de vie, un sens et une longueur de déplacement qui dépendent de sa position. On calcule alors les fluctuations de vitesse induites par ces tourbillons à chaque instant. Lorsqu'un tourbillon atteint la fin de sa durée de vie, on réinitialise ses caractéristiques.

H. Toubin a implémenté une *vortex method* améliorée (Xie, 2016) avec python. La méthode introduit quatre paramètres (C_1, C_2, C_3, C_4) à optimiser en fonction du cas étudié. Les différentes étapes sont les suivantes :

1. On choisit aléatoirement N_v centres de tourbillons dans le plan d'entrée. Le tourbillon i a pour centre $(x_i = 0, y_i, z_i)$
2. On calcule les grandeurs turbulentes au niveau de chaque tourbillon, à partir des profils identifiés expérimentalement.

$$\text{Énergie cinétique turbulente} \rightarrow k_i = \frac{3}{2} U_\infty I_u(z_i)^2$$

$$\text{Taux dissipiation d'énergie} \rightarrow \epsilon_i = C_\mu^{3/4} \frac{k_i^{3/2}}{\sqrt{L_u(z_i)^2 + L_v(z_i)^2 + L_w(z_i)^2}}$$

avec $C_\mu = 0.09$.

3. À chaque tourbillon, on assigne :
   (a) Une taille σ_i telle que :
   $$\sigma_i = C_1 \times \sqrt{L_u(z_i)^2 + L_v(z_i)^2 + L_w(z_i)^2}$$
   (b) Une circulation Γ_i telle que :
   $$\Gamma_i = C_2 \times 2\bar{U}(z_i) I_u(z_i) \sqrt{\frac{\pi S}{3N_v(2\ln(3) - 3\ln(2))}}$$
   avec S l'aire du plan d'entrée qui vaut 1 m^2.
   (c) Une durée de vie τ_i telle que :
   $$\tau_i = C_3 \times \frac{k_i}{\epsilon_i}$$
   (d) Un sens. C'est un nombre aléatoire choisi dans $\{-1; 1\}$ qui multipliera Γ_i dans le calcul des fluctuations de vitesse.
4. À chaque pas de temps, on calcule les fluctuations de vitesse induite en utilisant la loi de Biot-Savart. Ainsi, en chaque point $\boldsymbol{x} = (x, y, z)$, le vecteur des vitesse induites $\boldsymbol{v'}$ est donné par :

$$\boldsymbol{v}'(\boldsymbol{x}) = \frac{1}{2\pi}\sum_{i=1}^{N_v} \pm\Gamma_i \frac{(\boldsymbol{x}_i - \boldsymbol{x}) \times \boldsymbol{e}_1}{||\boldsymbol{x}_i - \boldsymbol{x}||_2^2} \left(1 - \exp\left[-\frac{||\boldsymbol{x}_i - \boldsymbol{x}||_2^2}{2\sigma_i^2}\right]\right) \exp\left[-\frac{||\boldsymbol{x}_i - \boldsymbol{x}||_2^2}{2\sigma_i^2}\right]$$

où $\boldsymbol{e}_1$ est le vecteur unitaire dans la direction longitudinale.

5. On déplace les tourbillons de façon aléatoire : on choisit un angle α entre 0 et 2π et on déplace selon :

$$\begin{cases} y_i \leftarrow y_i + C_4 \times \sigma_i \sin(\alpha) \\ x_i \leftarrow x_i + C_4 \times \sigma_i \cos(\alpha) \end{cases}$$

Si le tourbillon atteint une des parois du plan d'entrée, il *rebondit* : on change l'angle α de $\pi/2$ ou π.

6. Lorsqu'un tourbillon atteint la fin de sa durée de vie, on réinitialise sa position, ses grandeurs turbulentes, sa taille, sa circulation, sa durée de vie et son sens.

Validation de la méthode

H. Toubin a réalisé plusieurs simulations de la veine vide pour optimiser les paramètres de la *vortex method*. Partant des valeurs proposées par Xie (2016) pour la plaque plane, la circulation et la durée de vie ont été augmentées jusqu'à atteindre des niveaux d'intensité turbulente satisfaisants. Les valeurs retenues sont dans le tableau C.1.

Nombre vortex	Taille	Circulation	Durée de vie	Déplacement
$N_v = 200$	$C_1 = 0.0125$	$C_2 = 2.8$	$C_3 = 40$	$C_4 = 0.0625$

Tableau C.1 – *Paramètres de la vortex method.*

On commence par vérifier que le profil moyen correspond bien au profil obtenu expérimentalement, de forme logarithmique (voir C.2). Le profil est quasi-identique en entrée du domaine ($x = -2m$), là où sont générés les vortex) mais il est un peu déformé à l'endroit où la tour sera positionnée ($x = 0m$). On regarde ensuite les profils d'intensité turbulente dans les différentes directions. Avec la figure C.3, on s'aperçoit que les niveaux de turbulence sont sous-estimés. En particulier, pour la composante longitudinale, on observe un déficit important pour des hauteurs supérieures à 1.5 fois la hauteur de la tour [2]. Enfin, on s'intéresse aux corrélations dans la direction latérale, calculées à l'aide de sondes placées sur une ligne $x = 0$ et $z = H$. Par exemple pour la composante de vitesse longitudinale, on a :

$$R_u(\Delta y) = \frac{\text{covariance}\left[U(y_0), U(y_0 + \Delta y)\right]}{\text{variance}\left[U(y_0)\right]}$$

2. Les résultats de Sheng (2017) avec des fluctuations gaussiennes ne sont pas meilleures. Pour la *vortex method*, on a au moins une bonne estimation de I_u sur la hauteur de la tour.

où la covariance et la variance sont calculées en faisant varier y_0. On observe sur la figure C.4 que les corrélations sont assez proches des résultats expérimentaux.

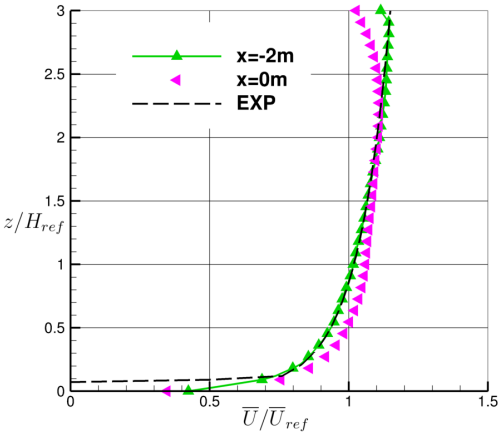


Figure *C.2* – *Comparaison du profil de vitesse moyenne avec le profil expérimental. L'abscisse $x = -2$ m correspond à l'entrée du domaine et l'abscisse $x = 0$ m correspond à l'emplacement où la tour sera placée.*

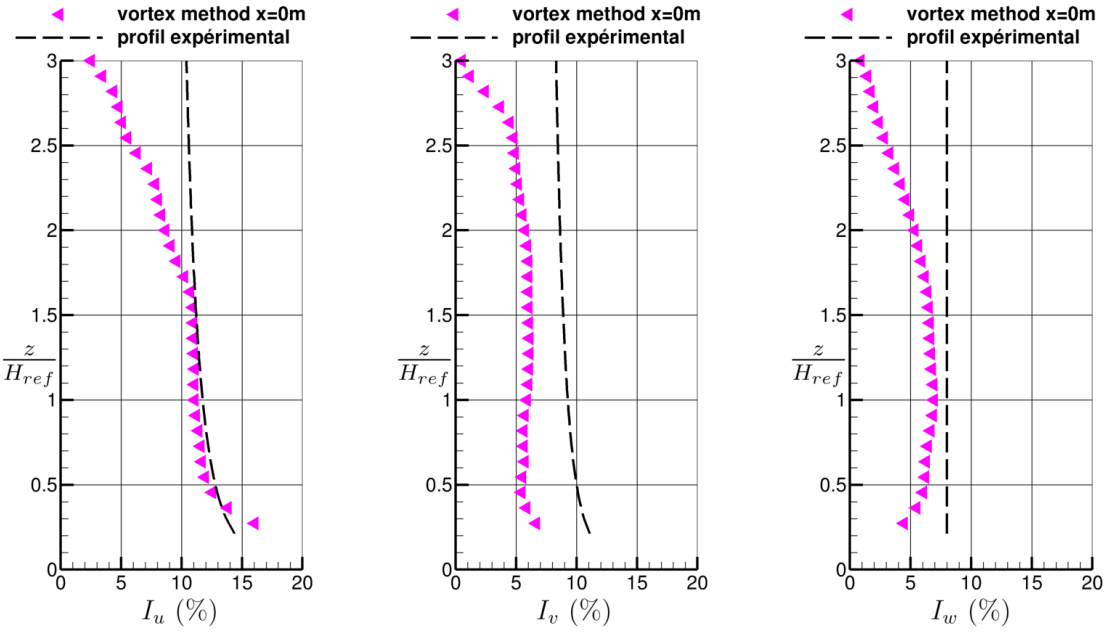
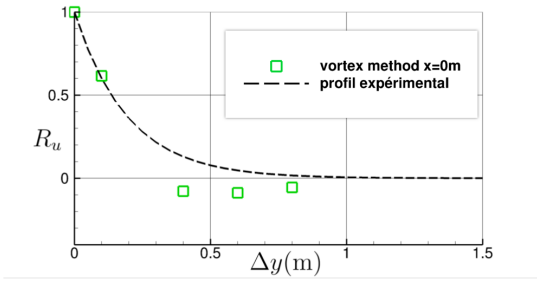
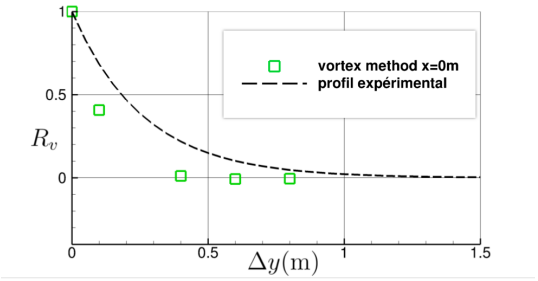


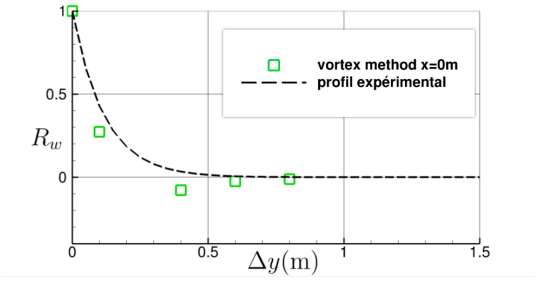
Figure *C.3* – *Profil d'intensités turbulentes numériques et expérimentaux à l'emplacement où se situera la tour.*



(a) $R_u(\Delta y)$



(b) $R_v(\Delta y)$



(c) $R_w(\Delta y)$

Figure *C.4* – *Corrélations spatiales pour les trois composantes de vitesse, dans la direction latérale.*

Critiques

En veine vide, la simulation des grandes échelles avec génération de turbulence par *vortex method* donne des résultats corrects. Certaines critiques sont toutefois à faire :

- La simulation est *peu* validée. Les données expérimentales ont été mises à disposition après le départ de la post doctorante. Dans le cadre de cette thèse, nous n'avons pas passé plus de temps sur la simulation car ce n'était pas l'objectif.
- La méthode de génération des vortex est peu pratique en réalité car chaque simulation requiert de stocker un grand volume de données.
- Un post doctorant est en train de reprendre la simulation. Un regard beaucoup plus critique sera fait sur la validation de la *vortex method*. Les méthodes d'apprentissage automatique seront alors testés dans le volume mais il y a une étape d'interpolation vers un maillage plus lâche.

D. Erreurs d'auto-encodage

Cette annexe donne le tableau des erreurs moyennes normalisées d'auto-encodage pour les écoulements considérés. On peut formuler les commentaires suivants :

1. La réduction non linéaire variationnelle a l'erreur moyenne de réduction la plus élevée. Ce résultat n'est pas étonnant. D'une part, parce que la détermination des structures est basée sur un critère énergétique (moindres carrés) *et* un critère de régularité (divergence de Kullback Leibler) alors que l'erreur normalisée moyenne ne regarde que le premier critère. D'autre part, parce que les hyper-paramètres du réseau ne sont pas optimisés.
2. En augmentant la dimension de l'espace latent, l'erreur d'auto-encodage est diminuée.
3. Pour les méthodes de réduction linéaires, les erreurs d'auto-encodage sont similaires. Ce n'est pas étonnant étant donné que les modes déterminés génèrent le même sous espace.
4. Le sur-apprentissage est notable sur les réductions VAE des trois premières configurations. Pour l'écoulement autour de la tour, le sur-apprentissage est important pour toutes les méthodes de réduction.

Méthode	**Écoulement**	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$ (cas 0) or $r_{0.95}$
POD	0	[12.88 12.6]	[1.34 1.32]	[0.12 0.12]
	1	[22.79 23.5]	[8.88 9.59]	[2.4 3.13]
	2	[45.05 43.26]	[29.14 31.17]	[9.37 15.4]
LAE	0	[13. 12.68]	[1.34 1.32]	[0.12 0.12]
	1	[22.83 23.57]	[9.22 9.95]	[2.87 3.61]
	2	[45.65 44.06]	[30.29 32.11]	[11.65 16.9]
LVAE	0	[12.69 12.51]	[1.53 1.5]	[1.74 1.72]
	1	[22.84 23.6]	[9.02 9.8]	[2.8 3.55]
	2	[45.76 44.17]	[32.13 33.75]	[25.4 28.35]
VAE	0	[25.09 24.79]	[16.6 16.31]	[3.72 3.69]
	1	[29.77 30.65]	[23.38 24.26]	[19.43 20.66]
	2	[51.59 50.43]	[41.16 41.41]	[32.86 35.55]

(a) Les trois premiers écoulements.

	Plan	11	12	13	14	21	31	32
POD	Entraînement	25.32	24.51	25.1	24.96	24.69	27.62	28.62
	Test	35.7	34.47	34.58	34.36	39.84	35.04	36.14
LAE	Entraînement	39.58	44.03	44.05	39.86	47.67	30.45	31.75
	Test	47.81	50.35	50.1	47.5	57.61	37.84	39.25
LVAE	Entraînement	29.31	28.85	29.75	29.19	30.18	28.91	30.15
	Test	38.89	37.94	38.3	37.89	43.34	36.35	37.76
VAE	Entraînement	62.51	61.93	61.64	60.59	60.68	67.53	66.89
	Test	69.7	69.71	68.66	67.48	70.65	71.89	72.05

(b) Écoulement urbain dans les différents plans (niveau de réduction $r_{0.8}$)

Tableau D.1 *– Erreurs moyennes normalisées d'auto-encodage pour les différents écoulements considérés. Pour les trois premiers cas, les erreurs sont données sous la forme [entraînement - test].*

E. Erreurs de reconstruction

Cette annexe rassemble les résultats quantitatifs des reconstructions. Pour un écoulement, une méthode de réduction, une grille de capteurs et une méthode de reconstruction, on dispose :

— De l'erreur moyenne normalisée (NMSE). Il s'agit du ratio entre l'erreur quadratique moyenne d'estimation *des clichés du champ de vitesse* et la variabilité attendue. L'erreur englobe les failles de reconstruction de l'état latent *et* les failles du décodage. Une valeur proche de zéro est souhaitée.

— Du coefficient de détermination moyen (R^2). C'est le ratio entre la variabilité restituée dans l'estimation *des états latents* et la variance de l'état latent réel. Une valeur proche de 1 est souhaitée.

Les tableaux sont *peu* interprétables tels quels. Il sont donnés pour référence, afin de jauger la performance d'une méthode de reconstruction pour un nouveau cas. Quelques tendances se dégagent toutefois :

1. Les scores de reconstruction sont généralement bons c'est-à-dire que les états latents sont bien estimés par les mesures. Le passage vers l'espace physique introduit de l'erreur car les modes ne caractérisent pas toute l'énergie de l'écoulement.
2. Pour l'écoulement périodique autour du cylindre, les scores sont excellents (quasiment égaux à 1) avec les méthodes supervisées. Avec la réduction linéaire variationnelle, la reconstruction directe avec cinq modes de référence n'est pas adaptée.
3. Plus l'écoulement se complexifie, plus les scores se détériorent. Augmenter le nombre de capteurs améliore le score de reconstruction mais ce résultat est biaisé ; il aurait fallu utiliser un coefficient de détermination ajusté.
4. Malgré nos efforts d'optimisation des hyper-paramètres, le sur-apprentissage des méthodes se manifeste lorsque la complexité de l'écoulement augmente. En ajoutant le sur-apprentissage des modes, la différence entre les estimations des clichés d'entraînement et les estimations des clichés de test peut être importante.
5. Avec l'expérience acquise, un conseil serait de commencer par une réduction POD et une reconstruction SVR (si cas simple) ou LM (si cas complexe).

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[14.42 12.84 12.91]	[1.53 1.37 1.35]	[1.44 0.16 0.13]
	LAE	[14.19 12.96 13.06]	[1.53 1.37 1.35]	[0.73 0.16 0.13]
	LVAE	[15.03 12.71 12.70]	[1.73 1.49 1.46]	[10.32 9.73 1.32]
	VAE		Not implemented	
LM	POD	[12.88 12.88 12.88]	[1.44 1.34 1.34]	[0.28 0.12 0.12]
	LAE	[13.01 13.00 13.00]	[1.44 1.34 1.34]	[0.27 0.12 0.12]
	LVAE	[12.70 12.69 12.69]	[1.63 1.53 1.53]	[1.84 1.74 1.74]
	VAE	[25.09 25.09 25.09]	[16.46 16.52 16.52]	[3.61 3.47 3.49]
LS	POD	[12.88 12.88 12.88]	[1.44 1.34 1.34]	[0.28 0.12 0.12]
	LAE	[13.01 13.00 13.00]	[1.44 1.34 1.34]	[0.27 0.12 0.12]
	LVAE	[12.70 12.69 12.69]	[1.63 1.53 1.53]	[1.84 1.74 1.74]
	VAE	[25.09 25.09 25.09]	[16.46 16.52 16.53]	[3.61 3.46 3.46]
SVR	POD	[12.99 12.95 12.94]	[1.46 1.47 1.50]	[0.24 0.26 0.29]
	LAE	[13.11 13.07 13.08]	[1.45 1.47 1.50]	[0.21 0.20 0.20]
	LVAE	[12.89 12.87 12.87]	[1.82 1.81 1.91]	[2.08 2.15 2.25]
	VAE	[25.11 25.10 25.09]	[16.62 16.95 16.86]	[3.55 3.67 3.70]
NN	POD	[12.88 12.89 12.86]	[1.38 1.65 1.34]	[0.43 0.13 0.33]
	LAE	[13.02 13.02 13.00]	[1.75 1.42 1.36]	[0.31 0.13 0.13]
	LVAE	[12.68 12.87 12.72]	[2.13 1.54 1.54]	[1.76 1.76 1.79]
	VAE	[25.15 25.09 25.08]	[16.62 16.67 16.72]	[3.77 4.48 3.73]
GB	POD	[12.89 12.88 12.89]	[1.44 1.41 1.36]	[0.25 0.2 0.15]
	LAE	[13.03 13.02 13.01]	[1.44 1.42 1.37]	[0.25 0.17 0.18]
	LVAE	[12.75 12.73 12.71]	[1.71 1.60 1.73]	[1.93 1.92 1.93]
	VAE	[25.10 25.10 25.10]	[17.42 16.98 16.73]	[3.95 3.83 3.86]

(a) Erreurs d'entraînement

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[14.08 12.56 12.63]	[1.52 1.35 1.33]	[1.55 0.16 0.13]
	LAE	[13.81 12.64 12.74]	[1.52 1.35 1.33]	[0.78 0.16 0.13]
	LVAE	[14.78 12.54 12.53]	[1.73 1.46 1.44]	[10.06 9.66 1.29]
	VAE		Not implemented	
LM	POD	[12.6 12.6 12.6]	[1.44 1.32 1.32]	[0.30 0.12 0.12]
	LAE	[12.69 12.68 12.68]	[1.44 1.32 1.32]	[0.29 0.12 0.12]
	LVAE	[12.53 12.51 12.51]	[1.61 1.50 1.50]	[1.82 1.72 1.72]
	VAE	[24.79 24.79 24.79]	[16.13 16.25 16.22]	[3.49 3.45 3.48]
LS	POD	[12.6 12.6 12.6]	[1.44 1.32 1.32]	[0.30 0.12 0.12]
	LAE	[12.69 12.68 12.68]	[1.44 1.32 1.32]	[0.29 0.12 0.12]
	LVAE	[12.53 12.51 12.51]	[1.61 1.5 1.5]	[1.82 1.72 1.72]
	VAE	[24.79 24.79 24.79]	[16.13 16.23 16.24]	[3.49 3.45 3.42]
SVR	POD	[12.71 12.68 12.65]	[1.44 1.45 1.48]	[0.25 0.26 0.29]
	LAE	[12.79 12.75 12.75]	[1.43 1.45 1.47]	[0.21 0.2 0.21]
	LVAE	[12.72 12.71 12.69]	[1.80 1.80 1.89]	[2.06 2.12 2.22]
	VAE	[24.80 24.80 24.79]	[16.29 16.65 16.56]	[3.51 3.65 3.67]
NN	POD	[12.60 12.61 12.58]	[1.36 1.63 1.32]	[0.37 0.14 0.35]
	LAE	[12.69 12.71 12.68]	[1.73 1.40 1.34]	[0.30 0.14 0.13]
	LVAE	[12.52 12.71 12.55]	[2.13 1.53 1.52]	[1.75 1.74 1.78]
	VAE	[24.85 24.79 24.78]	[16.32 16.36 16.42]	[3.76 4.51 3.69]
GB	POD	[12.61 12.60 12.62]	[1.55 1.44 1.37]	[0.40 0.26 0.19]
	LAE	[12.70 12.73 12.68]	[1.49 1.42 1.37]	[0.32 0.19 0.2]
	LVAE	[12.59 12.57 12.55]	[1.8 1.61 1.72]	[1.96 1.96 1.92]
	VAE	[24.80 24.79 24.80]	[17.18 16.7 16.42]	[4.05 3.80 3.78]

(b) Erreurs de test

Tableau E.1 *– Erreurs moyennes normalisées de reconstruction pour l'écoulement laminaire autour du cylindre. Les valeurs sont indiquées selon [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$]*

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[91.13 99.76 99.93]	[99.23 99.87 99.97]	[93.85 99.89 99.97]
	LAE	[93.55 99.56 99.90]	[99.23 99.87 99.97]	[95.03 99.78 99.95]
	LVAE	[83.59 99.81 99.77]	[98.43 99.45 99.66]	[33.47 46.59 47.51]
	VAE		Not implemented	
LM	POD	[99.98 100. 100.]	[99.59 100. 100.]	[99.28 100. 100.]
	LAE	[99.99 100. 100.]	[99.59 100. 100.]	[98.87 100. 100.]
	LVAE	[99.9 100. 100.]	[99.6 100. 100.]	[99.57 100. 100.]
	VAE	[89.33 98.67 100.]	[94.77 98.08 98.30]	[97.16 98.59 98.89]
LS	POD	[99.98 100. 100.]	[99.59 100. 100.]	[99.28 100. 100.]
	LAE	[99.99 100. 100.]	[99.59 100. 100.]	[98.87 100. 100.]
	LVAE	[99.9 100. 100.]	[99.6 100. 100.]	[99.57 100. 100.]
	VAE	[89.33 98.67 100.]	[94.77 98.04 98.34]	[97.16 98.59 98.59]
SVR	POD	[99.5 99.58 99.32]	[99.54 99.48 99.33]	[99.53 99.48 99.33]
	LAE	[99.54 99.53 99.3]	[99.54 99.44 99.36]	[99.48 99.49 99.5]
	LVAE	[99.36 99.46 99.36]	[99.54 99.51 99.37]	[99.5 99.44 99.35]
	VAE	[95.99 98.77 99.44]	[97.74 99.32 99.42]	[98.96 99.43 99.48]
NN	POD	[99.97 99.96 99.84]	[99.83 98.69 99.99]	[98.77 99.95 99.19]
	LAE	[99.91 99.95 99.88]	[98.31 99.66 99.93]	[99.17 99.93 99.96]
	LVAE	[99.82 98.07 99.95]	[97.72 99.96 99.94]	[99.9 99.97 99.89]
	VAE	[95.6 99.9 99.54]	[99.82 97.88 99.17]	[99.42 96.19 99.88]
GB	POD	[99.7 99.94 99.92]	[99.57 99.71 99.91]	[99.52 99.68 99.9]
	LAE	[99.71 99.88 99.95]	[99.61 99.67 99.9]	[99.42 99.79 99.78]
	LVAE	[99.84 99.9 99.97]	[99.64 99.88 99.66]	[99.74 99.83 99.83]
	VAE	[99.66 99.76 99.87]	[94.73 99.15 99.81]	[99.47 99.72 99.71]

(a) Scores d'entraînement

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[91.78 99.79 99.93]	[99.16 99.89 99.97]	[93.27 99.88 99.97]
	LAE	[94.12 99.61 99.9]	[99.17 99.89 99.97]	[94.63 99.76 99.95]
	LVAE	[84.44 99.8 99.77]	[98.34 99.5 99.68]	[31.63 46.65 43.76]
	VAE		Not implemented	
LM	POD	[99.98 100. 100.]	[99.51 100. 100.]	[99.2 100. 100.]
	LAE	[99.99 100. 100.]	[99.52 100. 100.]	[98.78 100. 100.]
	LVAE	[99.88 100. 100.]	[99.53 100. 100.]	[99.5 100. 100.]
	VAE	[89.24 99.19 99.92]	[94.54 97.56 97.92]	[96.97 98.52 98.83]
LS	POD	[99.98 100. 100.]	[99.51 100. 100.]	[99.2 100. 100.]
	LAE	[99.99 100. 100.]	[99.52 100. 100.]	[98.78 100. 100.]
	LVAE	[99.88 100. 100.]	[99.53 100. 100.]	[99.5 100. 100.]
	VAE	[89.24 99.19 99.92]	[94.54 97.57 97.95]	[96.97 98.54 98.57]
SVR	POD	[99.5 99.58 99.29]	[99.51 99.47 99.31]	[99.51 99.48 99.31]
	LAE	[99.54 99.55 99.29]	[99.53 99.46 99.36]	[99.51 99.50 99.50]
	LVAE	[99.34 99.42 99.37]	[99.54 99.52 99.37]	[99.45 99.47 99.34]
	VAE	[96.18 99.02 99.37]	[97.33 99.29 99.34]	[98.96 99.39 99.43]
NN	POD	[99.97 99.96 99.86]	[99.83 98.68 99.99]	[98.99 99.95 99.14]
	LAE	[99.89 99.94 99.89]	[98.29 99.67 99.92]	[99.20 99.93 99.95]
	LVAE	[99.83 97.79 99.95]	[97.6 99.96 99.94]	[99.90 99.97 99.89]
	VAE	[96.13 99.90 99.30]	[99.74 97.95 99.10]	[99.34 96.14 99.91]
GB	POD	[99.34 99.89 99.85]	[99.03 99.50 99.80]	[98.92 99.44 99.75]
	LAE	[99.23 99.78 99.88]	[99.30 99.58 99.78]	[99.09 99.67 99.68]
	LVAE	[99.79 99.79 99.92]	[99.34 99.68 99.62]	[99.53 99.67 99.72]
	VAE	[99.28 98.35 98.17]	[92.74 98.28 99.49]	[98.99 99.44 99.47]

(b) Scores de test

Tableau E.2 – *Coefficients de détermination de reconstruction pour l'écoulement laminaire autour du cylindre. Les scores sont indiqués selon [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$]*

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[26.76 24.98 23.14]	[26.44 22.6 10.78]	[112. 21.61 7.26]
	LAE	[27.3 25.1 23.17]	[31.64 26.28 11.28]	[108.03 27.99 9.86]
	LVAE	[28.19 25.32 23.18]	[30.67 31.12 10.94]	[123.56 22.29 8.00]
	VAE		Not implemented	
LM	POD	[25.10 23.34 22.80]	[21.11 17.61 9.52]	[19.96 15.32 5.40]
	LAE	[25.13 23.36 22.84]	[21.37 17.77 9.89]	[20.26 15.77 5.8 0]
	LVAE	[25.16 23.38 22.85]	[21.18 17.70 9.63]	[20.15 15.59 5.62]
	VAE	[29.72 29.32 29.28]	[27.66 25.97 22.17]	[27.05 24.89 19.77]
LS	POD	[25.08 23.34 22.80]	[21.17 17.28 9.52]	[20.27 15.39 5.41]
	LAE	[25.1 23.36 22.84]	[21.5 17.61 9.88]	[20.39 15.81 6.01]
	LVAE	[25.12 23.38 22.85]	[21.19 17.45 9.67]	[20.26 15.54 5.88]
	VAE	[29.72 29.33 29.28]	[27.55 25.92 22.15]	[26.96 24.65 19.71]
SVR	POD	[24.59 23.21 22.87]	[19.94 16.57 9.32]	[18.39 14.07 3.20]
	LAE	[24.63 23.26 22.91]	[20.63 16.38 9.67]	[19.19 14.19 4.39]
	LVAE	[24.89 23.26 22.92]	[20.5 16.59 9.46]	[18.63 14.00 4.55]
	VAE	[29.58 29.42 29.53]	[27.16 25.52 22.55]	[26.22 24.05 19.14]
NN	POD	[25.16 23.41 22.95]	[21.3 17.82 9.66]	[21.61 16.08 6.58]
	LAE	[24.67 23.68 22.98]	[20.88 17.76 10.21]	[20.52 15.75 6.38]
	LVAE	[24.9 23.31 22.93]	[21.76 18.20 9.82]	[19.54 15.03 6.11]
	VAE	[29.72 29.63 29.63]	[27.72 26.10 23.72]	[27.77 25.01 19.33]
GB	POD	[23.8 22.95 22.82]	[16.84 12.10 9.09]	[13.61 7.23 2.88]
	LAE	[24.01 22.95 22.92]	[17.02 11.36 9.48]	[14.06 5.89 3.42]
	LVAE	[23.65 23.02 22.93]	[15.57 12.12 9.32]	[13.41 7.08 3.46]
	VAE	[29.65 29.62 29.66]	[25.46 23.94 22.97]	[23.57 20.46 19.12]

(a) Erreurs d'entraînement

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[28.31 26.18 23.89]	[29.29 25.06 11.27]	[125.56 23.60 8.60]
	LAE	[28.91 26.29 23.94]	[33.36 27.56 11.58]	[110.76 29.68 11.07]
	LVAE	[29.67 26.47 23.97]	[32.17 33.40 11.40]	[126.37 24.90 9.20]
	VAE		Not implemented	
LM	POD	[25.90 24.22 23.51]	[23.13 20.05 10.26]	[22.15 18.10 6.30]
	LAE	[26.00 24.28 23.58]	[23.41 20.26 10.64]	[22.43 18.49 6.76]
	LVAE	[26.04 24.33 23.62]	[23.26 20.16 10.44]	[22.35 18.39 6.58]
	VAE	[30.47 30.19 30.17]	[28.80 27.41 22.96]	[28.68 26.93 20.55]
LS	POD	[25.86 24.22 23.51]	[23.10 19.76 10.25]	[22.23 18.06 6.31]
	LAE	[25.95 24.28 23.58]	[23.63 20.07 10.64]	[22.73 18.61 6.97]
	LVAE	[25.96 24.33 23.62]	[23.24 20.06 10.48]	[22.54 18.58 6.77]
	VAE	[30.46 30.20 30.18]	[28.73 27.44 22.98]	[28.62 26.90 20.51]
SVR	POD	[25.68 24.13 23.63]	[22.92 19.76 10.39]	[21.72 17.80 5.91]
	LAE	[25.88 24.18 23.71]	[23.21 19.75 10.73]	[22.28 18.15 6.48]
	LVAE	[25.87 24.25 23.73]	[23.12 19.84 10.51]	[22.07 17.81 6.35]
	VAE	[30.48 30.32 30.47]	[28.85 27.38 23.50]	[28.42 26.78 20.34]
NN	POD	[25.87 24.45 23.95]	[22.91 19.79 10.53]	[22.99 18.11 7.87]
	LAE	[25.94 24.57 23.78]	[23.22 19.99 11.32]	[22.11 18.12 9.46]
	LVAE	[25.86 24.35 23.80]	[23.29 20.16 10.59]	[21.70 17.50 6.91]
	VAE	[30.52 30.51 30.55]	[28.72 27.37 24.64]	[28.91 26.58 20.71]
GB	POD	[26.24 24.70 23.97]	[22.98 19.63 12.09]	[21.89 17.93 8.68]
	LAE	[26.52 24.75 24.18]	[23.51 20.07 13.13]	[22.17 18.09 10.61]
	LVAE	[26.52 25.11 24.41]	[23.82 20.27 13.09]	[22.31 18.55 10.11]
	VAE	[30.82 30.71 30.64]	[28.9 27.27 23.82]	[28.10 25.82 20.99]

(b) Erreurs de test

Tableau E.3 *– Erreurs moyennes normalisées de reconstruction pour la couche de mélange. Les valeurs sont indiquées selon* $[p_{0.2}, p_{0.4}, p_{0.8}]$

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[75.11 87.48 99.22]	[47.61 62.42 97.1]	[0.0 58.57 94.33]
	LAE	[72.49 86.82 99.0]	[31.25 51.62 95.67]	[0.0 38.02 88.42]
	LVAE	[60.49 82.86 98.97]	[21.33 18.9 94.64]	[0.0 33.91 81.64]
	VAE		Not implemented	
LM	POD	[86.28 96.74 99.98]	[64.2 76.21 99.13]	[59.24 72.89 96.63]
	LAE	[86.55 96.75 99.97]	[62.04 75.01 98.83]	[54. 68.88 95.4]
	LVAE	[83.14 95.74 99.97]	[55.45 68.21 98.57]	[44.37 60.3 91.7]
	VAE	[76.01 81.34 83.85]	[53.57 63.64 80.08]	[44.51 55.7 83.28]
LS	POD	[86.35 96.74 99.98]	[63.87 77.06 99.09]	[58.31 72.68 96.57]
	LAE	[86.65 96.75 99.97]	[61.51 75.36 98.8]	[53.65 68.68 94.89]
	LVAE	[83.17 95.74 99.97]	[55.41 69.13 98.4]	[43.96 60.47 90.84]
	VAE	[75.98 81.27 83.77]	[54.07 63.7 80.53]	[44.65 56.38 83.48]
SVR	POD	[89.24 97.65 99.72]	[67.8 79.07 99.42]	[63.28 75.74 98.84]
	LAE	[89.3 97.36 99.72]	[64.43 79.26 99.24]	[57.03 72.89 97.48]
	LVAE	[85.1 96.63 99.73]	[57.92 72.21 99.04]	[49.81 65.71 94.82]
	VAE	[84.25 90.19 95.61]	[56.71 67.19 92.02]	[48.2 59.78 88.91]
NN	POD	[85.6 95.68 98.9]	[63.19 75.56 98.56]	[53.01 70.65 93.07]
	LAE	[88.7 95.13 98.86]	[63.46 75.0 97.02]	[52.74 68.8 91.91]
	LVAE	[85.01 96.44 99.26]	[52.6 65.98 97.85]	[46.01 61.91 90.07]
	VAE	[84.49 88.02 97.21]	[54.89 66.86 87.59]	[41.0 55.59 91.7]
GB	POD	[93.58 99.14 99.76]	[76.27 90.79 99.4]	[73.14 89.2 98.97]
	LAE	[92.81 99.33 99.41]	[73.48 92.44 99.05]	[69.12 91.67 98.34]
	LVAE	[94.04 98.76 99.44]	[74.38 88.28 98.98]	[65.54 86.61 97.66]
	VAE	[92.53 95.45 97.54]	[69.95 84.08 96.62]	[58.55 79.34 96.76]

(a) Scores d'entraînement

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[67.93 83.72 99.06]	[40.72 55.72 97.41]	[0.0 53.97 93.58]
	LAE	[64.58 83.3 98.89]	[24.9 47.43 96.37]	[0.0 33.87 87.67]
	LVAE	[51.96 78.91 98.92]	[15.44 10.29 95.56]	[0.0 27.62 80.65]
	VAE		Not implemented	
LM	POD	[84.54 95.51 99.98]	[59.41 70.6 98.97]	[54.84 67.25 96.33]
	LAE	[84.77 95.54 99.96]	[57.16 68.94 98.62]	[49.49 63.48 94.97]
	LVAE	[81.78 94.29 99.94]	[49.66 61.35 98.28]	[40.47 55.19 90.97]
	VAE	[72.16 75.71 78.1]	[48.8 56.51 77.14]	[40.61 49.32 80.43]
LS	POD	[84.69 95.51 99.98]	[59.38 71.07 98.93]	[54.45 67.22 96.26]
	LAE	[84.91 95.54 99.96]	[56.35 69.36 98.55]	[48.96 63.17 94.49]
	LVAE	[82.06 94.29 99.94]	[49.59 61.31 98.11]	[39.99 54.25 90.23]
	VAE	[72.15 75.43 77.86]	[48.87 56.16 76.68]	[40.37 49.06 80.35]
SVR	POD	[85.99 96.24 99.58]	[60.34 71.41 98.67]	[56.13 68.07 96.29]
	LAE	[85.58 96.19 99.58]	[57.99 70.73 98.4]	[50.35 64.44 94.95]
	LVAE	[82.98 95.13 99.66]	[50.28 62.53 98.04]	[42.65 57.09 91.21]
	VAE	[78.39 82.63 82.98]	[49.41 57.08 80.52]	[41.33 49.81 81.73]
NN	POD	[84.52 93.33 96.1]	[59.73 71.0 98.09]	[50.88 66.74 91.72]
	LAE	[85.01 94.02 98.74]	[58.08 69.8 95.88]	[49.95 64.12 86.91]
	LVAE	[83.21 94.57 98.61]	[49.6 60.5 97.58]	[42.23 57.24 89.92]
	VAE	[81.9 87.14 95.2]	[50.76 60.14 84.46]	[38.41 51.21 84.83]
GB	POD	[82.56 92.97 96.88]	[58.75 70.3 92.18]	[53.78 65.57 87.05]
	LAE	[81.96 93.08 95.7]	[55.22 68.65 89.28]	[48.54 61.5 80.25]
	LVAE	[78.19 89.03 94.75]	[47.7 60.39 86.78]	[39.49 51.08 72.6]
	VAE	[73.83 77.75 81.6]	[49.01 58.76 77.57]	[40.3 51.14 76.77]

(b) Scores de test

Tableau E.4 – *Coefficients de détermination de reconstruction pour la couche de mélange. Les scores sont indiqués selon [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$]*

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[47.36 46.59 45.45]	[48.84 42.86 33.76]	[62.91 74.23 27.44]
	LAE	[47.81 47.07 46.]	[52.07 45.57 34.67]	[78.36 86.67 29.95]
	LVAE	[47.71 47.07 46.02]	[48.95 44.12 34.48]	[70.5 97.72 43.51]
	VAE		Not implemented	
LM	POD	[46.8 45.85 45.17]	[40.47 36.91 31.16]	[38.39 31.78 16.41]
	LAE	[47.35 46.42 45.74]	[41.6 38.1 32.15]	[38.92 32.82 18.53]
	LVAE	[47.33 46.41 45.79]	[42.19 38.95 33.63]	[41.5 37.5 28.99]
	VAE	[49.55 48.98 48.45]	[44.8 42.9 40.44]	[42.28 38.97 33.66]
LS	POD	[46.82 45.84 45.19]	[40.57 36.98 31.26]	[38.66 32.48 17.3]
	LAE	[47.33 46.42 45.75]	[41.58 38.19 32.28]	[38.98 32.99 19.16]
	LVAE	[47.31 46.42 45.81]	[42.21 38.98 33.77]	[41.46 37.3 29.05]
	VAE	[49.55 49. 48.41]	[44.71 42.79 40.43]	[42.3 39.04 33.9]
SVR	POD	[45.63 45.43 45.26]	[33.7 31.25 29.56]	[28.31 19.05 10.61]
	LAE	[46.24 45.98 45.75]	[34.92 32.76 30.73]	[28.61 21.58 13.17]
	LVAE	[46.16 45.98 45.84]	[36.21 34.92 32.47]	[33.93 30.21 26.24]
	VAE	[50.04 49.99 50.96]	[42.12 41.31 40.41]	[36.54 34.43 32.58]
NN	POD	[45.78 45.36 45.91]	[38.14 35.41 31.35]	[39.22 32.03 18.52]
	LAE	[46.83 46.51 45.72]	[42. 38.45 31.52]	[32.38 31.06 21.42]
	LVAE	[47.16 45.99 45.84]	[39.21 38.23 33.5]	[40.85 36.33 29.98]
	VAE	[53.04 50.75 51.54]	[43.96 42.47 41.32]	[38.96 37.6 33.85]
GB	POD	[45.48 45.45 45.23]	[33.12 31.62 30.37]	[26.47 22.6 17.71]
	LAE	[46.3 45.88 45.77]	[33.58 32.42 31.35]	[24.71 20.31 16.89]
	LVAE	[46.3 46.11 45.93]	[35.32 34.28 33.39]	[31.44 29.17 27.57]
	VAE	[50.68 51.29 51.08]	[41.91 41.36 41.08]	[35.54 34.29 33.31]

(a) Erreurs d'entraînement

Méthode	Décodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[45.12 44.43 43.56]	[47.98 42.36 35.2]	[65.96 96.11 30.79]
	LAE	[45.97 45.18 44.31]	[51.51 45.75 36.56]	[80.78 107.44 34.43]
	LVAE	[45.81 45.21 44.35]	[48.85 44.81 36.24]	[77.93 124.96 58.26]
	VAE		Not implemented	
LM	POD	[44.77 43.91 43.38]	[40.81 38.28 33.51]	[39.83 36. 25.51]
	LAE	[45.5 44.65 44.14]	[41.98 39.68 34.53]	[40.32 36.77 26.47]
	LVAE	[45.55 44.72 44.21]	[42.39 40.19 35.57]	[42.22 39.51 32.61]
	VAE	[48.29 47.89 47.2]	[44.27 42.71 40.65]	[42.59 40.21 35.59]
LS	POD	[44.8 43.93 43.39]	[40.9 38.48 33.68]	[40.1 36.49 25.69]
	LAE	[45.49 44.66 44.15]	[42.06 39.85 34.71]	[40.48 37.12 26.91]
	LVAE	[45.58 44.74 44.23]	[42.41 40.34 35.75]	[42.21 39.5 32.68]
	VAE	[48.31 47.9 47.13]	[44.23 42.63 40.62]	[42.54 40.23 35.76]
SVR	POD	[44.28 43.8 43.94]	[38.91 37.22 33.81]	[37.6 34.6 24.98]
	LAE	[45.03 44.56 44.14]	[40.09 38.45 34.51]	[38.63 35.86 25.93]
	LVAE	[45.07 44.66 44.28]	[40.65 39.09 35.35]	[39.89 37.86 32.2]
	VAE	[49.51 48.6 49.01]	[43.19 42.2 40.65]	[40.59 38.93 35.68]
NN	POD	[44.31 43.92 44.29]	[39.7 38.64 34.74]	[41.19 37.61 27.94]
	LAE	[45.17 44.87 44.21]	[42.53 40.72 35.42]	[38.94 38.63 29.77]
	LVAE	[45.8 44.69 44.27]	[41.17 39.98 36.18]	[42.14 39.84 33.45]
	VAE	[52.29 49.5 49.95]	[43.7 42.73 41.64]	[41.33 40.57 36.77]
GB	POD	[44.47 44.22 43.88]	[39.42 38.1 35.31]	[38.74 36.84 31.81]
	LAE	[45.22 44.82 44.48]	[40.88 39.3 36.29]	[40.13 38.12 33.1]
	LVAE	[45.38 45.09 44.63]	[41.3 40.02 37.47]	[40.23 38.58 35.33]
	VAE	[49.65 49.66 49.49]	[43.35 42.44 41.35]	[40.96 39.63 37.02]

(b) Erreurs de test

Tableau E.5 *– Erreurs moyennes normalisées de reconstruction pour l'écoulement turbulent autour du cylindre. Les valeurs sont indiquées selon [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$]*

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[93.52 96.06 99.45]	[62.81 77.13 93.94]	[45.58 46.1 84.97]
	LAE	[93.09 95.91 99.26]	[42.31 61.04 89.11]	[0. 0. 68.83]
	LVAE	[92.82 95.32 98.59]	[25.33 41.96 76.47]	[0. 0. 0.]
	VAE		Not implemented	
LM	POD	[95.01 97.99 99.83]	[81.12 87.98 97.55]	[70.39 79.1 94.77]
	LAE	[94.73 97.81 99.73]	[70.1 80.3 95.65]	[53.19 64.6 89.41]
	LVAE	[94.35 97.7 99.5]	[59.9 72.08 93.17]	[49.77 62.11 88.31]
	VAE	[75.33 80.67 83.17]	[75.91 81.94 90.04]	[55.89 66.17 82.94]
LS	POD	[94.96 98.02 99.79]	[80.97 87.92 97.42]	[70.35 78.56 93.9]
	LAE	[94.78 97.77 99.68]	[70.2 79.94 95.29]	[53.28 64.52 88.49]
	LVAE	[94.33 97.64 99.44]	[59.83 71.98 92.58]	[49.61 62.39 87.78]
	VAE	[75.39 80.52 83.83]	[75.97 82.2 90.]	[55.57 65.64 81.69]
SVR	POD	[98.49 99.07 99.39]	[92.91 96.66 99.25]	[83.31 91.59 98.73]
	LAE	[98.29 99.04 99.66]	[88.14 93.9 98.81]	[71.91 84.24 97.75]
	LVAE	[98.4 99.02 99.56]	[83.11 88.68 98.51]	[72.59 84.71 97.52]
	VAE	[91.37 92.21 99.2]	[88.63 91.98 96.2]	[77.47 87.46 97.81]
NN	POD	[98.01 99.16 97.14]	[84.79 89.19 97.09]	[68.63 78.79 92.04]
	LAE	[96.46 97.13 99.70]	[69.61 78.82 96.99]	[66.15 68.5 85.15]
	LVAE	[95.24 99.04 99.47]	[71.38 75.92 94.21]	[51.45 64.86 85.28]
	VAE	[81.88 89.05 99.03]	[81.49 86.21 95.46]	[69.61 73.07 84.64]
GB	POD	[98.90 98.90 99.50]	[93.51 95.83 97.95]	[84.63 88.49 92.98]
	LAE	[97.94 99.22 99.50]	[90.98 94.2 96.84]	[78.36 86.01 91.69]
	LVAE	[97.95 98.92 99.40]	[87.99 91.83 94.88]	[80.78 87.69 91.97]
	VAE	[91.96 96.27 96.41]	[92.02 95.16 96.48]	[83.39 88.79 92.84]

(a) Scores d'entraînement

Méthode	Encodage	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[94.83 97.07 99.59]	[66.42 80.40 94.34]	[46.81 33.26 85.99]
	LAE	[93.78 96.76 99.45]	[45.41 63.11 88.57]	[0.00 0.00 67.14]
	LVAE	[93.77 96.26 98.83]	[22.72 38.37 73.05]	[0.00 0.00 0.00]
	VAE		Not implemented	
LM	POD	[95.61 98.21 99.82]	[83.13 88.46 96.98]	[72.92 79.2 91.93]
	LAE	[95.33 98.06 99.71]	[72.07 79.76 94.02]	[54.62 62.72 84.15]
	LVAE	[94.77 97.87 99.48]	[59.59 68.51 89.06]	[49.87 59.33 83.06]
	VAE	[77.85 81.25 84.37]	[77.82 82. 88.83]	[56.47 63.05 76.22]
LS	POD	[95.54 98.14 99.79]	[82.94 88.2 96.77]	[72.8 78.77 91.69]
	LAE	[95.3 98.02 99.66]	[71.96 79.37 93.57]	[54.45 62.31 83.46]
	LVAE	[94.67 97.81 99.43]	[59.41 68.11 88.46]	[49.59 58.86 82.14]
	VAE	[77.6 81.21 84.76]	[77.74 81.95 88.6]	[56.26 62.61 75.36]
SVR	POD	[97.02 98.45 97.92]	[87.12 90.47 95.64]	[76.9 81.29 90.92]
	LAE	[96.91 98.29 99.63]	[78.55 83.43 93.64]	[59.53 65.63 84.56]
	LVAE	[96.51 98.09 99.45]	[66.84 73.27 90.06]	[55.87 63.35 83.7]
	VAE	[83.73 84.43 86.59]	[80.95 83.94 89.26]	[61.55 66.82 77.45]
NN	POD	[96.94 98.05 96.59]	[84.76 86.48 95.1]	[70.31 77.35 88.49]
	LAE	[96.34 97. 99.41]	[70.96 76.1 91.77]	[58.26 59.6 78.03]
	LVAE	[94.25 97.85 99.4]	[65.5 70.3 86.91]	[49.72 58.6 80.35]
	VAE	[74.62 85.95 94.14]	[81.23 83.73 89.33]	[62.49 63.42 72.04]
GB	POD	[96.51 97.3 98.35]	[85.68 88.12 92.76]	[74.82 77.5 82.83]
	LAE	[96.18 97.56 98.45]	[75.66 80.28 87.48]	[55.8 60.39 69.18]
	LVAE	[95.57 96.71 98.15]	[64.19 69.1 78.64]	[53.2 58.39 68.31]
	VAE	[83.7 87.81 89.57]	[80.35 83.17 87.07]	[59.97 63.92 71.72]

(b) Scores de test

Tableau E.6 – *Coefficients de détermination de reconstruction pour l'écoulement turbulent autour du cylindre. Les scores sont indiqués selon [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$]*

	Plan	11	12	13	14	21	31	32
DR	POD	58.46	59.41	63.69	74.76	62.67	46.46	52.6
	LAE	70.23	75.36	80.24	88.29	78.58	53.03	58.75
	LVAE	60.79	60.72	65.84	76.56	65.07	55.39	62.64
	VAE	Not implemented						
LM	POD	47.07	47.12	50.24	57.48	50.04	37.66	42.5
	LAE	57.04	61.21	63.28	66.78	64.85	40.59	45.05
	LVAE	50.42	50.66	53.85	60.78	54.5	39.29	43.52
	VAE	73.68	73.99	75.56	77.9	75.09	72.9	74.25
LS	POD	47.85	47.34	50.39	57.96	50.57	37.77	42.15
	LAE	57.41	61.46	63.58	67.18	65.6	40.61	45.07
	LVAE	50.4	50.68	53.82	60.55	54.65	39.32	43.53
	VAE	72.87	73.21	74.89	77.3	74.8	72.67	73.95
SVR	POD	29.9	30.25	32.33	36.45	29.24	29.65	31.75
	LAE	43.56	47.2	47.64	52.09	50.05	32.58	34.7
	LVAE	34.14	33.99	34.54	40.67	34.25	32.15	33.43
	VAE	63.48	62.9	63.13	65.14	61.45	67.68	67.58
NN	POD	43.55	49.24	53.23	57.48	51.22	39.73	40.47
	LAE	56.79	64.26	62.15	69.0	65.52	42.64	46.09
	LVAE	49.6	52.91	54.5	61.84	56.2	36.42	44.46
	VAE	72.85	74.06	69.58	69.73	73.24	73.26	71.91
GB	POD	36.89	37.3	38.21	39.73	39.55	35.05	36.64
	LAE	45.89	49.88	50.17	47.84	53.19	37.38	38.72
	LVAE	39.34	39.42	40.41	40.73	40.95	36.77	38.11
	VAE	64.3	64.25	64.16	63.53	63.46	68.73	68.33

(a) Erreurs d'entraînement

	Plan	11	12	13	14	21	31	32
DR	POD	78.97	82.79	85.07	95.03	96.74	61.36	65.09
	LAE	87.09	93.07	97.2	107.33	98.76	72.01	75.71
	LVAE	83.61	85.77	89.48	99.98	103.8	76.74	83.76
	VAE	Not implemented						
LM	POD	65.21	67.37	69.3	74.86	76.22	52.09	56.5
	LAE	72.92	77.24	78.07	80.99	86.21	54.41	59.34
	LVAE	66.94	69.14	70.99	75.82	77.41	52.78	57.84
	VAE	79.5	81.08	82.34	84.63	85.71	77.54	80.47
LS	POD	65.85	68.43	70.43	75.81	77.6	52.39	57.39
	LAE	74.11	79.11	80.04	82.39	87.85	55.17	60.02
	LVAE	68.19	70.38	72.46	77.19	79.2	53.45	58.56
	VAE	79.76	81.48	82.95	85.08	86.36	77.83	80.8
SVR	POD	66.9	68.34	71.1	77.45	78.16	53.2	58.02
	LAE	74.61	79.3	81.45	83.34	88.61	55.92	60.63
	LVAE	68.03	70.34	73.31	78.05	79.88	54.89	59.39
	VAE	80.51	82.06	83.9	85.66	87.77	78.63	81.84
NN	POD	73.85	72.99	74.08	78.5	81.38	57.0	67.77
	LAE	91.38	80.6	87.04	85.46	96.17	57.78	63.22
	LVAE	70.84	74.96	75.18	80.63	84.36	59.35	66.43
	VAE	81.69	83.51	86.31	89.79	91.14	78.93	82.35
GB	POD	73.59	76.46	78.65	81.28	82.24	65.84	68.15
	LAE	79.31	83.65	84.08	86.23	88.06	68.39	70.83
	LVAE	75.84	78.2	79.07	82.63	83.84	67.17	68.2
	VAE	81.59	83.81	85.14	86.32	87.18	80.56	82.29

(b) Erreurs de test

Tableau E.7 – *Erreurs moyennes normalisées de reconstruction pour l'écoulement urbain. Les erreurs sont calculés avec une réduction $r_{0.8}$ et une grille de capteurs $p_{0.8}$.*

	Plan	11	12	13	14	21	31	32
DR	POD	0.79	0.78	0.76	0.68	0.76	0.92	0.89
	LAE	0.58	0.51	0.51	0.46	0.49	0.87	0.83
	LVAE	0.58	0.59	0.55	0.45	0.56	0.79	0.74
	VAE	Not implemented						
LM	POD	0.86	0.86	0.84	0.79	0.83	0.96	0.94
	LAE	0.8	0.79	0.77	0.71	0.76	0.94	0.92
	LVAE	0.77	0.77	0.76	0.68	0.75	0.93	0.91
	VAE	0.65	0.63	0.6	0.55	0.62	0.77	0.75
LS	POD	0.85	0.85	0.84	0.78	0.83	0.96	0.94
	LAE	0.8	0.79	0.78	0.71	0.75	0.94	0.92
	LVAE	0.77	0.77	0.76	0.68	0.75	0.93	0.9
	VAE	0.66	0.64	0.61	0.56	0.62	0.78	0.76
SVR	POD	0.97	0.96	0.95	0.91	0.97	0.99	0.98
	LAE	0.94	0.95	0.95	0.85	0.95	0.98	0.98
	LVAE	0.95	0.94	0.95	0.88	0.96	0.97	0.97
	VAE	0.92	0.93	0.92	0.82	0.95	0.95	0.94
NN	POD	0.87	0.83	0.8	0.78	0.82	0.94	0.92
	LAE	0.79	0.72	0.8	0.69	0.78	0.92	0.91
	LVAE	0.78	0.75	0.75	0.67	0.74	0.93	0.83
	VAE	0.68	0.65	0.74	0.73	0.66	0.79	0.82
GB	POD	0.88	0.87	0.86	0.86	0.84	0.92	0.92
	LAE	0.87	0.87	0.86	0.86	0.84	0.91	0.91
	LVAE	0.85	0.84	0.84	0.84	0.84	0.88	0.89
	VAE	0.88	0.86	0.86	0.85	0.86	0.9	0.9

(a) Scores d'entraînement

	Plan	11	12	13	14	21	31	32
DR	POD	0.71	0.68	0.66	0.59	0.62	0.9	0.87
	LAE	0.47	0.4	0.4	0.35	0.37	0.81	0.77
	LVAE	0.4	0.41	0.38	0.27	0.27	0.7	0.63
	VAE	Not implemented						
LM	POD	0.8	0.78	0.76	0.72	0.75	0.93	0.91
	LAE	0.71	0.68	0.68	0.64	0.63	0.91	0.88
	LVAE	0.67	0.66	0.65	0.6	0.63	0.88	0.84
	VAE	0.57	0.53	0.51	0.48	0.5	0.69	0.67
LS	POD	0.79	0.77	0.75	0.71	0.74	0.93	0.91
	LAE	0.7	0.67	0.67	0.62	0.61	0.9	0.87
	LVAE	0.65	0.64	0.64	0.59	0.61	0.86	0.83
	VAE	0.55	0.51	0.49	0.46	0.48	0.68	0.66
SVR	POD	0.77	0.76	0.74	0.69	0.73	0.9	0.89
	LAE	0.68	0.65	0.65	0.61	0.6	0.86	0.85
	LVAE	0.63	0.63	0.61	0.56	0.59	0.82	0.8
	VAE	0.55	0.5	0.48	0.45	0.47	0.68	0.65
NN	POD	0.73	0.72	0.7	0.68	0.7	0.89	0.78
	LAE	0.46	0.59	0.6	0.59	0.53	0.86	0.84
	LVAE	0.62	0.58	0.59	0.55	0.55	0.77	0.68
	VAE	0.53	0.49	0.4	0.35	0.37	0.68	0.64
GB	POD	0.58	0.54	0.52	0.52	0.5	0.65	0.66
	LAE	0.45	0.42	0.43	0.43	0.36	0.58	0.59
	LVAE	0.35	0.34	0.35	0.35	0.32	0.47	0.5
	VAE	0.45	0.4	0.38	0.37	0.36	0.53	0.53

(b) Scores de test

Tableau E.8 – *Coefficients de détermination de reconstruction pour l'écoulement urbain. Les scores sont calculés avec une réduction $r_{0.8}$ et une grille de capteurs $p_{0.8}$*

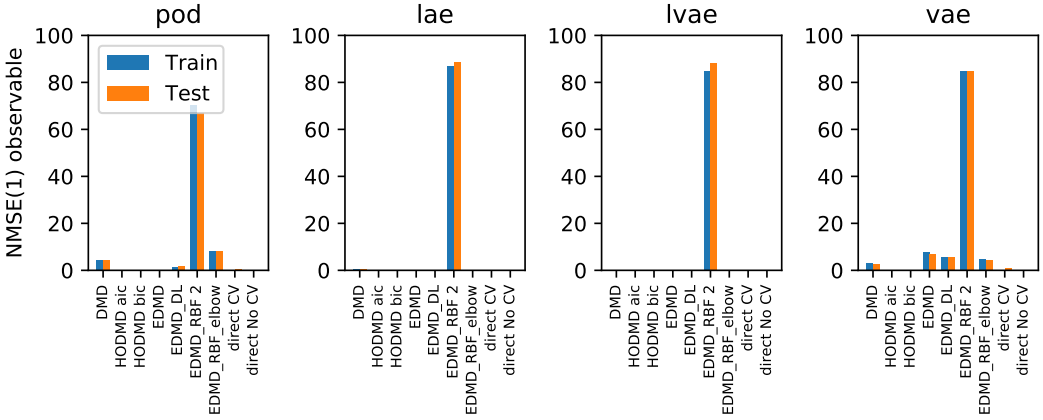
F. Erreurs de prédiction

Cette annexe rassemble les résultats quantitatifs des prédictions. Contrairement à l'étape de reconstruction où les données n'avaient pas besoin d'être organisées, l'étape de prédiction impose de les structurer. On rappelle que pour m_b clichés et un horizon de prédiction H, les données sont divisées en $m_b - H - 1$ séquences de $H + 1$ pas de temps. Pour un écoulement, une méthode de réduction et une méthode d'approximation de l'opérateur de Koopman, on dispose :

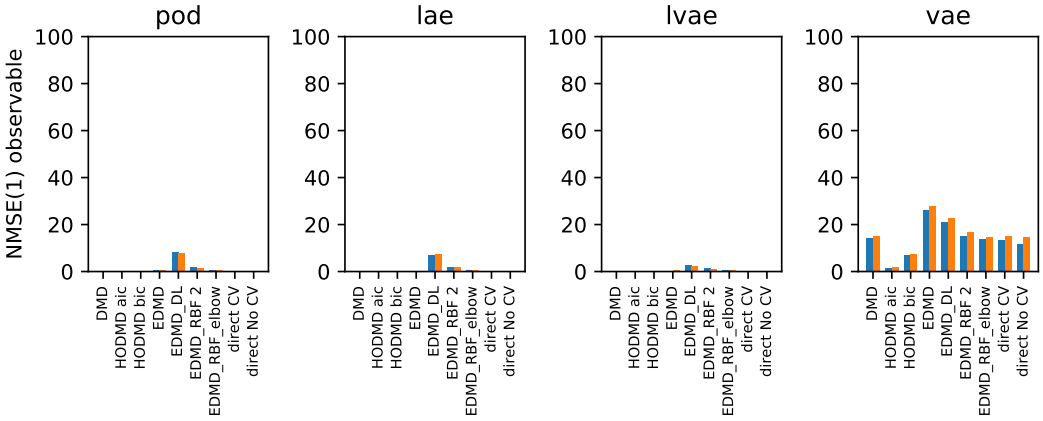
- De l'erreur *un pas de temps*. Il s'agit de l'erreur moyenne normalisée entre les états prédits et les états réels. Cela revient à prendre $H = 1$. On note l'erreur NMSE(1)
- De l'erreur *prédictive*. Les données d'entraînement et de test sont divisées en trajectoires de seize pas de temps. On calcule l'erreur moyenne normalisée entre les séquences réelles et les séquences prédites (par récursion) puis on moyenne sur l'horizon. Cela revient à prendre $H = 16$. Le choix de seize pas de temps est heuristique ; un trop petit horizon ne refléterait pas la qualité prédictive tu modèle tandis qu'un horizon trop grand mènerait à une erreur maximale du fait des fonctions propres fallacieuses. On note l'erreur NMSE.

Ces erreurs sont calculées dans l'espace des observables et dans l'espace latent. Quelques tendances se dégagent des résultats :

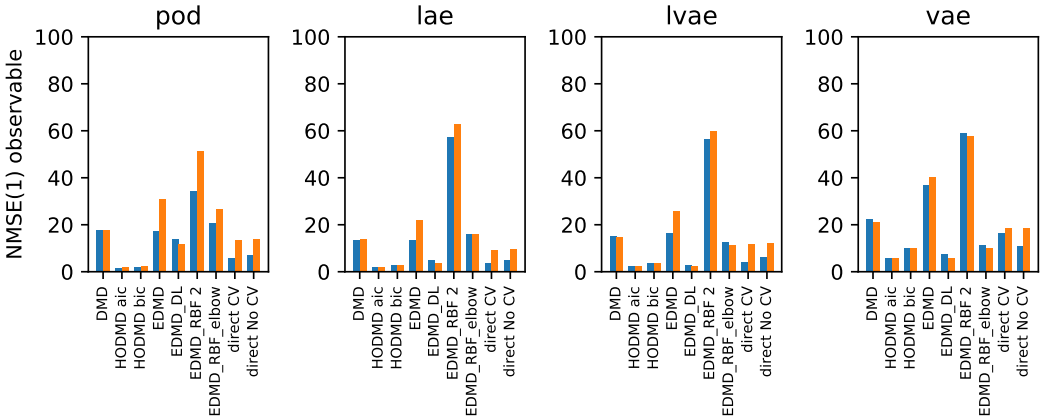
1. Pour l'écoulement laminaire autour du cylindre, les états latents issus des réductions linéaires non orthogonales sont les plus faciles à prédire. On évitera la réduction non linéaire variationnelle pour l'écoulement de couche de mélange tandis qu'on la préférera pour l'écoulement turbulent autour du cylindre.
2. L'erreur de prédiction DMD augmente avec la complexité de l'écoulement. Les décompositions modales dynamiques d'ordre plus élevé sont pertinentes, avec une préférence pour le critère BIC.
3. L'utilisation de modèles directs permet d'améliorer la qualité des prédictions récursives. La validation croisée des modèles diminue l'écart entre les erreurs d'entraînement et de test.
4. Les modèles prédictifs basés sur une décomposition étendue quadratique ou radiale ont des performances similaires.
5. L'apprentissage automatique du dictionnaire est décevant. Dans l'espace latent, les performances sont mauvaises pour les cas 1 et 2. Dans l'espace des observables, c'est pourtant la méthode la plus performante pour l'écoulement de sillage. Pour cette méthode, approcher l'inverse des observables par moindres carrés n'est donc pas suffisant et un réseau de neurones serait plus adapté.



(a) Écoulement laminaire autour du cylindre

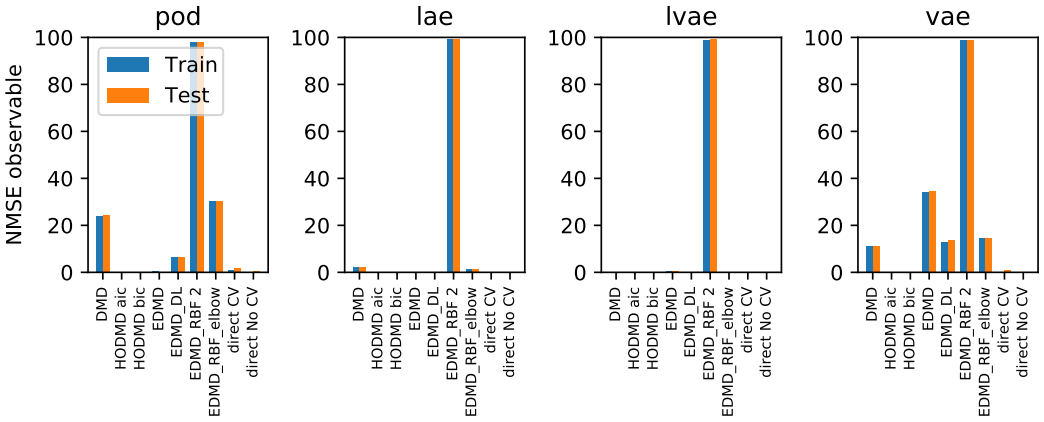


(b) Écoulement de couche de mélange

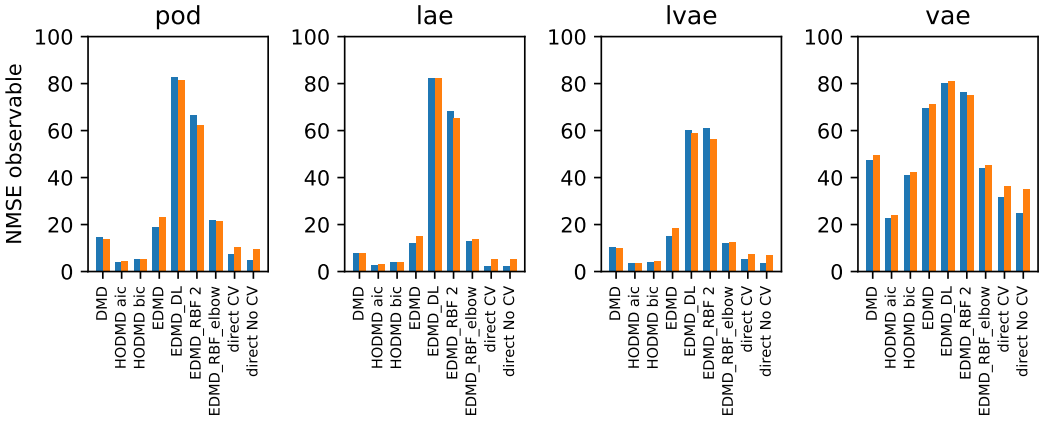


(c) Écoulement turbulent autour du cylindre

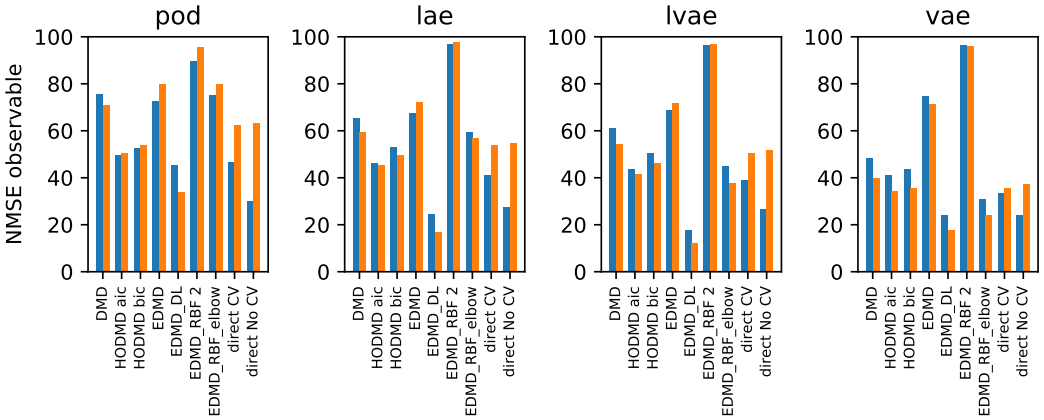
Figure *F.1* – *Erreur moyenne normalisée un pas de temps dans l'espace des observables (en %).*



(a) Écoulement laminaire autour du cylindre

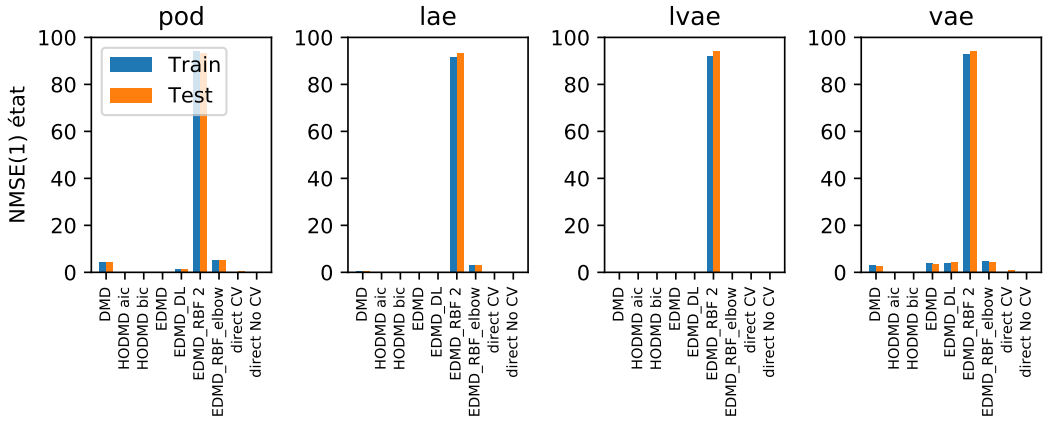


(b) Écoulement de couche de mélange

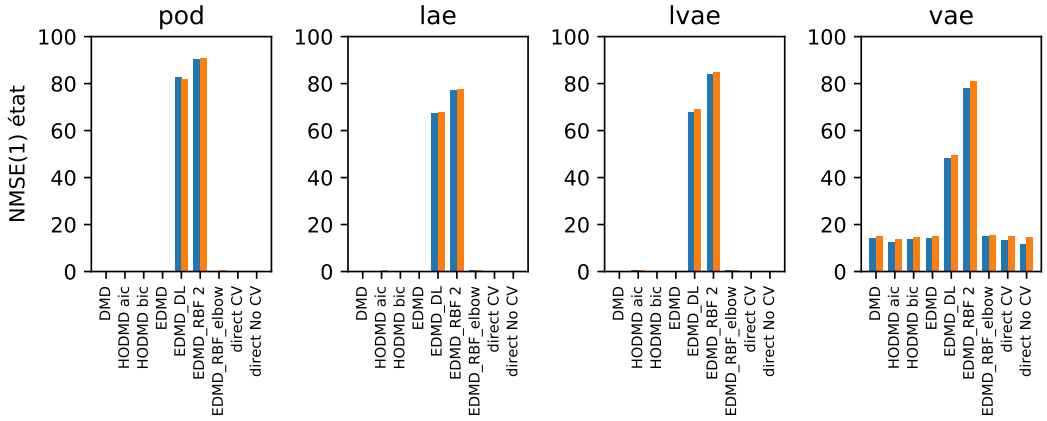


(c) Écoulement turbulent autour du cylindre

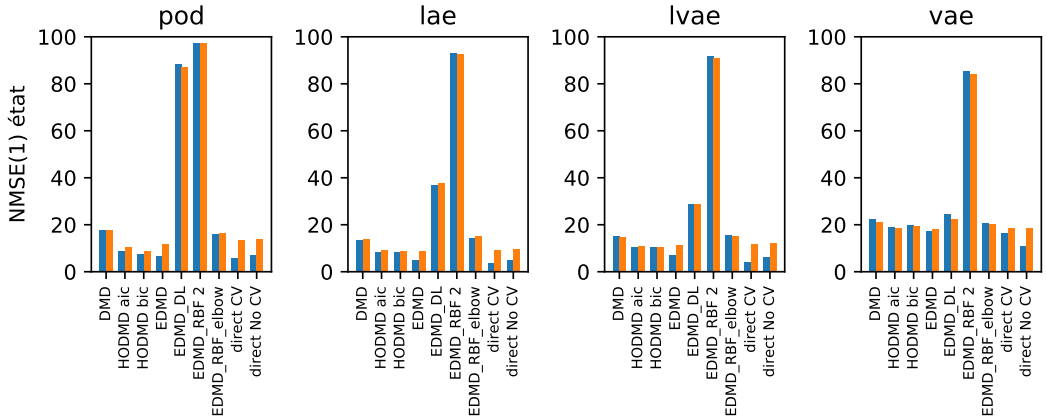
Figure *F.2* – *Erreur moyenne normalisée de prédiction dans l'espace des observables (en %).*



(a) Écoulement laminaire autour du cylindre

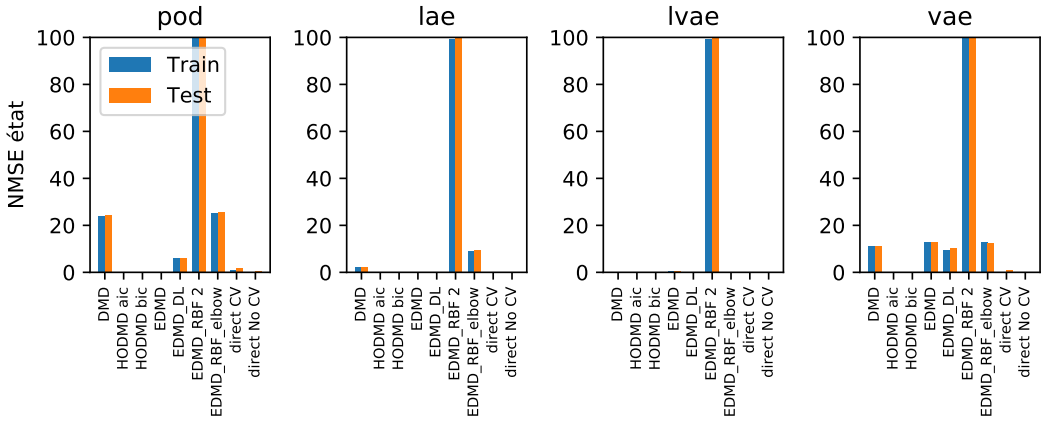


(b) Écoulement de couche de mélange

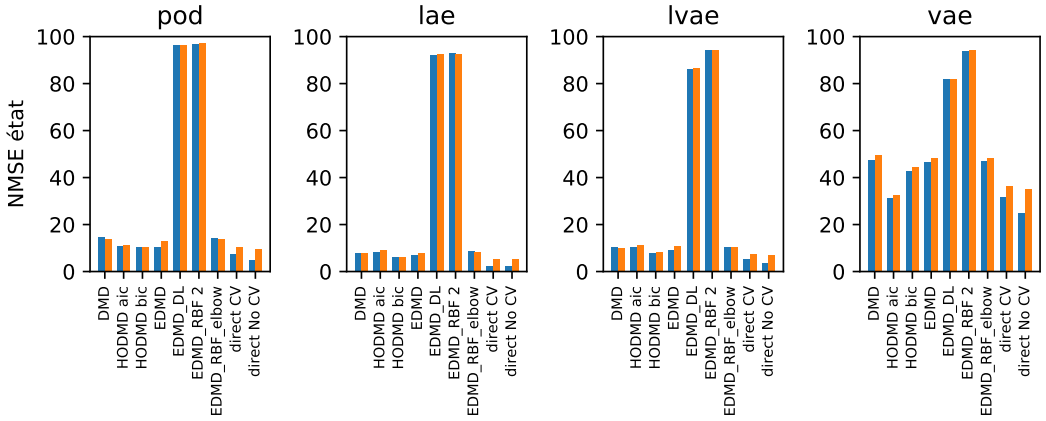


(c) Écoulement turbulent autour du cylindre

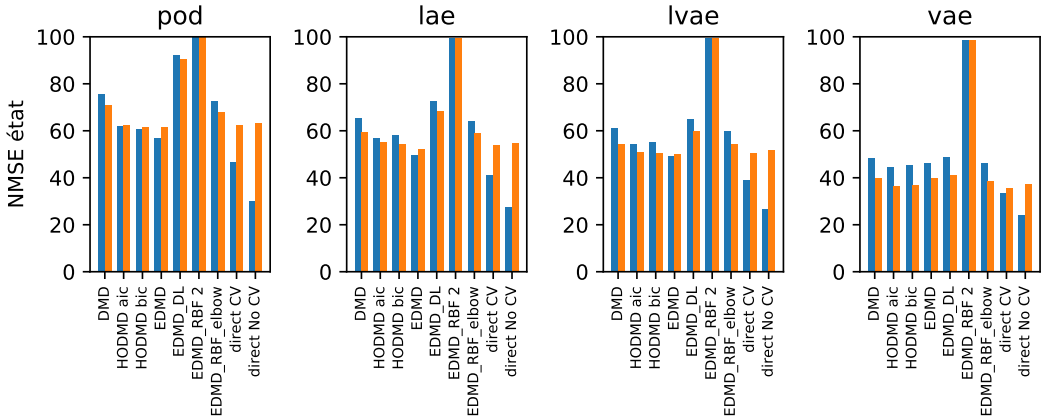
Figure *F.3* – *Erreur moyenne normalisée un pas de temps dans l'espace latent (en %).*



(a) Écoulement laminaire autour du cylindre



(b) Écoulement de couche de mélange



(c) Écoulement turbulent autour du cylindre

Figure *F.4* – *Erreur moyenne normalisée de prédiction dans l'espace latent (en %).*

G. Prédiction orientées données du système Lorenz 63

Cette annexe reprend un article publié dans *Physica D* en 2020. Ces travaux ont été effectué pendant la première de thèse pour se faire la main sur les outils de l'apprentissage automatique. On y présente l'utilisation de réseaux de neurones récurrents pour la prédiction du système chaotique de Lorenz 63. Les prédictions sont mises à jour via un réseau de neurones auxiliaire qui effectue une assimilation de données de manière régressive.

Ces travaux sont utiles pour deux questionnements dans le manuscrit. Au niveau de la **réduction**, nous avons soulevé le problème de régularité *en temps* des réductions variationnelles. L'idée serait d'utiliser des cellules récurrentes au niveau de la couche latente pour améliorer cette régularité. Au niveau de la **prédiction**, on s'est concentré sur l'approximation finie de l'opérateur de Koopman. Pour des systèmes complexes, trouver les bons observables est difficile, même avec des approches automatiques comme EDMD_DL. Dans ces circonstances et si l'interprétabilité des modèles ne prime pas, les réseaux de neurones récurrents avec cellules de mémoire sont de bonnes pistes.

Physica D 408 (2020) 132495

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Data-driven predictions of the Lorenz system

Pierre Dubois [a,*], Thomas Gomez [a], Laurent Planckaert [a], Laurent Perret [b]

[a] Univ. Lille, CNRS, ONERA, Arts et Metiers Institute of Technology, Centrale Lille, UMR 9014 - LMFL - Laboratoire de Mécanique des fluides de Lille - Kampé de Fériet, F-59000 Lille, France
[b] Centrale Nantes, LHEEA UMR CNRS 6598, Nantes, France

ARTICLE INFO

Article history:
Received 8 November 2019
Received in revised form 12 March 2020
Accepted 3 April 2020
Available online 10 April 2020
Communicated by T. Sauer

Keywords:
Data-driven modeling
Data assimilation
Chaotic system
Neural networks

ABSTRACT

This paper investigates the use of a data-driven method to model the dynamics of the chaotic Lorenz system. An architecture based on a recurrent neural network with long and short term dependencies predicts multiple time steps ahead the position and velocity of a particle using a sequence of past states as input. To account for modeling errors and make a continuous forecast, a dense artificial neural network assimilates online data to detect and update wrong predictions such as non-relevant switchings between lobes. The data-driven strategy leads to good prediction scores and does not require statistics of errors to be known, thus providing significant benefits compared to a simple Kalman filter update.

1. Introduction

Chaotic dynamical systems exhibit characteristics (nonlinearities, boundedness, initial condition sensitivity) [1] encountered in real-world problems such as meteorology [2] and oceanography [3]. The multiple time steps ahead prediction of such a system is challenging because governing equations may be unknown or too costly to evaluate. For instance, the Navier–Stokes equations require prohibitive computational resources to predict with great accuracy the velocity field of a turbulent flow [4].

Data-driven modeling of dynamical systems is an active research field whose objective is to infer dynamics from data [5]. Regressive methods in machine learning [6] are particularly suitable for such tasks and have proven to reliably reconstruct the state of a given system [7]. If parameters are not overfitted to training examples, the data-driven model can also be used for predictive tasks, provided the input lies in the input domain used for training. Main techniques in the literature include autoregressive techniques [8], dynamical mode decomposition (DMD) [9], Hankel alternative view of Koopman (HAVOK) [10] or unsupervised methods such as CROM [11]. Neural networks are also of increasing interest since they can perform nonlinear regressions that are fast to evaluate. Architectures with recurrent units are recommended for time-series predictions because memory is incorporated in the prediction process. Neural networks can then learn chaotic dynamics [12] and predict with great accuracy the future state [13].

However, errors in modeling can lead to bad multiple time steps ahead predictions of chaotic dynamical systems: a tiny change in the initial condition results in a big change in the output [12]. To overcome the propagation of uncertainties from the dynamical model (bad regression choice in a data-driven approach or bad turbulence modeling in CFD for instance) data assimilation (DA) techniques have been developed [14]. They combine the predicted state of a system with online measurements to get an updated state. Such methods have successfully been applied in fluid mechanics to obtain a better description of initial or boundary conditions by finding the best compromise between experimental measurements and CFD predictions [15]. Nevertheless, the dynamical model can be slow to evaluate (limiting the use to offline assimilations) and errors (initial condition, dynamical model, measurements and uncertainties) can be hard to estimate in real-world applications.

In this paper, a data-driven approach is used to discover a dynamical model for the Lorenz system. To handle the chaotic nature of the system, a recurrent neural network (RNN) dealing with long and short term dependencies (LSTM) is considered [16]. To correct modeling errors, a dense neural network (denoted hereafter DAN) whose design is based on Kalman filtering techniques is developed. Results are promising for predicting multiple steps ahead the position and velocity of a particle on the Lorenz attractor, using only the initial sequence and real-time measurements of the complete acceleration, the complete velocity or a single component of the velocity.

The paper is organized as follows. In Section 2, the overall strategy is presented with a quick understanding of how neural networks work. In Section 3, results about the low dimensional Lorenz system are shown, with a particular interest in the impact

* Corresponding author.
E-mail address: pierre.dubois@onera.fr (P. Dubois).

https://doi.org/10.1016/j.physd.2020.132495
0167-2789/

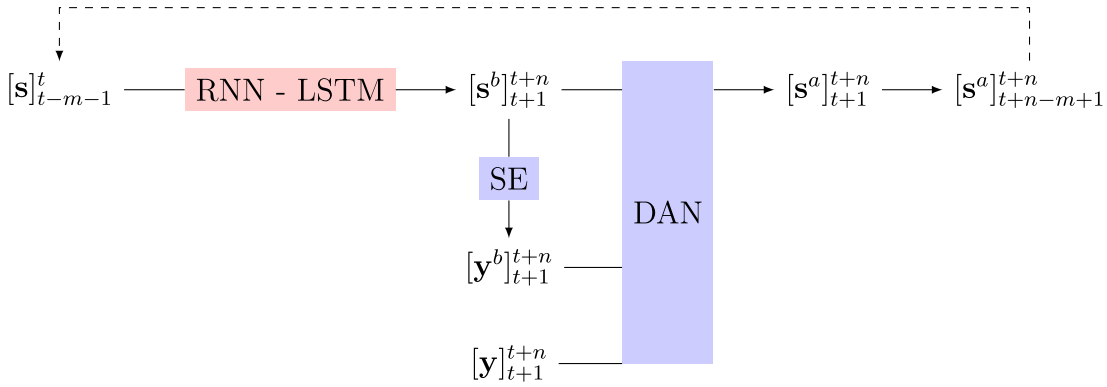


Fig. 1. Summary of the data-driven method to make predictions of a chaotic system. A data-driven dynamical model (RNN-LSTM) predicts n future states of the system and the predicted sequence is updated according to a real sequence of measurements.

of forecast horizon and noise. A discussion is given in Section 4 before giving concluding remarks.

2. Strategy

2.1. Proposed methodology

This paper investigates the use of neural networks to continuously predict a chaotic system using a data-driven dynamical model and online measurements. The method is summarized in Fig. 1 and contains the following steps:

- Consider m temporal states of the system. The sequence is denoted $[\mathbf{s}]^{t}_{t-m-1}$ where $\mathbf{s}$ is the state of the system and whose dimension is n_f.
- Predict n future states using a RNN with long and short-term memory (LSTM). This gives a predicted sequence $[\mathbf{s}^b]^{t+n}_{t+1}$ where superscript b indicates a prediction.
- Predict the measured sequence. This gives $[\mathbf{y}^b]^{t+n}_{t+1}$ where $\mathbf{y}^b$ is the predicted measure of the state. The mapping between the state space and the measurement space is performed by a dense neural network called the shallow encoder (SE).
- Assimilate the exact sequence of measurements $[\mathbf{y}]^{t+n}_{t+1}$ and update the predicted sequence of states. This work is performed by a dense neural network which gives an updated sequence $[\mathbf{s}^a]^{t+n}_{t+1}$ where superscript a stands for "analyzed". The network is called the data assimilation network (DAN).
- Construct $[\mathbf{s}^a]^{t+n}_{t+n-m+1}$ by adding $m-n$ updated states from the previous iteration. This gives a new input that can be used to cycle and continue the forecasting process.

In this section, we give a quick overview of neural networks and explain architectures behind the dynamical model (RNN-LSTM), the measurement operator (SE) and the data assimilation process (DAN).

2.2. Quick overview of neural networks

A neuron is a unit passing a sum of weighted inputs through an activation function that introduce nonlinearities. These functions are classically a sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$, a hyperbolic tangent $\tanh(x)$ or a rectified linear unit $\mathrm{relu}(x) = \max(0, x)$. When neurons are organized in fully connected layers, the resulting network is called a dense neural network. The universal approximation theorem [17] states that any function can be approximated by a sufficiently large network i.e. one hidden layer with a large number of neurons. Just like a linear regression $y = ax + b$ aims at learning the best a and b parameters, a neural network regression $y = NN(x)$ aims at learning the best weights and biases in the network by optimizing a loss function evaluated on a set of training data.

Although they are universal approximators, dense neural networks face some limitations: they may suffer from vanishing or exploding gradient (arising from derivatives of activation functions, see [18]), are prone to overfitting (fitting that corresponds too much to training data) and inputs are not individually processed. Other architectures of artificial neural networks have then been developed, including convolutional networks (CNN, for image recognition) or recurrent neural networks (RNN, inputs are taken sequentially). Recurrent networks use their internal state (denoted h) to process each input from the sequence of inputs. This internal state is computed using an activation function but to avoid limitations from dense networks, its form is more elaborate. For example, Long–Short-Term Memory (LSTM) cells [19] are combinations of classical activation functions (sigmoids and tanh) that incorporate a long and short term memory mechanism through the cell state (see Fig. 2).

Several techniques exist to learn parameters in neural networks. The most common is the gradient descent, which iteratively updates parameters according to the gradient of the cost function with respect to weights and biases. The computation of gradients is made by backpropagating errors in the network, using backpropagation for dense neural networks or backpropagation through time for RNN [6]. The equations can be found in [20] for the curious reader. In this paper, all neural networks are implemented using the Keras library [21].

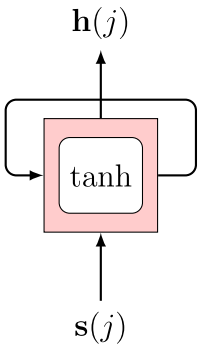
In this paper, hyperparameters are not tuned. No grid search or genetic optimization is intended and number of neurons, number of hidden layers and activation functions are found by successive trials. Defined architectures must not then be considered as a rule of thumb.

2.3. Novelty of the work

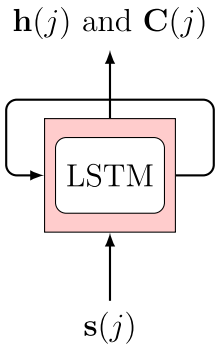
This paper proposes a regressive framework for assimilating data as opposed to standard data assimilation techniques whose architecture does not depend on the problem. Besides, the present paper considers time marching of an entire sequence of the state while the most standard approaches involve a time marching of the predicted state at regular time units. More details about existing works are given in Section 4.

2.4. Dynamical model

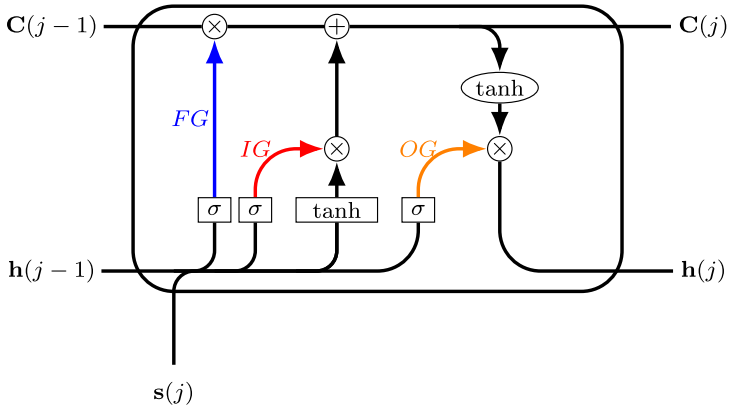
The first step is to establish a dynamical model mapping m previous states $\mathbf{s}(t)$ to n future states. The chosen architecture is summarized in Fig. 3. In the recurrent layer, $2m$ LSTM cells (making the cell state a $2m$ dimensional vector) process the input



(a) RNN



(b) RNN - LSTM



(c) LSTM cell. The recurrent unit is composed of a cell state and gating mechanisms. The cell state C is modified when fed with a new time step from the input sequence, forgetting past information (via Forget Gate FG), storing new information (via Input Gate IG) and creating a short-memory (via Output Gate OG). Mathematical details are given in the appendix.

Fig. 2. Two types of recurrent neural networks: simple RNN handling short-term dependencies via a hidden state **h** (subfigure a) and RNN-LSTM handling short and long-term dependencies via a hidden state **h**, a cell state **C** and gating mechanisms (subfigures b and c). Each time step **s**(j) from the input sequence is combined with **h**($j-1$) (and **C**($j-1$) for LSTM-RNN) which was (were) computed at previous time step.

sequence $[\mathbf{s}]_{t-m+1}^{t}$. This results in a final output $o(t) = h(t)$ summarizing all relevant information from the input sequence. In dense layers, the final output from the recurrent layer is used to predict n future states $[\mathbf{s}^b]_{t+1}^{t+n}$. Concerning the number of recurrent units, it has been chosen to echo results of Faqih et al. [1] where best scores were obtained by considering twice as many neurons than the history window. Authors made this conclusion after trying to predict multiple steps ahead the state of the Lorenz 63 system using a dense neural network with radial basis functions. About the training of the model, the procedure is as follows:

1. Simulate the system to get data $t \rightarrow \mathbf{s}(t)$. For the considered Lorenz system, only one trajectory is simulated but it covers a good region of the phase space.
2. Split data into training and testing sets. In this work, 2/3 of the data is used for the training and 1/3 is used for the testing. To detect possible overfitting, cross validation is first performed by considering 20% of training data as validation data. Depending on the evolution of both training and validation errors, dropout layers can be added to neural networks. The model is then trained on the whole training data set and errors are computed on the yet unseen testing data set. This step is necessary to ensure that test errors are representative of generalization errors.
3. Define supervised problems by writing data as $[\mathbf{s}]_{t-m+1}^{t} \rightarrow [\mathbf{s}]_{t+1}^{t+n}$. The number of training examples is increased by considering a sliding window of one time step i.e. training set is composed of $[\mathbf{s}]_{0}^{m-1} \rightarrow [\mathbf{s}]_{m}^{m+n-1}$, $[\mathbf{s}]_{1}^{m} \rightarrow [\mathbf{s}]_{m+1}^{m+n}$, etc. For the testing set, a sliding window of n time steps is used.
4. Find optimal weights and biases in the network by minimizing the mean square error evaluated on batches of training data. The chosen optimization algorithm is ADAM [22]. There are numerous parameters to find, including all weights and biases for each LSTM cell in the recurrent architecture and all parameters in dense layers. During the optimization process, the mean square error is also computed on the validation set. To avoid overfitting and ensure that weights and biases learned during training are relevant for future use on test set, errors evaluated on training and validation sets should be close.
5. Evaluate the performance of the final model using test data. Test 1 uses exact input sequences $[\mathbf{s}]_{t-m+1}^{t}$ to compute $[\mathbf{s}^b]_{t+1}^{t+n}$. Test 2 uses the first exact sequence $[\mathbf{s}]_{0}^{m-1}$ to

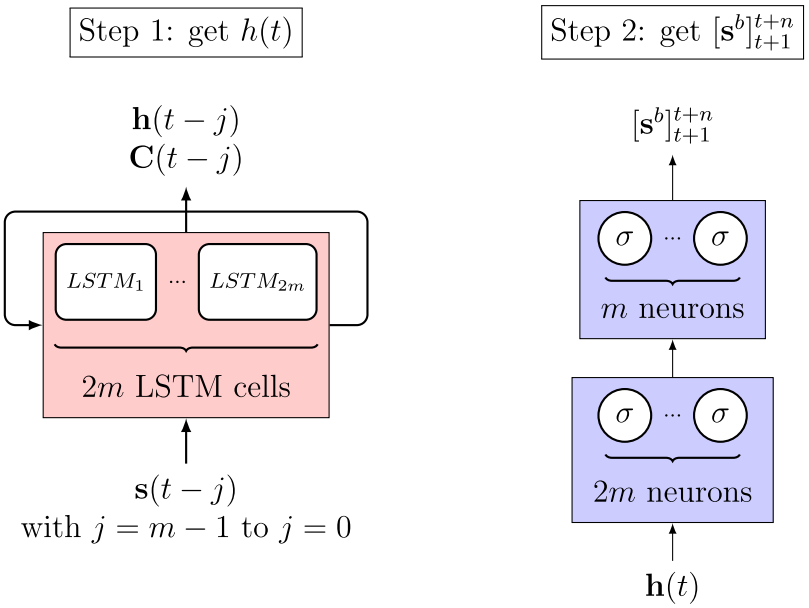


Fig. 3. Architecture of the dynamical model, a network composed of a recurrent layers and dense layers. Idea behind the design: $2m$ cells are used to echo the results obtained in [1] where best prediction scores were obtained when considering two times the history window.

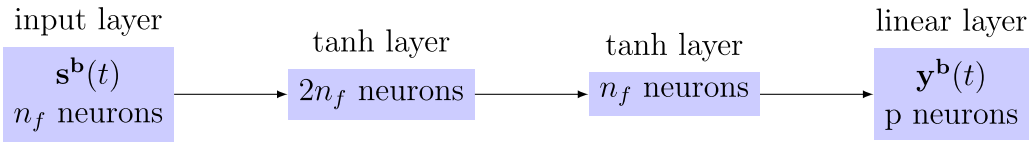


Fig. 4. Shallow artificial neural network to map a state to its measurement i.e. approximation of H operator.

compute $[\mathbf{s}^b]_m^{m+n-1}$ which is used as a new input and so on. The metric to quantify errors is the normalized mean square error which indicates how far predictions are from expectations on average. It is computed at the end of the process, using temporal testing states as a reference. Its formula is based on all predicted states $\mathbf{s}^b$ and corresponding true states $\mathbf{s}$:

$$NMSE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \frac{\|\mathbf{s}(t_i) - \mathbf{s}^b(t_i)\|^2}{\|\mathbf{s}(t_i)\|^2}$$

where n_{test} is the total number of test states and $\|.\|$ is the l_2 operator to compute the norm of the considered vector (dimension n_f).

2.5. Data assimilation

To make a continuous forecast of the state using a data-driven dynamical model, it is necessary to limit the accumulation of prediction errors [23] by incorporating online data in the prediction process. Consider $\mathbf{y}(t)$ an exact measurement of the state at t. The mapping between the state space and the measurement space is done using the measurement operator H. In Kalman filtering techniques, a predicted state $\mathbf{s}^b(t)$ is updated according to $\mathbf{s}^a(t) = \mathbf{s}^b(t) + K_t[\mathbf{y}(t) - H(\mathbf{s}^b(t))]$ where the Kalman gain K_t blends errors from the prediction and the measurement. Such a method is based on the Bayes theorem which helps to compute the density probability function of the state conditioned by the measurement. However, these techniques require statistics of errors to explicitly be known and work on a sequence of states only when considering the sequence as the state. The objective here is to adapt the strategy to directly update, in a regressive manner, a sequence of states using a sequence of measurements.

The first stage is to establish a relationship between the state and its measurement i.e. find an approximation of H operator. This task is performed by a shallow encoder which nonlinearly explains a measurement by its state. Fig. 4 summarizes the retained architecture, with n_f describing the number of features in the state and p being the number of observed variables. For simple and known mappings, this step is not necessary. However, to develop a complete data-driven framework, we choose to keep the shallow encoder despite the simplicity of the true measurement operator for the considered Lorenz system.

The training and testing of the shallow encoder are performed using data $\mathbf{s}(t) \rightarrow \mathbf{y}(t)$. The determination coefficient R^2 is used as a metric to assess the quality of the regression. It is defined by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n_s} \|\mathbf{y}(t_i) - \mathbf{y}^b(t_i)\|^2}{\sum_{i=1}^{n_s} \|\mathbf{y}(t_i) - \bar{\mathbf{y}}\|^2}$$

where n_s is the number of samples (n_{train} for assessing quality of the fit, n_{test} for assessing the quality of prediction) and $\bar{\mathbf{y}}$ is the temporal mean measurement vector.

The second stage is to blend a predicted sequence $[\mathbf{s}^b]_{t+1}^{t+n}$ with its associated sequence of measurements $[\mathbf{y}^b]_{t+1}^{t+n}$ and the real sequence of measurements $[\mathbf{y}]_{t+1}^{t+n}$ to produce the updated sequence $[\mathbf{s}^a]_{t+1}^{t+n}$. This job is done by a dense neural network whose architecture is summarized in Fig. 5. The training process is as follows:

1. Simulate the system to get $t \rightarrow \mathbf{s}(t), \mathbf{y}(t)$. Measurements are supposed to be exact and no noise is applied yet.
2. Split temporal states and measurements into training and testing sets. The procedure is the same as for the dynamical model training preparation.

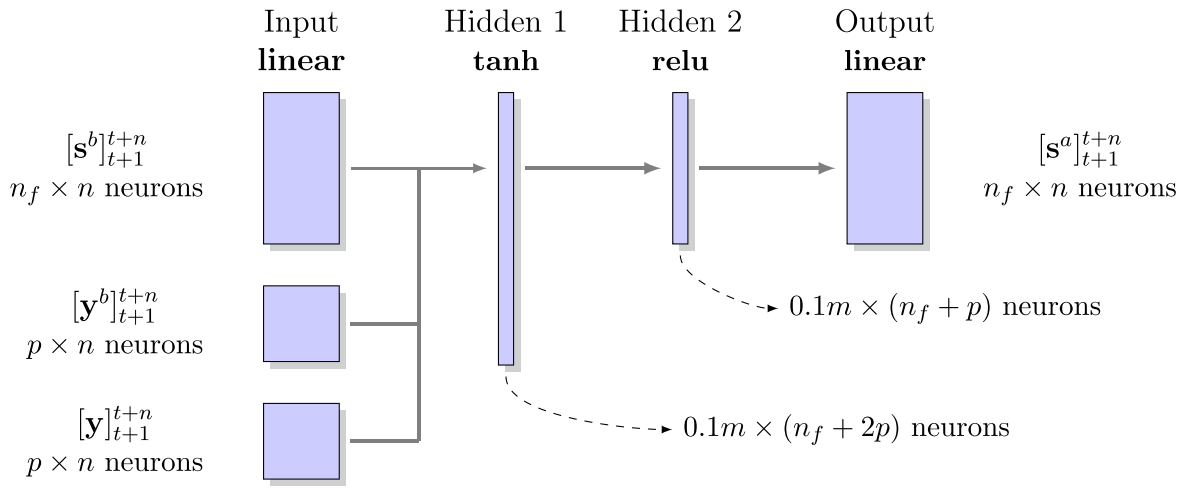


Fig. 5. Data assimilation network. The nonlinear regression correct predicted sequences of states by assimilating sequences of real measurements. Hyperparameters must not be considered as a rule of thumb and are well tailored for the Lorenz 63 system.

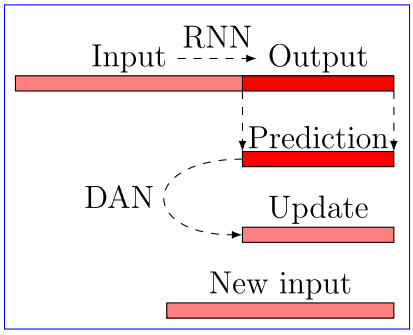


Fig. 6. Procedure to test the data assimilation network. The dynamical model is used to predict n future states of the system using a history of m states. The predicted sequence is updated using the data assimilation network and next input is formed. All predicted sequences are then compared to all expected sequences using the normalized mean square error as a metric.

3. Get supervised formulation of training data. The sliding window is supposed to be n. Outputs sequences are $[\mathbf{s}, \mathbf{y}]_{jn}^{(j+1)n+m-1}$ where index j describes the jth output.
4. Perform test 1 (see Section 2.4) and shallow encoder on defined outputs. This leads to possible predicted states and measurements $[\mathbf{s}^b, \mathbf{y}^b]_{jn}^{(j+1)n+m-1}$.
5. Each training measurement sequence $[\mathbf{y}]_{t+1}^{t+n}$ is associated to 20% random pairs from the set defined in step 4. In doing so, a real sequence of measurement is associated to randomly selected 20% of all possible predictions in order to produce sequence $[\mathbf{s}]_{t+1}^{t+n}$. All possible predictions could be used but this would increase the computational time to prepare the training set.
6. Perform gradient descent to optimize weights and biases.
7. Evaluate the performance of the final model using test data. This is denoted as Test 3 which is summarized in Fig. 6. Like in Test 2, inputs are fed back in recursively to assess the continuous prediction of test data. The metric is still the normalized mean square error.

The assimilation technique proposed here is a nonlinear regression learned on training data. This is different from Kalman filtering techniques where the Kalman gain only relies on statistics of errors and whose formula does not depend on the considered case.

3. Results

3.1. Lorenz system

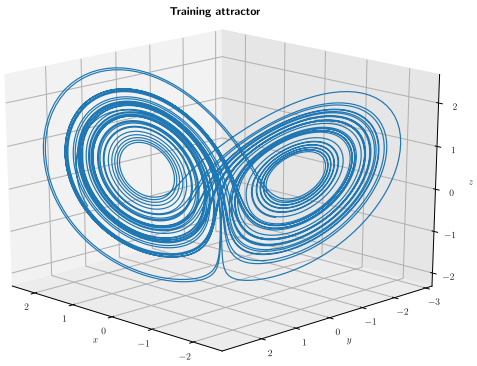
The Lorenz system of equations is a simplified model for atmospheric convection [2,24]. Close initial conditions lead to very different trajectories, making the Lorenz system a chaotic dynamical system. The system is defined by:

$$\begin{cases} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{cases}$$

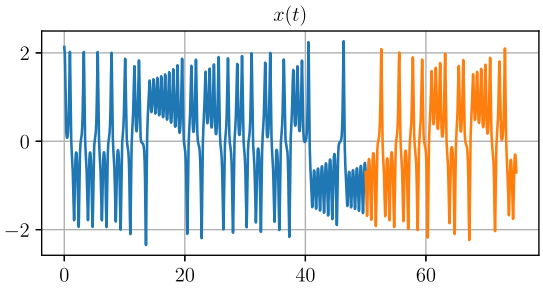
Parameters σ, ρ and β are respectively set to 10, 28 and 8/3. The trajectory of a particle lies in an attractor whose shape resembles a butterfly. In [10], Brunton proposed a method to write a chaotic system as a forced linear system. Following this method, forcing statistics appear non-gaussian, with long tails corresponding to rare intermitting forcing preceding switching events (see Figures 7(a) and 7(c)). The system is simulated using a Runge–Kutta 4 method, a random initial condition and a time step of 0.005s, for a total of 15000 samples. The chosen state is $\mathbf{s} = (x, y, z, \dot{x}, \dot{y}, \dot{z})$ which is the position and velocity of the particle on the attractor. The state is normalized using statistics from training data. The time-series of x, plotted in Fig. 7(b), clearly shows the lobe switching process (positive values when the particle travels on the right wing and negative values when it travels on the left wing). The objective is to extract from the simulated data a dynamical model mapping m past states to n future states. To account for modeling error, predictions are enforced using sequences of measurements. Our choice of observed variables is $\mathbf{y} = (\dot{x}, \ddot{y}, \ddot{z})$ or $\mathbf{y} = (\dot{x}, \dot{y}, \dot{z})$ or $y = \dot{x}$. Measurements are directly linked to the state and data-driven models should automatically detect these relations.

Before training models, the impact of n on the global error is investigated. Considering that wrong predictions are more likely to appear when predicting lobe switchings, two sources of errors, quantified on training data, can be found:

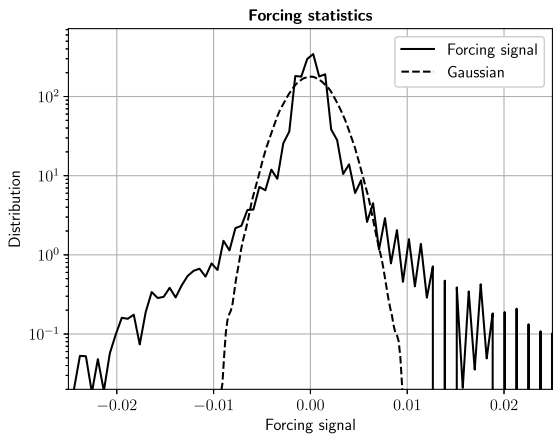
- Source 1, denoted $e_1 \rightarrow$ the ratio between the mean position of switching in a switching sequence and the prediction horizon n. The smaller the ratio, the bigger the impact on the global error.
- Source 2, denoted $e_2 \rightarrow$ the ratio between the number of sequences with switchings and the number of training sequences. The bigger the ratio, the bigger the impact on the global error.



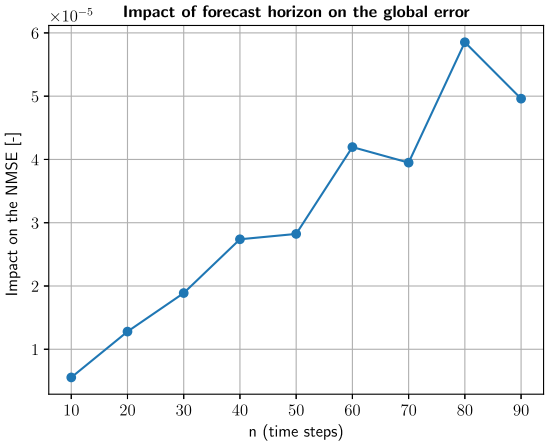
(a) Lorenz attractor



(b) Time-series of x feature.



(c) Distribution of forcing signal



(d) Impact of forecast horizon

Fig. 7. Analysis of training data. The attractor (Figure a) can be seen as a forced linear system with a non-gaussian distribution for the forcing signal (Figure c, method from [10]). Lobe switchings are visible in time-series of x feature (Figure b) where the blue signal corresponds to training data and the orange signal corresponds to testing data. The forecast horizon n has an impact on the global score as shown in subfigure d.

Fig. 7(d) shows the impact of global error for $n \in [10, 90]$. As expected, increasing the forecast window leads to a bigger impact on the global score (e_2/e_1 increasing) because prediction errors accumulate on longer sequences. However, the impact is not strictly monotone, indicating a dependence on the initial position of the particle. For the considered starting point, a forecast horizon $n = 80$ has a bigger impact on NMSE than for $n = 90$, indicating that lobe switchings (so possibly wrong predictions) are more likely to appear at the beginning of a new sequence to predict for $n = 80$ and in the middle of the sequence for $n = 90$. In next sections, a history window $m = 100$ to capture one lobe switching or zero.

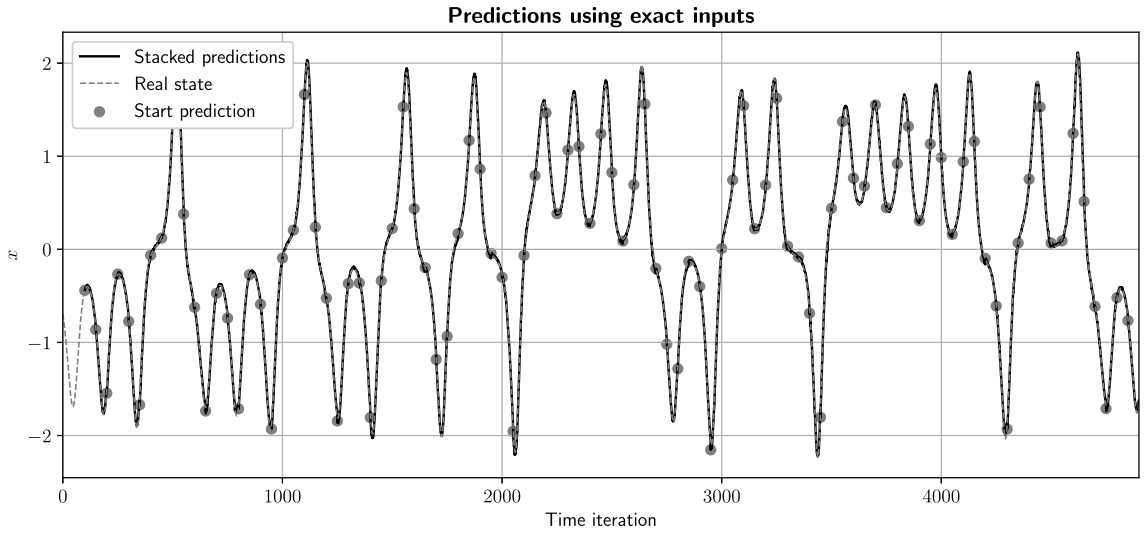
3.2. Testing the dynamical model

Nine dynamical models are established with $m = 100$ and $n \in [10, 90]$. Learning is stopped when the validation mean square error does not decrease for 3 epochs in a row. All learning curves show converged and close training and validation errors. The use of dropout layers or regularization techniques is then not necessary since no overfitting is noted. With models trained on the whole training data set, errors calculated on testing data are less than 1% for Test 1 but always exceed 100% for Test 2 (see step 5 in Section 2.4 for the definition of tests). It means that the dynamical model alone has a great performance only for small term predictions. Fig. 8 shows predictions of x feature for both tests with a forecast horizon of 50-time steps. It is worth noting that despite the global score for Test 2, the dynamical model successfully recovers the region of phase space it was learnt on.

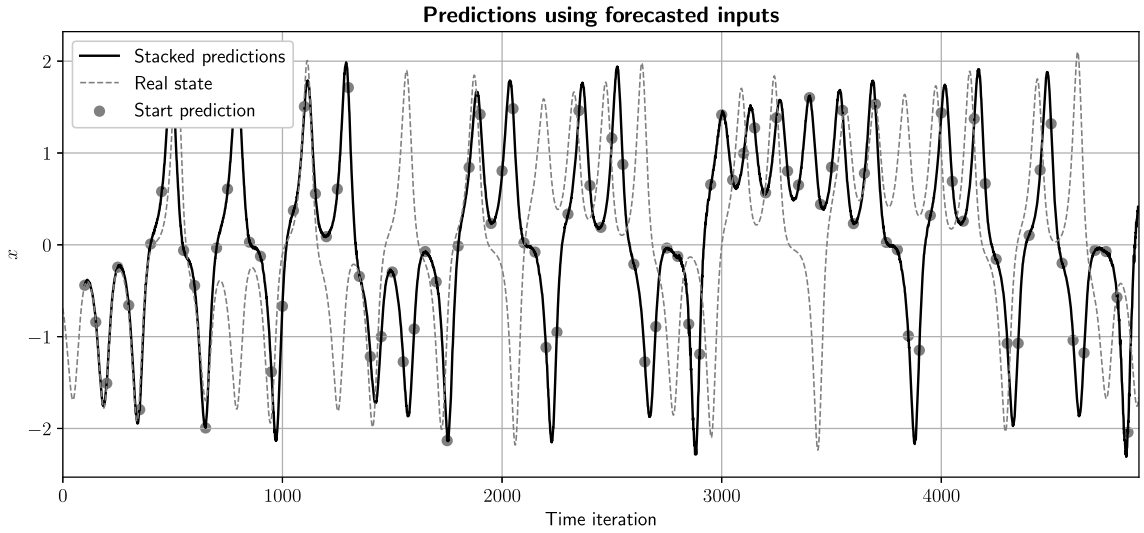
3.3. Testing the data assimilation network

Concerning the shallow encoder (to map a state to its measurement), training is extremely fast and accurate with a training and testing determination coefficient close to 1. It means that the nonlinear regression performed by the neural network recovers nearly all the variance observed in training and testing data. This is not a surprise since the relationship between $\mathbf{s}^n$ and $\mathbf{y}^b$ is simple (derivation for the acceleration or selecting features for velocity). Concerning the data assimilation network, quantitative results of Test 3 are shown in Fig. 9. Qualitative results for x and v_x predictions using acceleration are shown in Fig. 10. Several comments can be made:

1. Qualitatively, most of bad predictions are followed by good predictions: wrong predictions are detected and corrected by the DAN given online measurements.
2. Using the complete velocity leads to slightly better results than using the complete acceleration which seems reasonable because giving the velocity means giving three features out of six in the sequence to update.
3. Using only v_x leads to bad results for small sequences. To understand this behavior, the mean linear correlation



(a) Predictions of x (test set) when performing Test 1.



(b) Predictions of x (test set) when performing Test 2.

Fig. 8. Test of the dynamical model for $n = 50$. Dots indicate the start of a new prediction, using m past exact states (Test 1) or m past predicted states (Test 2) as input.

coefficient between sequences of v_x and sequences of the state has been studied. It is defined by:

$$\bar{r}(v_x, s_k) = \frac{1}{n_{s_{train}}} \sum_{i=1}^{n_{s_{train}}} r([v_x]_i, [s_k]_i)$$

$$\text{with } r(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

where $n_{s_{train}}$ is the number of output training sequences of size n, $[v_x]_i$ is the ith training sequence of v_x (size n) and $[s_k]_i$ is the ith training sequence of the kth feature of the state. Results are shown in Fig. 11. It appears that small sequences of v_x are linearly correlated to all features in the state (linear correlation coefficient close to 1), which is no longer the case for medium and large sequences where nonlinearities arise (linear correlation coefficient between 0.6 and 0.7). Therefore, the data assimilation network has a too complex architecture for updating small sequences: a lot of unnecessary parameters must be calculated during learning because the state could entirely be recovered by a linear regression on online v_x. This is a form overfitting and hyperparameters should be optimized but this is not the scope here.

4. Worst results are obtained for $n = 80$ which is linked to the dependence on the initial state to generate training data (Fig. 7(d)).

We now suppose that the initial sequence is noisy (Gaussian noise with standard deviation $\sigma_0 = 0.3$) just like online measurements (Gaussian noise with standard deviation $\sigma_y = 0.2$). The objective is to compare the proposed strategy with a simple Kalman filter update. Tests are restricted:

1. The simplest Kalman filtering technique requires the mapping between the state and its measurement to be linear (i.e. the H operator is simply a matrix). Tests will then concern v_x or the complete velocity as online measurements.

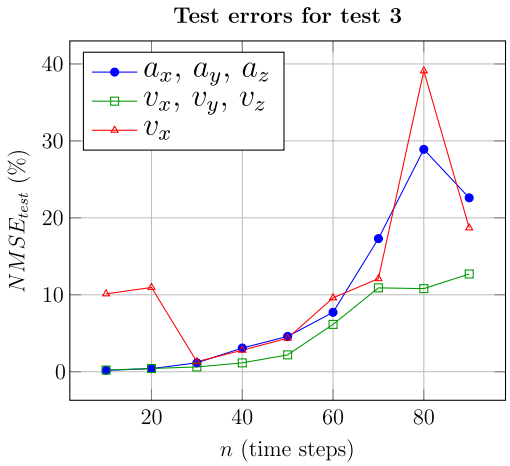


Fig. 9. Testing errors computed from Test 3 using the complete acceleration, the complete velocity or v_x alone to update predictions.

2. A Kalman filter requires the covariance of the prediction error to be advanced in time. In this paper, the covariance of the prediction error is supposed to be known at each new prediction and the predicted sequences are updated as a whole state.

Quantitative results are shown in Fig. 12. We can observe that the DAN performs better for medium and large sequences but has poor performance on small sequences compared to the Kalman filter. This result was expected: the limited size of the noisy input sequence makes it harder to detect and learn regularity, resulting in sub-optimal performance. This effect is not noticeable for large sequences because a pattern can still be detected in the noisy sequence. It is also worth noting that the noise applied to a small initial sequence has no influence on performance since no weights are attributed to predicted sequences in the DAN architecture. When considering the complete velocity, results obtained from the DAN are close to those obtained by Kalman filter (which requires the error to be known while the DAN does not). The influence of noise on small sequences is not as important as when using v_x alone since the correction does not rely on a single neuron.

4. Discussions

The data-assimilation framework proposed in this paper is based on regressive methods. Its success is then tailored to the good choice/design of the method and the relevance of training data. If the trajectory to predict lies in the learned phase space's region, one can expected the DAN to have great performance. Otherwise, Kalman filtering techniques which do not depend on the problem should be preferred. The reader is referred to Mons et al. [14] for an overview about existing techniques.

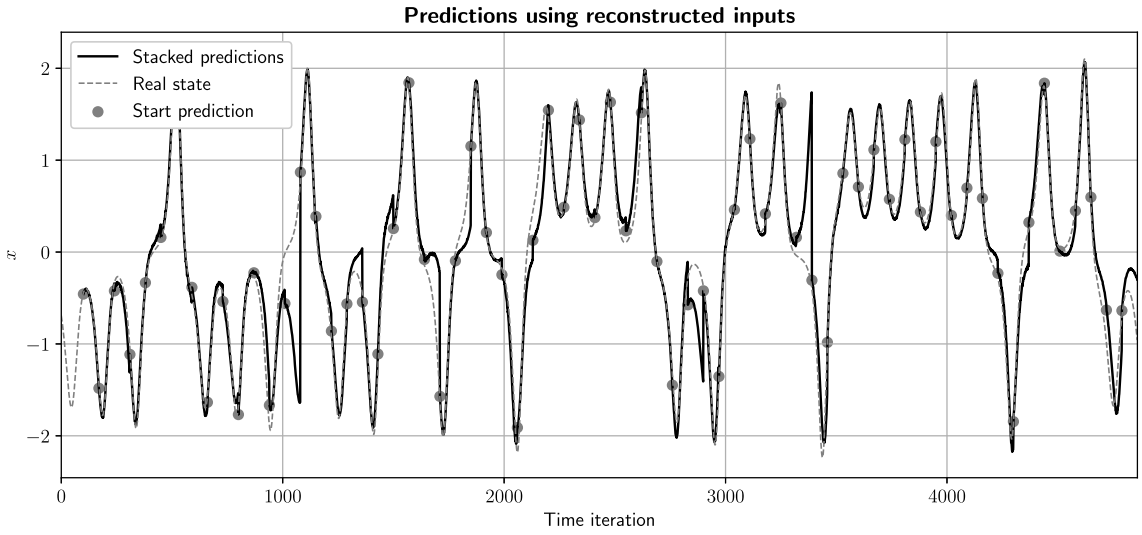
Concerning the combination of neural networks with Kalman filtering techniques, some works have already been done in the literature. Most innovations concern the estimation of uncertainties. For instance, Coskun et al. [25] use LSTM cells to predict the internals of the Kalman filter. In doing so, the authors implicitly learn a dynamical model and covariance errors to use for a Kalman update. In Loh et al. [23], authors update LSTM predictions of flow rates in gas wells using an ensemble Kalman filter, thus estimating errors via the covariance of an ensemble of predictions. In Becker et al. [26], a new architecture called recurrent Kalman network is developed: using an encoder–decoder network, authors are able to estimate uncertainties in the features. Finally, Vashista [27] directly train a RNN – LSTM network to simulate ensemble Kalman filter data assimilation using the differentiable architecture search framework. In this paper, we proposed a framework to directly work on sequences of data, separating the dynamical model from the data assimilation process. Future investigations could include the explicit estimation of uncertainties using for instance bootstrapping methods or gaussian processes [28].

Before applying the proposed strategy to a higher dimensional system, several challenges need to be addressed. First, the relevant phase region where to learn regressive models may be hard to detect as more features are involved. Second, optimal hyperparameters may not be easily guessable, thus requiring the need to grid search or genetic algorithm. Third, the dimensionality of the system could be reduced (by projecting it on a well defined basis) but some information would be sacrificed, thus raising the problem of what relevant features should be kept. Vlachas et al. [29] address some of the problems by establishing RNN-LSTM to forecast the Lorenz 96 system but no data assimilation coupled to their dynamical model has been yet intended.

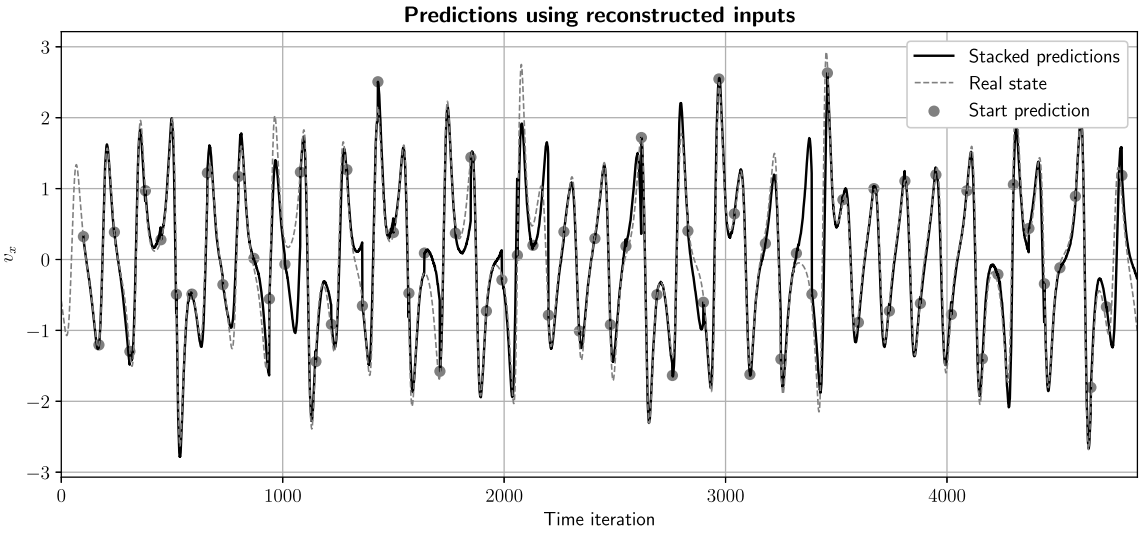
Finally, we are enthusiastic to use this framework for simple fluid mechanics problems: after reducing the state to estimate (using POD technique, see [30]), we aim to use data assimilation to continuously predict a simple flow. Neural networks involved in the dynamical or the data assimilation process will be improved by incorporating physics. For instance, a neural network with physical constraints is proposed by Ling [31] to establish a deep learning Reynolds Average Navier–Stokes model embedding the invariance of the anisotropy stress tensor.

5. Conclusion

In this paper, we investigated the use of neural networks to predict multiple steps ahead the state of the Lorenz system. The first stage consisted in establishing a dynamical model mapping previous states to future states. A recurrent neural network handling long and short-term dependencies was used for this purpose. Supposing the input sequence was exact, the output proved to be accurate with less than 1% error. However, when running the dynamical model with predicted states as new inputs, errors accumulated at each new prediction, leading to a bad prediction the time-series: the system being chaotic with extreme events, a small error in the initial condition leads to a radically different output. To overcome this accumulation of errors and make a continuous forecast of the Lorenz system, a data assimilation strategy based on sequential techniques was developed. This consisted in a nonlinear network mapping between a predicted sequence of states and a corresponding sequence of online measurements. This strategy proved to be effective when starting with the exact initial sequence and feeding the system with exact online measurements, notably when using the complete acceleration or the complete velocity. A deeper analysis of the DAN structure showed that this strategy was less relevant when using a single measurement or when working with small forecast windows. Besides, the DAN proved to be sensitive (at least for small forecast windows) to noise in measurements but not to noise in the initial condition. The DAN remains a good alternative to a simple Kalman filter where the estimation of errors may be a difficult task, especially when updating sequences. It nonetheless must be noted that the success of the DAN is mainly due to the quality of training data and extra care must be taken when learning regression parameters. All in all, the global strategy developed here seems promising to continuously forecast other chaotic systems evolving on an attractor. Future works could include the tuning of hyperparameters (to have an optimal design for each neural networks) and the application to a high dimensional attractor where, similarly to the Lorenz system, extreme events could be encountered.



(a) Predictions of x (test set) when performing Test 3.



(b) Predictions of v_x (test set) when performing Test 3.

Fig. 10. Test of the data assimilation network for $n = 70$. Dots indicate the start of a new prediction, using m past reconstructed states as input (test 3).

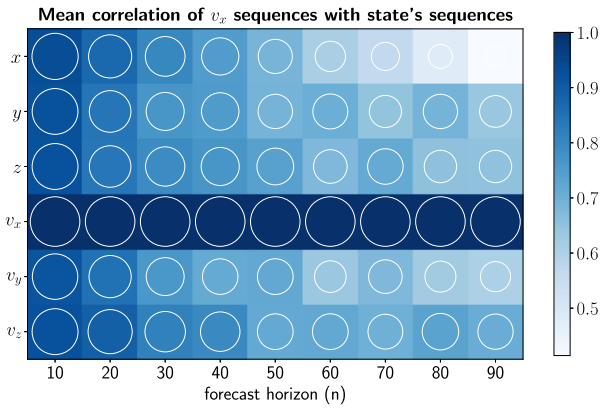


Fig. 11. Mean correlation coefficient between output training sequences of v_x measurements and each feature in output training sequences of the state. Small sequences of v_x measurements are linearly correlated to all features in small sequences of the state. Nonlinearities arise for higher forecast horizons.

Declaration of competing interest

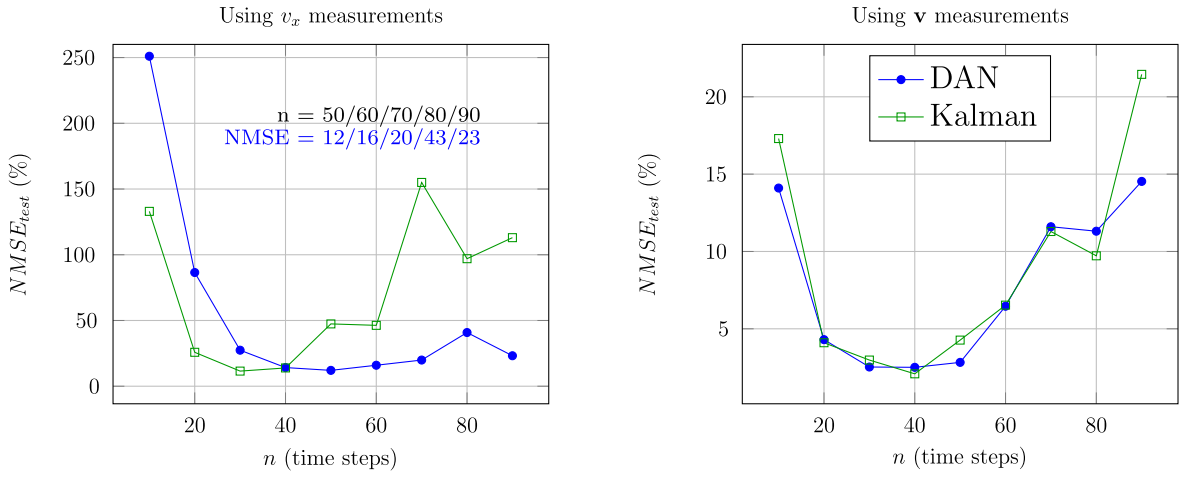
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Pierre Dubois: Investigation, Methodology, Writing. **Thomas Gomez:** Supervision, Investigation. **Laurent Planckaert:** Supervision. **Laurent Perret:** Supervision, review and editing.

Acknowledgments

The authors wish to thank ONERA, France and Région Hauts-De-France, France for their funding.



(a) Using noisy v_x as online measurement (b) Using noisy velocity as online measurement

Fig. 12. Results from Test 3 using a Kalman filter update or the data assimilation network. Noise was incorporated in both the initial sequence (starting point of the continuous forecast) and online measurements.

Appendix. LSTM cell

Long–Short Term Memory cells are recurrent units deploying a cell state and gating mechanisms. Equations of forget (f_t), input (i_t), output (o_t) and activation (a_t) gates are as follows:

$$\begin{cases} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ a_t &= \tanh(W_a x_t + U_a h_{t-1} + b_a) \end{cases}$$

where W are weights associated to the input x_t, U are weights associated to the hidden input h_{t-1} and b are biases. Outputs are the cell state c_t and the hidden state h_t, computed according to:

$$\begin{cases} c_t &= c_{t-1} \odot f_t + a_t \odot i_t \\ h_t &= o_t \odot \tanh(c_t) \end{cases}$$

where $\odot$ is the pointwise product. Given a new information $[x_t, h_{t-1}]$, the long-term memory c_t forgets information (via $f_t \odot c_{t-1}$) and stores a part of new information (via $a_t \odot i_t$). The short-term memory h_t depends on the long-term memory (via $\tanh(c_t)$) and the activation of the cell (via o_t) given new information.

References

[1] A. Faqih, B. Kamanditya, B. Kusumoputro, Multi-step ahead prediction of Lorenz's chaotic system using SOM ELM-RBFNN, in: 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), IEEE, 2018, pp. 1–5.

[2] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.

[3] J. Overland, J. Adams, H. Mofjeld, Chaos in the North Pacific: spatial modes and temporal irregularity, Prog. Oceanogr. 47 (2–4) (2000) 337–354.

[4] K.T. Carlberg, A. Jameson, M.J. Kochenderfer, J. Morton, L. Peng, F.D. Witherden, Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning, J. Comput. Phys. (2019).

[5] J. Kutz, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press, 2013.

[6] S. Brunton, B. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, 2019, arXiv preprint arXiv:1905.11075.

[7] N.B. Erichson, L. Mathelin, Z. Yao, S.L. Brunton, M.W. Mahoney, J.N. Kutz, Shallow learning for fluid flow reconstruction with limited sensors and limited data, 2019, arXiv preprint arXiv:1902.07358.

[8] C.-K. Ing, Multistep prediction in autoregressive processes, Econometric Theory 19 (2) (2003) 254–279.

[9] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28.

[10] S.L. Brunton, B.W. Brunton, J.L. Proctor, E. Kaiser, J.N. Kutz, Chaos as an intermittently forced linear system, Nature Commun. 8 (1) (2017) 19.

[11] E. Kaiser, B.R. Noack, L. Cordier, A. Spohn, M. Segond, M. Abel, G. Daviller, J. Östh, S. Krajnović, R.K. Niven, Cluster-based reduced-order modelling of a mixing layer, J. Fluid Mech. 754 (2014) 365–414.

[12] R. Yu, S. Zheng, Y. Liu, Learning chaotic dynamics using tensor recurrent neural networks, in: Proceedings of the ICML, vol. 17.

[13] J. Wang, Y. Li, Multi-step ahead wind speed prediction based on optimal feature extraction, long short term memory neural network and error correction strategy, Appl. Energy 230 (2018) 429–443.

[14] V. Mons, J.-C. Chassaing, T. Gomez, P. Sagaut, Reconstruction of unsteady viscous flows using data assimilation schemes, J. Comput. Phys. 316 (2016) 255–280.

[15] A. Gronskis, D. Heitz, E. Mémin, Inflow and initial conditions for direct numerical simulation based on adjoint data assimilation, J. Comput. Phys. 242 (2013) 480–497.

[16] J.-S. Zhang, X.-C. Xiao, Predicting chaotic time series using recurrent neural network, Chin. Phys. Lett. 17 (2) (2000) 88.

[17] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[18] R. Pascanu, T. Mikolov, Y. Bengio, Understanding the exploding gradient problem, CoRR, abs/1211.5063 2 (2012).

[19] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[20] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, S. Valaee, Recent advances in recurrent neural networks, 2017, arXiv preprint arXiv:1801.01078.

[21] F. Chollet, et al., Keras, GitHub, 2015, https://github.com/fchollet/keras.

[22] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[23] K. Loh, P.S. Omrani, R. van der Linden, Deep learning and data assimilation for real-time production prediction in natural gas wells, 2018, arXiv preprint arXiv:1802.05141.

[24] Z.-M. Chen, W. Price, On the relation between Rayleigh–Bénard convection and Lorenz system, Chaos Solitons Fractals 28 (2) (2006) 571–578.

[25] H. Coskun, F. Achilles, R. DiPietro, N. Navab, F. Tombari, Long short-term memory kalman filters: Recurrent neural estimators for pose regularization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5524–5532.

[26] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, J. Taylor, G. Neumann, Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces, 2019, arXiv preprint arXiv:1905.07357.

[27] H.V. Vashistha, RNN/LSTM Data Assimilation for the Lorenz Chaotic Models, University of Maryland, Baltimore County, 2018.

[28] X. Qiu, E. Meyerson, R. Miikkulainen, Quantifying point-prediction uncertainty in neural networks via residual estimation with an i/o kernel, 2019, arXiv preprint arXiv:1906.00588.

[29] P.R. Vlachas, W. Byeon, Z.Y. Wan, T.P. Sapsis, P. Koumoutsakos, Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 474 (2213) (2018) 20170844.

[30] A.T. Mohan, D.V. Gaitonde, A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks, 2018, arXiv preprint arXiv:1804.09269.

[31] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.

H. Assimilation de données

Cette annexe reprend un article de conférence pour la 3AF. Les travaux traitent de la dernière méthode d'estimation i.e. l'assimilation de données. Le modèle prédictif est l'approximation finie de l'opérateur de Koopman par décomposition modale dynamique. Le modèle de reconstruction est une régression par vecteurs supports. La combinaison de ces deux modèles, **purement appris à partir de données**, donnent de bons résultats pour l'estimation continue des quatre écoulements. Ce travail est également un bon résumé de toute la stratégie du manuscrit.

Association Aéronautique et Astronautique de France

55th 3AF International Conference on Applied Aerodynamics
12 — 14 April 2021, Poitiers – France

FP06-AERO2020+1-dubois

Data driven estimation of fluid flows: long-term prediction of velocity fields using machine learning

Pierre Dubois[(1)], Thomas Gomez [(1)], Laurent Planckaert [(1)] and Laurent Perret [(2)]
[(1)] Univ. Lille, CNRS, ONERA, Arts et Metiers Institute of Technology, Centrale Lille
UMR 9014 - LMFL - Laboratoire de Mécanique des fluides de Lille
Kampé de Fériet, F-59000 Lille, France
[(2)]Centrale Nantes, LHEEA UMR CNRS 6598, Nantes, France

ABSTRACT

This paper gives a framework for the data-driven estimation of an unsteady fluid flow field. The strategy combines machine learning tools for the reduction, the reconstruction and the prediction of the considered system. The reduction is performed by linear autoencoding while support vector regression and dynamical mode decomposition are respectively used as reconstruction and prediction models. Starting from an initial condition, reconstructions are frequently assimilated to update erroneuous predictions. The procedure is tested on four cases with increasing complexity and robustness is assessed through training and testing errors. Quantitative results suggests that reconstruction and prediction models **purely learnt from data** can be used for effective data assimilation, hence enabling the long-term prediction of even complex fluid flows.

1. INTRODUCTION

In fluid mechanics, each task (modelisation, closure, control or reduction) can be written as an optimization problem. However, for high Reynolds number, the convection term dominates the diffusion term, yielding a nonlinear, high-dimensional, multi-scale and nonconvex problem. Solving directly this formulation is challenging even intractable and new methods must be developed. Given the huge amount of both numerical and experimental data, a possibility is to use machine learning tools to solve optimization problems purely from data [5]. This paper investigates such data-driven procedures to estimate a fluid flow velocity field. In particular, a dynamical mode decomposition (DMD) model is used in combination with support vector machine regression (SVR) to continuously predict four fluid flows: 2D vortex shedding, a spatial mixing layer, 3D vortex shedding and an urban flow.

2. STATE OF ART

The field to estimate is denoted U_t. Two approaches are possible to obtain the estimate: the reconstruction and the prediction [6]. In the **reconstruction** problem, limited measurements y_t at time t are used to recover the velocity field at the same time (interpolation in space). In the **prediction** problem, a dynamical model is used to advance in time the velocity field U_{t-1} (extrapolation in time). For turbulent flows, the state is high-dimensional because of the complex spatio-temporal dynamics. However, low dimensional features can be extracted, making relevant the use of dimensionality reduction techniques [21]. Reconstruction and prediction problems are therefore equivalent to the estimation of the reduced (also called latent) state, as shown in figure 1.

2.1 Reconstruction

The measurement operator $\mathscr{H}$ being likely ill conditioned thus not invertible, the inverse operator $\mathscr{G} = d \circ f$ is estimated from data. Three ideas were developed in the literature. The first approach is the direct reconstruction [1], evaluating $\mathscr{G}(y)$ for each new measurement vector. The flow field is written as a linear combination of reference modes which can be generic (e.g. Fourier modes) or tailored to the considered flow (data driven modal decomposition). The second approach is the regressive reconstruction where the complete operator $\mathscr{G}$ is learned using supervised learning methods. Given a parametric

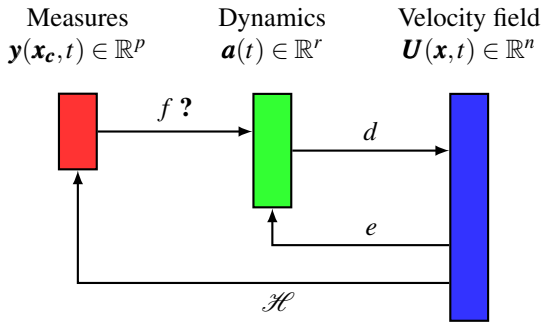


Figure 1: Representation of the reconstruction problem with dimensionality reduction: measurements are used to recover dynamics of dominant structures.

or nonparametric formulation of $\mathscr{G}$, a cost function evaluating the error between training examples (snapshots with associated measurements) and their reconstruction is minimized. First investigations of this method include the well-known stochastic estimation [2], where the reconstructed field is explained by a multi-linear function of available measurements. The third approach is data-assimilation where a dynamical model evolves the field estimate while measurements improve the forecasts [16]. The dynamical model may be a reduced-order approximation of Navier-Stokes equations, found by a Galerkin projection onto a data-driven basis or found by model identification.

2.2 Prediction

The velocity field satisfies a partial differential equation, namely Navier Stokes equations. To advance the state, the flow map is introduced, integrating the initial condition from t_0 to t [11]. Several strategies were developed to approximate this flow map in a data-driven fashion. A first approach consists in using supervised learning techniques to learn the input-output relation between past states and future states. Neural networks are particularly suitable given their high flexibility to capture nonlinearities. As an example, the reader is referred to [7] where we use a recurrent neural network with long-short term memory to continuously predict the chaotic Lorenz system. A second approach makes extensive use of the Koopman theory which introduces a linear but infinite dimensional operator to advance all possible observations of the state. Finite approximations of this so called Koopman operator give a linear dynamical model that can be used for prediction and control. Current research focus on finding a good space of observables where to learn the approximation or at least limit the spurious behaviour of identified eigenfunctions [14]. When working with latent state components as observables, the approximation of the Koopman operator is known to be the dynamical mode decomposition (DMD) model [22].

2.3 Reduction

Even highly turbulent flows exhibit low dimensional spatial directions called modes. Vortex shedding for wake flows, Kelvin Helmotz vortices for shear flows and coherent structures for boundary layers can be cited as examples. The extraction of such structures can be performed by linear or nonlinear encoding transformations e. If the decoding transformation d is known, the estimation of the latent structures dynamics $\boldsymbol{a}(t) \in \mathbb{R}^r$ is enough to infer the velocity field. The most common reduction technique is the proper orthogonal decomposition (POD) [19], which is the name given to principal components analysis applied to fluid flow data. Extracted modes are uncorrelated and hierarchically sorted by the data variability they recover. The simplicity of the implementation makes the POD a method of choice for dimensionality reduction. However, recent improvements in deep learning have made possible considerable progress in dimensionality reduction by using autoencoders. Two neural networks are therefore trained simultaneously to optimally encode and decode data. As a reference example, Xu et al. [23] took advantage of convolutional networks to leverage nested nonlinear manifolds and predict transient flows.

2.4 Work in this paper

This conference paper investigates the use of dynamical mode decomposition as a dynamical model and support vector regression as a reconstruction model to estimate four flow fields with increasing complexity: the flow in the wake of a 2D cylinder (Re = 200), a spatial mixing layer (Reynolds base on vorticity thickness Re = 500), the wake of a 3D cylinder (Re = 20000) and the flow in the viscinity of a tower placed in an atmospheric boundary layer (Reynolds base on the tower base length Re = 64000). Flow fields are reduced using a linear autoencoder and recursive forecasts of the latent state are sequentially enforced by the reconstructions. Data assimilation is performed using the models established solely from data, hence refered as data-driven assimilation. Figure 2 gives a general overview of the strategy.

3. METHODS AND DATA

This section gives details about notations, simulation data and mathematics behind the reduction, reconstruction, prediction and assimilation procedure.

3.1 Simulation data

Simulation data are written as a matrix $U \in \mathbb{R}^{n \times m}$ where n is the dimension (number of cells multiplied by the

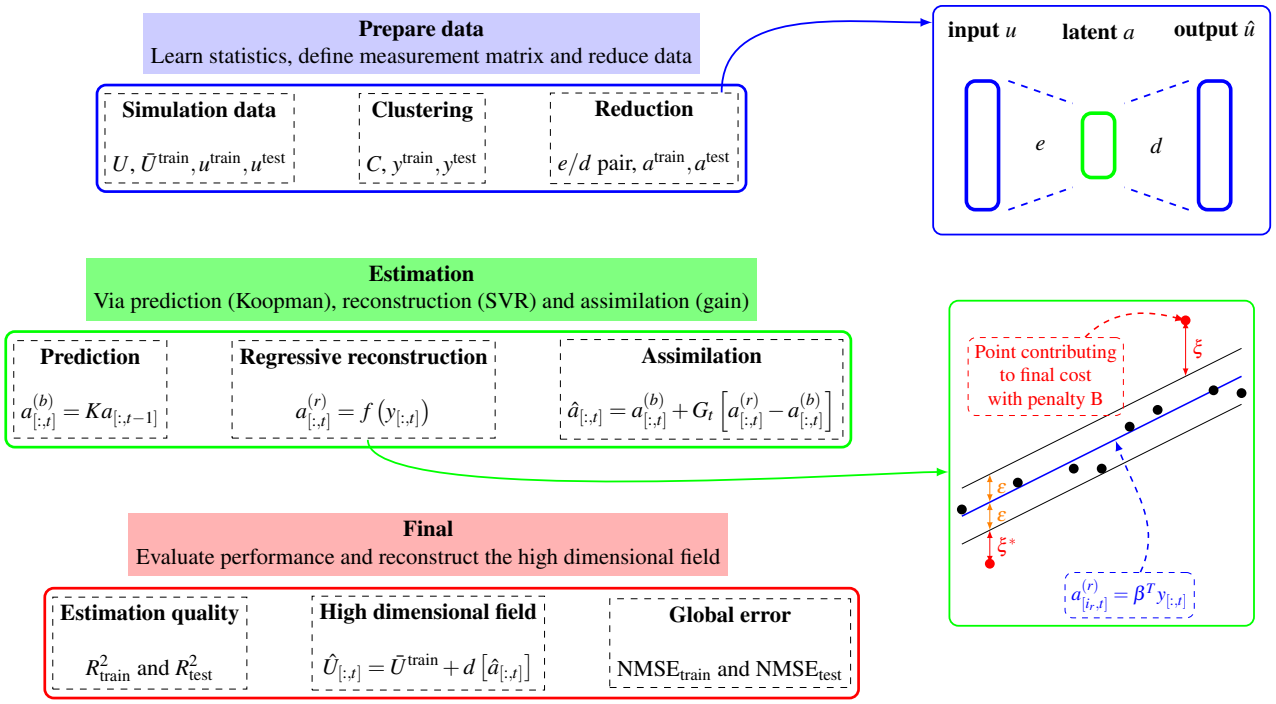


Figure 2: The proposed framework for the data-driven estimation of a fluid flow field.

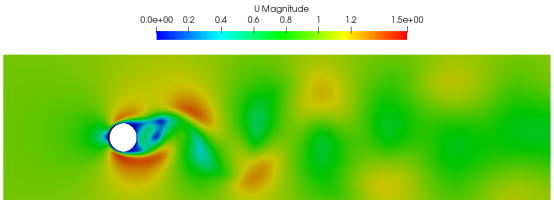
number of velocity components) and m is the number of snapshots. Considering a 70/30 split, $m^{\text{train}} = 0.7m$ and $m^{\text{test}} = 0.3m$ are respectively the number of training and testing snapshots. The fluctuant velocity field matrix is defined as $u = U - \bar{U}$ where $\bar{U}$ is the mean flow computed over all snapshots. This tall but skinny matrix is split into u^{train} and u^{test} matrices. The **case 0** corresponds to a URANS $k-\omega$ computation of a 2D cylinder placed in a uniform flow. The Reynolds number, based on the cylinder diameter, is $Re = 200$. The snapshot ensemble is composed of 125 snapshots written on a 5840 dimensional grid. The **case 1** corresponds to a direct numerical simulation of a 2D spatial mixing layer. The upper (fast) and lower (slow) stream velocities are respectively $U_1 = 30m/s$ and $U_2 = 10m/s$. The initial vorticity thickness is $\delta_{\omega_0} = 1m$ and the inlet profile is a hyperbolic tangent [12]. Stochastic perturbation is added to the inlet profile to trigger the Kelvin Helmotz instability [13]. A total number of 2700 snapshots is available, for a domain containing 3690 cells. The **case 2** corresponds to a large eddy simulation of a square cylinder wake. The Reynolds number based on the cylinder diameter is $Re = 20000$ and the snapshot ensemble contains 1312 snapshots for a domain with 48023 cells. The **case 3** corresponds to a large eddy simulation of the flow in the viscinity of a tower. A vortex method is used to reproduce the inlet turbulent velocity profile. The Reynolds number based on the tower width is $Re = 64000$. Estimations are performed in a transversal plane centered around the tower, which contains 28432 cells. A total number of 1596 snapshots is available. Figure 3 gives a visualisation of these flow fields.

3.2 Dimensionality reduction

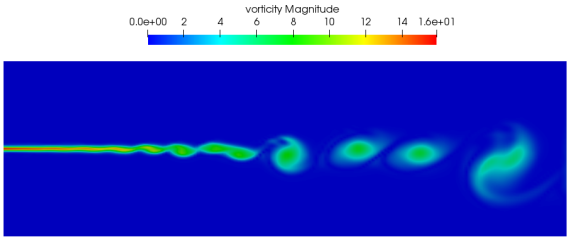
The encoder e compresses the data u from an initial space to a latent space and the decoder d decompresses encoded data. Given a family of candidate encoders E and decoders D, the best e/d pair is determined by:

$$(e^*, d^*) = \underset{(e,d)\in E\times D}{\arg\min}\ \varepsilon(u_{[:,t]}, d\left[e(u_{[:,t]})\right]) \qquad (1)$$

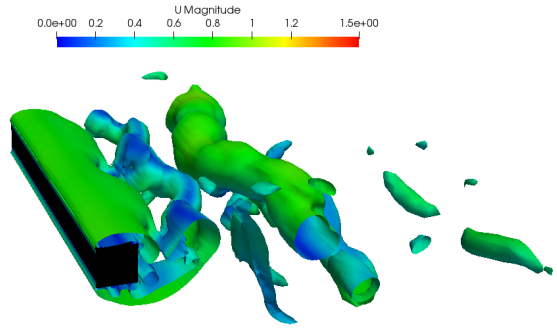
In proper orthogonal decomposition, the encoder and decoder are unitary matrices obtained from the spectral decomposition of the training covariance matrix $C_u = u^{\text{train}}[u^{\text{train}}]^T$. This decomposition yields: $C_u\Phi = \Phi\Lambda$ where the transfer matrix $\Phi \in \mathbb{R}^{n\times n}$ transforms initial basis vectors into uncorrelated directions. These modes are hierarchically sorted according to the variance Λ_{ii} they recover. When truncating the transformation to first r modes, initial data are written in the best r dimensional subspace to describe variability in u^{train}. In recent applications, encoders and decoders are neural networks. Here, we consider a linear autoencoder i.e., a neural network with one hidden layer and linear activations [18]. The weights and biases in the network are optimized by minimizing the mean square error between training samples and their autoencoding (see table 1). The dimension



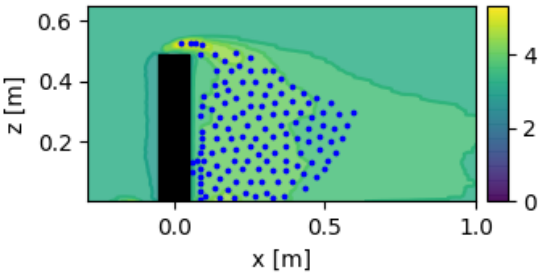
(a) Case 0 - 2D cylinder's wake colored by velocity magnitude (m/s)



(b) Case 1 - spatial mixing layer colored by vorticity magnitude (s^{-1})



(c) Case 2 - 3D cylinder with Q = 50 isosurfaces colored by velocity magnitude (m/s)



(d) Case 3 - plane in a urban flow, colored by the x componant of the root mean square velocity field (m/s). Sensors are also superimposed.

Figure 3: Visualisation of the four cases investigated in this communication.

	POD	**LAE**
Find	$\Phi_r \in \mathbb{R}^{n\times r}$	$W_1 \in \mathbb{R}^{n\times r}$ and $b_1 \in \mathbb{R}^r$ $W_2 \in \mathbb{R}^{r\times n}$ and $b_2 \in \mathbb{R}^n$
Encoding	$a_{[:,t]} = \Phi_r^T u_{[:,t]}$	$a_{[:,t]} = W_1 u_{[:,t]} + b_1$
Decoding	$\hat{u}_{[:,t]} = \Phi_r a_{[:,t]}$	$\hat{u}_{[:,t]} = W_2 a_{[:,t]} + b_2$
Noteworthy	Orthogonal modes $\Phi_r^T \Phi_r = I_{r\times r}$	Non orthogonal modes $W_1 = W_2^\dagger$
Modes	Φ_r	W_2
Method	SVD	Minimisation error $\varepsilon = \|\|u^{\text{train}} - \hat{u}^{\text{train}}\|\|_2^2$

Table 1: Autoencoding formulas

of the latent state is chosen accordingly with the number of POD modes required to recover 99% of the variance for case 0 and 80% for all other cases.

3.3 Sensor placement and reconstruction

Measurements are supposed to be p known locations in the fluctuant field to estimate. The measurement operator is a matrix $C \in \mathbb{R}^{p\times n}$ with $C_{ij} = 1$ if $y_i = u_j$ and 0 otherwise. Training and testing measurements are therefore $y^{\text{train}} = Cu^{\text{train}}$ and $y^{\text{test}} = Cu^{\text{test}}$. The spatial location of sensors is determined by enhanced clustering [10]. First, cell centres are partitioned into Voronoi cells defined by their centroids. Second, most energetic clusters are defined as sensors. In this paper, the number of clusters is set to 500 for all cases and sensors correspond to Voronoi centroids recovering 80% of the training field variance. The objective is now to learn the optimal mapping f so that $a^{(r)} = f(y)$ is a *good* estimate of the actual latent state a. This is a three-step procedure: choice of a form of a function, learning procedure to minimise a cost function and validation. In this paper, a focus is made on support vector regression (SVR). If this regression is performed in the latent state space, the estimation for the mode i_r is $a^{(r)}_{[i_r,t]} = \beta_{i_r}^T y_{[:,t]}$ i.e. a linear combination of measurements. Here, $\beta_{i_r} \in \mathbb{R}^{p\times 1}$ ensures at most an ε deviation from true targets and its optimal value is found by solving the primal formula:

$$\begin{cases} \min\limits_{\beta_{i_r},\xi,\xi^*} \dfrac{1}{2}\beta_{i_r}^T\beta + B\sum\limits_{t=1}^{m_{\text{train}}}(\xi_t + \xi_t^*) \\ a^{\text{train}}_{[i_r,t]} - \beta_{i_r}^T y^{\text{train}}_{[:,t]} \leq \varepsilon + \xi_t \ \forall t \\ \beta_{i_r}^T y^{\text{train}}_{[:,t]} - a^{\text{train}}_{[i_r,t]} \leq \varepsilon + \xi_t^* \ \forall t \\ \xi_t, \xi_t^* \geq 0 \ \forall t \end{cases} \tag{2}$$

With slack variables ξ and ξ^* to penalyze observations out of the ε tube. This regularization is controlled by the box constraint B. Instead of performing the linear regres-

sion in the latent state space, it can be perform in a higher even infinite dimensional subspace using the kernel trick. The estimation of a is therefore a nonlinear combination a measurements, yielding:

$$a^{(r)}_{[i_r,t]} = \sum_{t=1}^{m_{\text{train}}} (\alpha^*_t - \alpha_t) G(y^{\text{train}}_{[:,t]}, y_{[:,t]}) \tag{3}$$

Where G is a kernel function and (α, α^*) are Lagrange multipliers that intervene in the dual formulation of the problem. The kernel computes high dimensional interactions between variables y^{train} and y without actually transforming variables. The reader is referred to the tutorial of Smola [20] for a complete derivation of the equation and the cost function. To ensure that the model is robust on unseen data, hyperparameters of the SVR (the kernel and the box constraint) are cross-validated using a randomized grid search.

3.4 Dynamical mode decomposition

The dynamical mode decomposition finds the best one-step ahead linear dynamical model to describe the dynamics of data. The DMD matrix K is obtained by the Moore Penrose inverse:

$$K = \left[a^{\text{train}}\right]^{(+1)} \left[a^{\text{train}}\right]^{\dagger} \tag{4}$$

Where $\left[a^{\text{train}}\right]^{(+1)}$ is the time shifted version of a^{train}. This matrix can be used as a linear predictive model: starting from an initial condition a_{t_0}, the h-step ahead recursive forecast of the latent state is given by:

$$\hat{a}_{t_0+h} = K^h a_{t_0} \tag{5}$$

The K matrix is a finite approximation of the Koopman operator. If the dynamics of the latent state is nonlinear, identified eigenfunctions from left eigenvectors of K are spurious i.e. they do not evolved as predicted by their associated eigenvalues [4].

3.5 Data-driven assimilation

For nonlinear systems, the subspace spanned by latent state components is not Koopman invariant. Therefore, the dynamical model is erroneous and predictions must be updated. To make a continuous forecast of the latent state, the idea is to assimilate reconstructions. Three types of errors must be considered: the error on the initial condition, the dynamical model error and the reconstruction error. In this paper, reconstructions are assimilated each F = 5 new predictions. Covariance matrices of errors are defined as follows:

- $Q = \mathbb{E}\left[\left(a_{t_0+F} - K^F a_{t_0}\right)^2\right]$ for the recursive prediction using the dynamical mode. This matrix is estimated by a sample covariance, using training data.
- R for the covariance error on measurements. Noise is independantly applied to each component so that $R_{i_p,i_p} = \sigma^2_{i_p}$ with $\sigma_{i_p} = I_y \max\left[y_{[i_p,:]}\right]$.
- P_0 for the covariance error on the initial condition. Similarly to R, we define $P_{0_{i_r,i_r}} = \sigma^2_{i_r}$ with $\sigma_{i_r} = I_a \max\left[a_{[i_r,:]}\right]$.

The proposed assimilation scheme is based on Kalman filter [16] equations and is summarized in figure 4.

3.6 Metrics

To compare actual trajectories and estimated ones (reconstructed, predicted or assimilated), the normalized mean square error (NMSE) is used. Denoting s the true trajectory and $\hat{s}$ the estimated one, the error on component i for m_e samples is:

$$\text{NMSE}_i = \frac{\sum_{t=1}^{m_e} \left[s_{[i,t]} - \hat{s}_{[i,t]}\right]^2}{\sum_{t=1}^{m_e} \left[s_{[i,t]} - \overline{s}_{[i,:]}\right]^2} \tag{6}$$

A normalized error of zero means that on average, the error is small compared to the expected variability. The determination coefficient is also used, which is a "score" version of the NMSE metric, defined by:

$$R^2_i = 1 - \text{NMSE}_i \tag{7}$$

Metrics evaluated on training and testing data must be similar to ensure a good bias-variance trade-off, typically achieved by cross validation of hyperparameters. The global metric is determined by averaging over the number of modes. For the DMD results, training and testing data sets are split into *overlapping* trajectories with length $H = 16$. The h-step ahead prediction quality is quantified with a mean score over the number of trajectories while the global score also averages over the horizon[1]. For the assimilation results, metrics are evaluated by averaging over the ten *successive* trajectories that can be extracted from training or testing sets [2].

4. RESULTS

To reduce the dimension from n to $r \ll n$, the proper orthogonal decomposition and the linear autoencoder are used. The latent space dimension corresponds to the number of POD modes required to recover 99% of the variance for case 0 and 80% for other cases. Table 2

[1] DMD errors are evaluated on smaller sequences instead of the whole trajectory because the error would likely be 100% given the erroneous nature of the DMD model for nonlinear dynamics

[2] Assimilations are evaluated on ten smaller trajectories instead of the whole trajectory to test the procedure with different initial conditions.

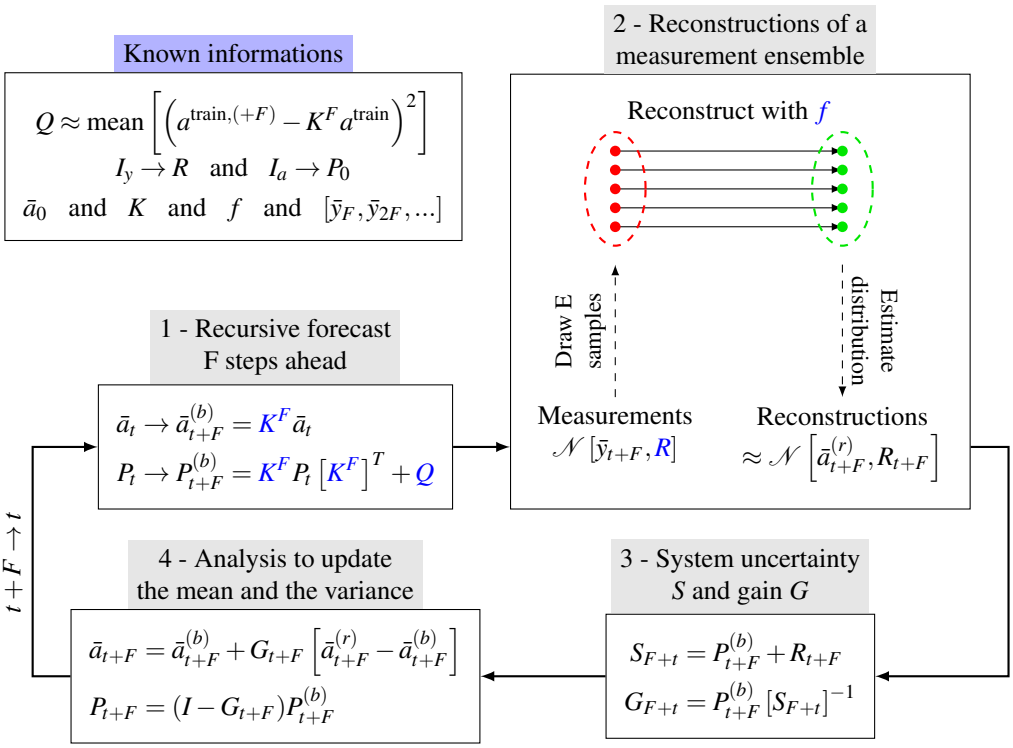


Figure 4: Data-assimilation procedure using data-driven models.

summarizes reduction results, including dimensions and autoencoding errors. As expected, the number of spatial modes and the error increases with the complexity of the flow. Testing errors are higher than training errors for case 2 and case 3 which is symptomatic of overfitted modes. To reduce this difference, robust principal components and cross validated neural network could be used, but this is out of the scope of this communication. Figure 5 gives a visualisation of the first POD mode (the most energetic) and the first LAE mode for the 2D cylinder. This comparison suggests that the nonorthogonal reduction is more interpretable than the orthogonal reduction, as supported by Erichson et al. conclusions [9].

Case	n	r	NMSE POD	NMSE LAE
0	11 680	5	[0.12 0.12]	[0.12 0.12]
1	27 360	11	[8.88 9.59]	[9.22 9.95]
2	144 069	21	[29.14 31.17]	[30.29 32.11]
3	85 296	139	[24.69 39.84]	[47.67 57.61]

Table 2: State space and latent space dimensions for each case and autoencoding errors. Results are written as [train test] errors (NMSE in %).

The clustering algorithm gives 500 centroids to optimally partition the mesh in an unsupervised fashion. Sensors are defined as the centroids that best describe the variability in training data. Considering the two (cases 0 and 1) or three (cases 2 and 3) components of the velocity field to estimate, the total number of sensors p is summarized in table 3. The choice of 500 clusters is arbitrary and could be optimized using heuristic criterions such as elbow and silouhette but this is not the scope here. In particular, the number of clusters for the 2D cylinder is undoubtedly high regarding the simple periodic behaviour of the flow. To learn the mapping between the measurement space and the latent state space with SVR, cross validation is performed. Tested hyperparameters are randomly chosen in a grid, for a total of 25 combinations and a 70% chance of hitting the optimal hyperparameter space [3]. Each mode is regressed independantly, yielding a total of 176 learned models to obtain reconstruction scores in table 3. For case 0, latent state components are perfectly estimated from measurements which is not surprising given the simplicity of the flow. Results are much more mitigated for case 3 where a strong overfit of training data is visible. To support this idea, figure 6 illustrates the reconstruction of the first latent state component, for the POD and the LAE methods. Reconstructions of training data (blue points) nearly recover all the expected variance, hence the close to unit determination score. For testing data, the determination coefficient (orange points) for each mode clearly respects the hierarchy imposed by POD reduction: first modes, corresponding to slow and coherent structures, are easier

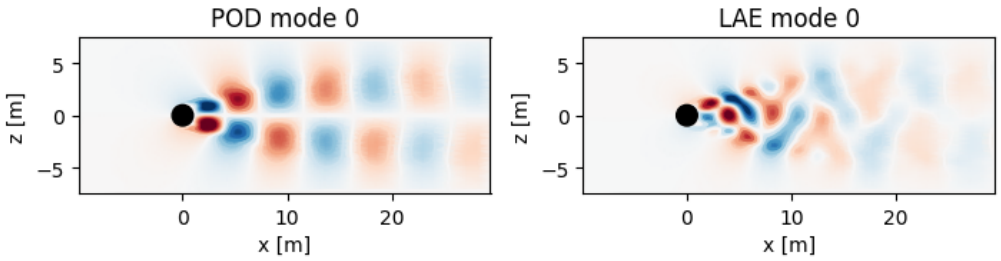


Figure 5: Comparison of a POD mode and a LAE mode for case 0. The vortex shedding is better captured with the nonorthogonal reduction.

to estimate than higher modes, corresponding to fast and small structures. At the opposite, the LAE modes are not hierarchically sorted and the estimation score is similar for each method. Overfitting being not a desirable property for robust fluid flow estimation, SVR doesn't seem to be tailored for the urban flow, and other regression techniques should be investigated.

Case	p	R^2 POD	R^2 LAE
0	92	[99.33, 99.31]	[99.5, 99.5]
1	122	[98.84 96.29]	[97.48 94.95]
2	372	[98.73 90.92]	[97.75 84.56]
3	393	[96.58 72.55]	[94.53 60.11]

Table 3: Reconstruction results for each case. Results are written as [train test] errors (R^2 in %).

Concerning the DMD model, normalized errors are summarized in table 4. The prediction error corresponds to the error integrated over the $H = 16$ horizon and the number of overlapping trajectories with length H in training or testing data. Interestingly, the latent state obtained with the LAE reduction for case 0 and case 1 is easier to predict with a linear model. Besides the global score, plotting the h-step ahead error as a function of h reveals how errors accumulate in the recursive process. Figure 7 gives an exemple when predicting testing sequences of the 3D cylinder with the POD reduction. The dynamics of the latent state being nonlinear, using the DMD matrix as a dynamical model for long term prediction is naturally erroneous, hence the bars of errors. This is confirmed by the green curves, corresponding to the h-step ahead score of each identified eigenfunctions. All of them are spurious (meaning they do not evolve as predicted by their eigenvalues) which is symptomatic of an observable subspace which is not Koopman invariant.

To correct predictions from the dynamical model, reconstructions are assimilated each $F = 5$ nondimensional time steps. Noise is applied to the initial condition a_0 and

Case	Prediction error	
	POD	LAE
0	[24.01 24.16]	[2.02 2.04]
1	[14.8 14]	[8.07 7.89]
2	[75.5 70.88]	[65.4 59.35]
3	[86.68 93.95]	[81.3 90.48]

Table 4: Errors in the recursive prediction of trajectories with length $H = 16$. NMSE values are obtained by averaging over the horizon and the number of overlapping trajectories with length $H = 16$ in training or testing data. Results are given as [train test] errors.

the measurement vector y, with intensity levels $I_a = 0.1$ and $I_y = 0.2$. Global scores are obtained by a mean over the number of modes and the one (case 0) or ten (other cases) *successive* trajectories in testing data. Results are given in table 5. It appears that using data assimilation results in a better long-term estimation of the latent state compared to the sole recursive forecast or the sole reconstruction at each time step. Figure 8 qualitatively supports this conclusion for the POD-reduced mixing layer.

Case	Dynamics	Reconstructions	Assimilation F = 5
0	[24.24 2.02]	[0.87 1.11]	[1.51 1.33]
1	[69.33 54.02]	[11.82 7.87]	[6.67 4.08]
2	[81.38 72.29]	[49.33 33.36]	[42.01 34.17]
3	[96.42 94.71]	[78.57 81.20]	[76.58 70.90]

Table 5: Comparison of errors for each estimation method (NMSE in %). These errors are averaged over the one (case 0) or ten (other cases) trajectories that can be extracted in the testing data. Results are written as [POD LAE] errors.

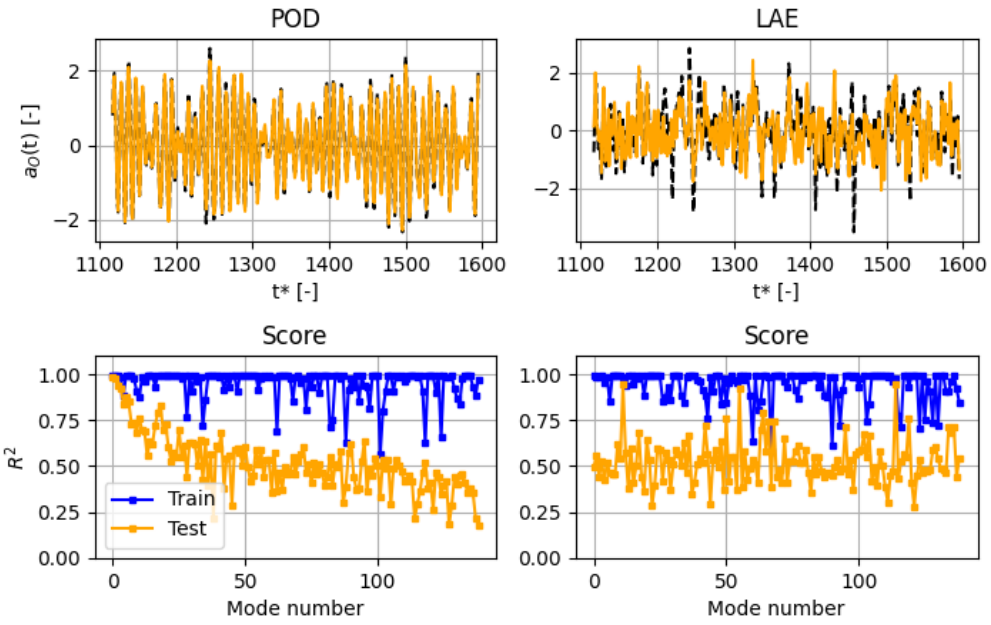


Figure 6: Reconstruction scores for case 3 and complete reconstruction of the first mode. For the POD reduction, large structures (first modes) are easier to reconstruct than small structures (last modes). For the LAE reduction, the modes are not hierarchically sorted and reconstructions scores are similar for all modes. A strong overfitting is visible, hence encouraging the use of other regression methods. Top figures: expected latent state in black and reconstructions in orange. Bottom figures: training $R^2_{i_r}$ in blue, testing in orange

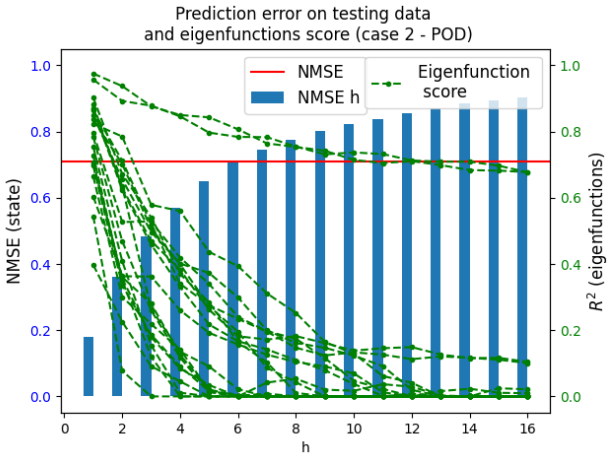


Figure 7: Bar plot corresponding to the h-step ahead error in the recursive prediction of testing data for case 2 with POD reduction. The h-step ahead score of each identified eigenfunction is also shown in green. The red line corresponds to the testing NMSE in table 4

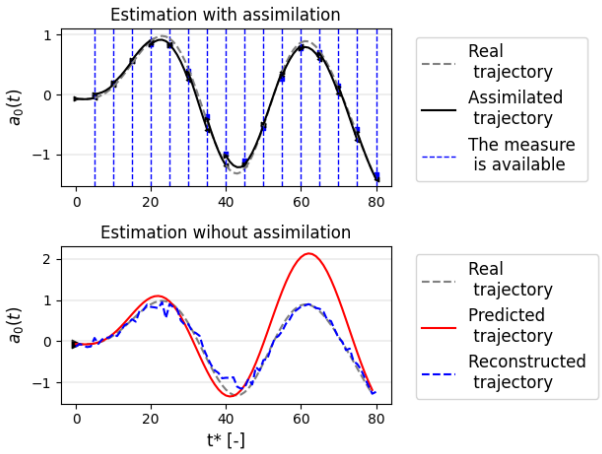


Figure 8: Exemple of the data-driven assimilation procedure on the mixing layer with POD reduction. Reconstructions are assimilated each $F = 5$ new predictions to update recursive forecasts by DMD.

Appart from the global error, the score as a function of the mode number can also be investigated. An exemple for the POD-reduced 3D cylinder is shown in figure 9. The curves quantitatively support the idea that on average, assimilated trajectories are better estimates than sole reconstructions or predictions. Concerning the POD-reduced 2D cylinder, changing the assimilation frequency

is of particular interest. This mode is badly predicted with a linear model but is perfectly reconstructed with measurements. With $F = 1$, the data assimilation procedure only gives credit to reconstructions (Kalman gain is the identity matrix) while for greater F, the assimilation is not enough frequently performed to significantly correct the mean determination coefficient. Quantitative results are given in table 6 to support this conclusion.

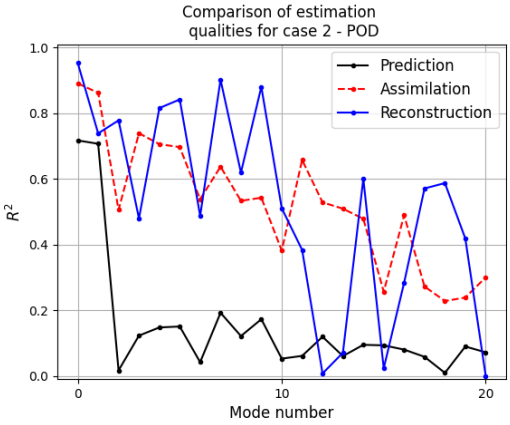


Figure 9: Comparison of estimation methods for testing trajectories of case 2 reduced with POD.

F	Reconstructions	Prediction	Assimilation
1			98.13
2	⋮	⋮	80.30
3	98.94	0.0	57.31
4			34.87
5	⋮	⋮	27.03
6			8.12

Table 6: Influence of the data assimilation frequency when estimating the fifth POD mode for case 0. Values correspond to the determination coefficient evaluated on the testing trajectory.

5. CONCLUSION

In this paper, different fluid flow estimation strategies are investigated on four increasing complexity cases. To reduce the dimension of the state to estimate, dominant spatial directions are extracted using the proper orthogonal decomposition or a linear autoencoder. Despite good results for the 2D cylinder, the mixing layer and the 3D cylinder, the reduction method clearly overfits the data for the urban flow, hence limiting the use of training modes for testing purposes. Latent states are then estimated by reconstruction and prediction. The reconstruction consists in using measurements of the fluctuant velocity field at current time to recover the latent state at the same time. This can be performed by a support vector regression, which independantly regresses each component of the latent state by available measurements. To respect a trade-off between the biais and the variance, hyperparameters are optimized by cross validation, leading to good training and testing scores for the 2D cylinder, the spatial mixing layer and the 3D cylinder. Results are much more mitigated on the urban flow, where modes dynamics are harder to reconstruct. The prediction consists in using a dynamical model to advance an initial condition in time. Dynamical mode decomposition is used for that purpose, to learn the one-step ahead linear model that optimally describes the dynamics of data. When used as a long-term predictive model, errors accumulate because the latent space is not Koopman invariant. To avoid this accumulation of errors, data assimilation is then investigated: reconstructions are sequentially used to update predictions at a frequency F. By blending the benefits of the reconstruction and the prediction models, data-assimilation enables the long-term prediction of the fluid flow field. The procedure being purely based on machine learning, it is a promising technique for estimating any fluid flow field where data is available. Further investigations on academic cases could include generative modeling [17] to account for inlet parameters (e.g. Reynolds number, turbulent intensity), the use of probabilistic models such as CROM [12] or the use of balanced truncation for sensor placement [15]. An extensive study of the reconstruction problem for the first three cases was submitted to *Journal of Computational Physics* [8]. For the reduction part, POD and variationnal autoencodeurs were considered. For the reconstruction part, linear multitask regression, SVR, neural network and gradient boosting decision trees were used. Results suggest that encoding velocity fields as distributions instead of single points improve robustness when decoding a latent state estimate. Another conclusion concerns the performance of each reconstruction method: using cross validation enable similar results for all methods so that the choice of one regression model towards another depends on the quality of the data, the interpretability and the cost of implementation/computation. Conclusions drawn from this study provide valuable informations for the development of new estimation techniques based on machine learning and their deployment on complex geometries that can be encoutered in industrial issues.

REFERENCES

[1] SM Al Mamun, Chen Lu, and Balaji Jayaraman. Extreme learning machines as encoders for sparse reconstruction. *Fluids*, 3(4):88, 2018.

[2] Anthony Arnault, Julien Dandois, J-C Monnier, Jérôme Delva, and J-M Foucaut. Analysis of the filtering effect of the stochastic estimation and accuracy improvement by sensor location optimization. *Experiments in Fluids*, 57(12):1–22, 2016.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Steven L Brunton, Bingni W Brunton, Joshua L Proctor, and J Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2):e0150171, 2016.

[5] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[6] Jared L Callaham, Kazuki Maeda, and Steven L Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907, 2019.

[7] Pierre Dubois, Thomas Gomez, Laurent Planckaert, and Laurent Perret. Data-driven predictions of the lorenz system. *Physica D: Nonlinear Phenomena*, 408:132495, 2020.

[8] Pierre Dubois, Thomas Gomez, Laurent Planckaert, and Laurent Perret. Machine learning for fluid flow reconstruction from limited measurements. *Journal of Computational Physics, submitted*, 2021.

[9] N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

[10] Balaji Jayaraman, Chen Lu, Joshua Whitman, and Girish Chowdhary. Sparse feature map-based markov models for nonlinear fluid flows. *Computers & Fluids*, 191:104252, 2019.

[11] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of koopman eigenfunctions for control. *arXiv preprint arXiv:1707.01146*, 2017.

[12] Eurika Kaiser, Bernd R Noack, Laurent Cordier, Andreas Spohn, Marc Segond, Markus Abel, Guillaume Daviller, Jan Östh, Siniša Krajnović, and Robert K Niven. Cluster-based reduced-order modelling of a mixing layer. *arXiv preprint arXiv:1309.0524*, 2013.

[13] Jordan Ko, Didier Lucor, and Pierre Sagaut. Sensitivity of two-dimensional spatially developing mixing layers with respect to uncertain inflow conditions. *Physics of Fluids*, 20(7):077102, 2008.

[14] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[15] Krithika Manohar, J Nathan Kutz, and Steven L Brunton. Optimal sensor and actuator selection using balanced model reduction. *arXiv preprint arXiv:1812.01574*, 2018.

[16] Vincent Mons, J-C Chassaing, Thomas Gomez, and Pierre Sagaut. Reconstruction of unsteady viscous flows using data assimilation schemes. *Journal of Computational Physics*, 316:255–280, 2016.

[17] Jeremy Morton, Mykel J Kochenderfer, and Freddie D Witherden. Parameter-conditioned sequential generative modeling of fluid flows. *AIAA Journal*, pages 1–17, 2021.

[18] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

[19] Clarence W Rowley and Scott TM Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[20] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[21] Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *Aiaa Journal*, 55(12):4013–4041, 2017.

[22] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[23] Jiayang Xu and Karthik Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020.

Table des figures

Liste des tableaux

Bibliographie

Adrian, R. J. On the role of conditional averages in turbulence theory. 1975.

Adrian, R. J. Conditional eddies in isotropic turbulence. *The Physics of Fluids*, 22 (11) :2065–2070, 1979.

Adrian, R. J. et Moin, P. Stochastic estimation of organized turbulent structure : homogeneous shear flow. *Journal of Fluid Mechanics*, 190 :531–559, 1988.

Al Mamun, S., Lu, C., et Jayaraman, B. Extreme learning machines as encoders for sparse reconstruction. *Fluids*, 3(4) :88, 2018.

Arnault, A., Dandois, J., Monnier, J.-C., Delva, J., et Foucaut, J.-M. Analysis of the filtering effect of the stochastic estimation and accuracy improvement by sensor location optimization. *Experiments in Fluids*, 57(12) :1–22, 2016.

Attili, A. et Bisetti, F. Statistics and scaling of turbulence in a spatially developing mixing layer at Re_λ= 250. *Physics of Fluids*, 24(3) :035109, 2012.

Bagheri, S. Koopman-mode decomposition of the cylinder wake. *Journal of Fluid Mechanics*, 726 :596–623, 2013.

Bai, H. et Alam, M. M. Dependence of square cylinder wake on reynolds number. *Physics of Fluids*, 30(1) :015102, 2018.

Bailly, C. et Comte-Bellot, G. Introduction to turbulence. In *Turbulence*, pages 1–31. Springer, 2015.

Baldi, P. et Hornik, K. Neural networks and principal component analysis : Learning from examples without local minima. *Neural networks*, 2(1) :53–58, 1989.

Balsa, T. F. On the receptivity of free shear layers to two-dimensional external excitation. *Journal of Fluid Mechanics*, 187 :155–177, 1988.

Barlow, J. F. Progress in observing and modelling the urban boundary layer. *Urban Climate*, 10 :216–240, 2014.

Beal, T. Digital simulation of atmospheric turbulence for dryden and von karman models. *Journal of Guidance, Control, and Dynamics*, 16(1) :132–138, 1993.

Becker, S., Lienhart, H., et Durst, F. Flow around three-dimensional obstacles in boundary layers. *Journal of Wind Engineering and Industrial Aerodynamics*, 90 (4-5) :265–279, 2002.

Belkin, M., Hsu, D., Ma, S., et Mandal, S. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv :1812.11118*, 2018.

Bergstra, J. et Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Berkooz, G., Holmes, P., et Lumley, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1) :539–575, 1993.

Bessonov, N., Sequeira, A., Simakov, S., Vassilevskii, Y., et Volpert, V. Methods of blood flow modelling. *Mathematical modelling of natural phenomena*, 11(1) : 1–25, 2016.

Biau, G. et Scornet, E. A random forest guided tour. *Test*, 25(2) :197–227, 2016.

Bieker, K., Peitz, S., Brunton, S. L., Kutz, J. N., et Dellnitz, M. Deep model predictive control with online learning for complex physical systems. *arXiv preprint arXiv :1905.10094*, 2019.

Biferale, L., Bonaccorso, F., Buzzicotti, M., Clark Di Leoni, P., et Gustavsson, K. Zermelo's problem : Optimal point-to-point navigation in 2d turbulent flows using reinforcement learning. *Chaos : An Interdisciplinary Journal of Nonlinear Science*, 29(10) :103–138, 2019.

Bishop, C. M. et James, G. D. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, 327(2-3) :580–593, 1993.

Bishop, C. M. et Tipping, M. E. Bayesian regression and classification. *Nato Science Series sub Series III Computer And Systems Sciences*, 190 :267–288, 2003.

Blackman, K., Perret, L., Calmet, I., et Rivet, C. Turbulent kinetic energy budget in the boundary layer developing over an urban-like rough wall using piv. *Physics of Fluids*, 29(8) :085113, 2017.

Blocken, B. Les over rans in building simulation for outdoor and indoor applications : a foregone conclusion ? 11(5) :821–870, 2018.

Bonnet, J. P., Cole, D. R., Delville, J., Glauser, M. N., et Ukeiley, L. S. Stochastic estimation and proper orthogonal decomposition : complementary techniques for identifying structure. *Experiments in fluids*, 17(5) :307–314, 1994.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks : Tricks of the trade*, pages 421–436. Springer, 2012.

Braza, M., Chassaing, P., et Minh, H. H. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of fluid mechanics*, 165 :79–130, 1986.

Breiman, L., Friedman, J., Stone, C. J., et Olshen, R. A. *Classification and regression trees*. CRC press, 1984.

Brochu, E., Cora, V. M., et De Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv :1012.2599*, 2010.

Brown, G. L. et Roshko, A. On density effects and large structure in turbulent mixing layers. *Journal of Fluid Mechanics*, 64(4) :775–816, 1974.

Brunton, S. L. et Kutz, J. N. *Data-driven science and engineering : Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

Brunton, S. L., Brunton, B. W., Proctor, J. L., et Kutz, J. N. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2) :e0150171, 2016a.

Brunton, S. L., Proctor, J. L., et Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15) :3932–3937, 2016b.

Brunton, S. L., Proctor, J. L., et Kutz, J. N. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49 :710–715, 2016c.

Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E., et Kutz, J. N. Chaos as an intermittently forced linear system. *Nature communications*, 8(1) :1–9, 2017.

Brunton, S. L., Noack, B. R., et Koumoutsakos, P. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52 :477–508, 2020.

Buschmann, M. H. et Gad-el Hak, M. Debate concerning the mean-velocity profile of a turbulent boundary layer. *AIAA journal*, 41(4) :565–572, 2003.

Callaham, J. L., Maeda, K., et Brunton, S. L. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10) : 103907, 2019.

Candes, E. J. et Tao, T. Near-optimal signal recovery from random projections : Universal encoding strategies ? *IEEE transactions on information theory*, 52 (12) :5406–5425, 2006.

Candes, E. J., Romberg, J. K., et Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics : A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8) : 1207–1223, 2006.

Candès, E. J., Li, X., Ma, Y., et Wright, J. Robust principal component analysis ? *Journal of the ACM (JACM)*, 58(3) :1–37, 2011.

Caraballo, E., Little, J., Debiasi, M., et Samimy, M. Development and implementation of an experimental-based reduced-order model for feedback control of subsonic cavity flows. 2007.

Carlberg, K. T., Jameson, A., Kochenderfer, M. J., Morton, J., Peng, L., et Witherden, F. D. Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning. *Journal of Computational Physics*, 395 :105–124, 2019.

Champion, K., Lusch, B., Kutz, J. N., et Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45) :22445–22451, 2019.

Chen, T. et Guestrin, C. Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Chowdhary, G. et Jategaonkar, R. Aerodynamic parameter estimation from flight data applying extended and unscented kalman filter. *Aerospace science and technology*, 14(2) :106–117, 2010.

Clarke, A. C. Profiles ofthe future : An inquiry into the limits of the possible, 1964.

Cohen, K., Siegel, S., et McLaughlin, T. Sensor placement based on proper orthogonal decomposition modeling of a cylinder wake. In *33rd AIAA Fluid Dynamics Conference and Exhibit*, page 4259, 2003.

Colburn, C., Cessna, J., et Bewley, T. State estimation in wall-bounded flow systems. part 3. the ensemble kalman filter. *Journal of Fluid Mechanics*, 682 : 289–303, 2011.

Cole, D. R. et Glauser, M. N. Applications of stochastic estimation in the axisymmetric sudden expansion. *Physics of Fluids*, 10(11) :2941–2949, 1998.

Costa, A. C., Ahamed, T., et Stephens, G. J. Adaptive, locally linear models of complex dynamics. *Proceedings of the National Academy of Sciences*, 116(5) : 1501–1510, 2019.

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., et Ho, S. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv :2006.11287*, 2020.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314, 1989.

de Silva, B. M., Callaham, J., Jonker, J., Goebel, N., Klemisch, J., McDonald, D., Hicks, N., Kutz, J. N., Brunton, S. L., et Aravkin, A. Y. Physics-informed machine learning for sensor fault detection with flight test data. *arXiv preprint arXiv :2006.13380*, 2020.

Deng, Z., Chen, Y., Liu, Y., et Kim, K. C. Time-resolved turbulent velocity field reconstruction using a long short-term memory (lstm)-based artificial intelligence framework. *Physics of Fluids*, 31(7) :075108, 2019.

Dubois, P., Gomez, T., Planckaert, L., et Perret, L. Data-driven predictions of the lorenz system. *Physica D : Nonlinear Phenomena*, 408 :132495, 2020.

Duraisamy, K., Iaccarino, G., et Xiao, H. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51 :357–377, 2019.

Durgesh, V. et Naughton, J. Multi-time-delay lse-pod complementary approach applied to unsteady high-reynolds-number near wake flow. *Experiments in fluids*, 49(3) :571–583, 2010.

Eckart, C. et Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3) :211–218, 1936.

Elgammal, A., Liu, B., Elhoseiny, M., et Mazzone, M. Can : Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv :1706.07068*, 2017.

Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., et Kutz, J. N. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238) :20200097, 2020.

ESDU. Strong winds in the atmospheric boundary layer, part 1 : Mean hourly wind speed - engineering sciences data unit, 1982.

Everson, R. et Sirovich, L. Karhunen–loeve procedure for gappy data. *JOSA A*, 12 (8) :1657–1664, 1995.

Eymard, R., Gallouët, T., et Herbin, R. Finite volume methods. *Handbook of numerical analysis*, 7 :713–1018, 2000.

Fan, D., Yang, L., Wang, Z., Triantafyllou, M. S., et Karniadakis, G. E. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42) :26091–26098, 2020.

Feynman, R. P., Leighton, R. B., et Sands, M. The feynman lectures on physics ; vol. ii. 1965.

Floors, R., Vincent, C. L., Gryning, S.-E., Peña, A., et Batchvarova, E. The wind profile in the coastal boundary layer : Wind lidar measurements and numerical modelling. *Boundary-layer meteorology*, 147(3) :469–491, 2013.

Fouatih, O. M., Medale, M., Imine, O., et Imine, B. Design optimization of the aerodynamic passive flow control on naca 4415 airfoil using vortex generators. *European Journal of Mechanics-B/Fluids*, 56 :82–96, 2016.

Freund, Y. et Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) : 119–139, 1997.

Fukami, K., Fukagata, K., et Taira, K. Assessment of supervised machine learning methods for fluid flows. *Theoretical and Computational Fluid Dynamics*, 34(4) : 497–519, 2020.

Fushiki, T. Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, 21(2) :137–146, 2011.

Galway, D., Etele, J., et Fusina, G. Development and implementation of an urban wind field database for aircraft flight simulation. *Journal of wind engineering and industrial aerodynamics*, 103 :73–85, 2012.

Garnier, P., Viquerat, J., Rabault, J., Larcher, A., Kuhnle, A., et Hachem, E. A review on deep reinforcement learning for fluid mechanics. *arXiv preprint arXiv :1908.04127*, 2019.

Gavish, M. et Donoho, D. L. The optimal hard threshold for singular values is 4/3. *IEEE Transactions on Information Theory*, 60(8) :5040–5053, 2014.

Gazzola, M., Van Rees, W. M., et Koumoutsakos, P. C-start : optimal start of larval fish. *Journal of Fluid Mechanics*, 698(9) :5–18, 2012.

George, W. K. Insight into the dynamics of coherent structures from a proper orthogonal decomposition dy. In *International seminar on wall turbulence*, 1988.

Glorot, X. et Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gonzalez, F. J. et Balajewicz, M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint arXiv :1808.01346*, 2018.

Gousseau, P., Blocken, B., et Van Heijst, G. Quality assessment of large-eddy simulation of wind flow around a high-rise building : Validation and solution verification. *Computers & Fluids*, 79 :120–133, 2013.

Graziano, M. D., D'Errico, M., et Rufino, G. Ship heading and velocity analysis by wake detection in sar images. *Acta astronautica*, 128 :72–82, 2016.

Gronskis, A., Heitz, D., et Mémin, E. Inflow and initial conditions for direct numerical simulation based on adjoint data assimilation. *Journal of Computational Physics*, 242 :480–497, 2013.

Guezennec, Y. Stochastic estimation of coherent structures in turbulent boundary layers. *Physics of Fluids A : Fluid Dynamics*, 1(6) :1054–1060, 1989.

Hadjighasem, A., Karrasch, D., Teramoto, H., et Haller, G. Spectral-clustering approach to lagrangian vortex detection. *Physical Review E*, 93(6) :063107, 2016.

Haller, G. An objective definition of a vortex. *Journal of fluid mechanics*, 525 :1, 2005.

Hey, A. J., Tansley, S., Tolle, K. M., *et al. The fourth paradigm : data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02) :107–116, 1998.

Hochreiter, S. et Schmidhuber, J. Long short-term memory. *Neural computation*, 9 (8) :1735–1780, 1997.

Hoerl, A. E. et Kennard, R. W. Ridge regression : Biased estimation for nonorthogonal problems. *Technometrics*, 12(1) :55–67, 1970.

Hornik, K., Stinchcombe, M., et White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

Hosain, M. L. et Fdhila, R. B. Literature review of accelerated cfd simulation methods towards online application. *Energy Procedia*, 75 :3307–3314, 2015.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6) :417, 1933.

Hui, M., Larsen, A., et Xiang, H. Wind turbulence characteristics study at the stonecutters bridge site : Part i—mean wind and turbulence intensities. *Journal of Wind Engineering and Industrial Aerodynamics*, 97(1) :22–36, 2009.

Hwangbo, J., Sa, I., Siegwart, R., et Hutter, M. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4) :2096–2103, 2017.

Igel, C., Hansen, N., et Roth, S. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1) :1–28, 2007.

Issa, R. I. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of computational physics*, 62(1) :40–65, 1986.

Jamieson, K. et Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248. PMLR, 2016.

Jasak, H. Error analysis and estimation for the finite volume method with applications to fluid flows. 1996.

Jayaraman, B., Al Mamun, S., et Lu, C. Interplay of sensor quantity, placement and system dimension in pod-based sparse reconstruction of fluid flows. *Fluids*, 4(2) :109, 2019.

Kaandorp, M. L. et Dwight, R. P. Data-driven modelling of the reynolds stress tensor using random forests with invariance. *Computers & Fluids*, 202 :104497, 2020.

Kaimal, J. C. et Finnigan, J. J. *Atmospheric boundary layer flows : their structure and measurement*. Oxford university press, 1994.

Kaiser, E., Noack, B. R., Cordier, L., Spohn, A., Segond, M., Abel, M., Daviller, G., Östh, J., Krajnović, S., et Niven, R. K. Cluster-based reduced-order modelling of a mixing layer. *Journal of Fluid Mechanics*, 754 :365–414, 2014.

Kaiser, E., Kutz, J. N., et Brunton, S. L. Data-driven discovery of koopman eigenfunctions for control. *arXiv preprint arXiv :1707.01146*, 2017.

Kaiser, E., Morzyński, M., Daviller, G., Kutz, J. N., Brunton, B. W., et Brunton, S. L. Sparsity enabled cluster reduced-order models for control. *Journal of Computational Physics*, 352 :388–409, 2018.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1) :35–45, 1960.

Kalnay, E. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

Kastner-Klein, P. et Rotach, M. W. Mean flow and turbulence characteristics in an urban roughness sublayer. *Boundary-Layer Meteorology*, 111(1) :55–84, 2004.

Kato, H., Yoshizawa, A., Ueno, G., et Obayashi, S. A data assimilation methodology for reconstructing turbulent flows around aircraft. *Journal of Computational Physics*, 283 :559–581, 2015.

Kelly, R. On the stability of an inviscid shear layer which is periodic in space and time. *Journal of Fluid Mechanics*, 27(4) :657–689, 1967.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., et Solenthaler, B. Deep fluids : A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

Kingma, D. P. et Ba, J. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

Kingma, D. P., Salimans, T., et Welling, M. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv :1506.02557*, 2015.

Klus, S., Nüske, F., Peitz, S., Niemann, J.-H., Clementi, C., et Schütte, C. Data-driven approximation of the koopman generator : Model reduction, system identification, and control. *Physica D : Nonlinear Phenomena*, 406 :132416, 2020.

Ko, J., Lucor, D., et Sagaut, P. Sensitivity of two-dimensional spatially developing mixing layers with respect to uncertain inflow conditions. *Physics of Fluids*, 20 (7) :077102, 2008.

Kodinariya, T. M. et Makwana, P. R. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6) :90–95, 2013.

Kolmogorov, A. N. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Cr Acad. Sci. URSS*, 30 :301–305, 1941.

Koopman, B. O. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5) :315, 1931.

Kuhn, H. et Tucker, A. Nonlinear programming. proceedings of the 2nd berkeley symposium on mathematical statistics and probability, university of california. 1951.

Kumar, B. et Mittal, S. Prediction of the critical reynolds number for flow past a circular cylinder. *Computer methods in applied mechanics and engineering*, 195 (44-47) :6046–6058, 2006.

Kutz, J. N. *Data-driven modeling & scientific computation : methods for complex systems & big data*. Oxford University Press, 2013.

Kutz, J. N., Brunton, S. L., Brunton, B. W., et Proctor, J. L. *Dynamic mode decomposition : data-driven modeling of complex systems*. SIAM, 2016a.

Kutz, J. N., Fu, X., et Brunton, S. L. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2) :713–735, 2016b.

Laizet, S., Lardeau, S., et Lamballais, E. Direct numerical simulation of a mixing layer downstream a thick splitter plate. *Physics of Fluids*, 22(1) :015104, 2010.

Lang, A. W., Bradshaw, M. T., Smith, J. A., Wheelus, J. N., Motta, P. J., Habegger, M. L., et Hueter, R. E. Movable shark scales act as a passive dynamic micro-roughness to control flow separation. *Bioinspiration & biomimetics*, 9(3) :036017, 2014.

Le Clainche, S. et Vega, J. M. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2) :882–925, 2017.

Le Dimet, F.-X. et Talagrand, O. Variational algorithms for analysis and assimilation of meteorological observations : theoretical aspects. *Tellus A : Dynamic Meteorology and Oceanography*, 38(2) :97–110, 1986.

LeCun, Y., Bengio, Y., et Hinton, G. Deep learning. *nature*, 521(7553) :436–444, 2015.

Lee, J. H., Seena, A., Lee, S.-h., et Sung, H. J. Turbulent boundary layers over rod-and cube-roughened walls. *Journal of Turbulence*, 13(1) :N40, 2012.

Lee, K. et Carlberg, K. Deep conservation : A latent-dynamics model for exact satisfaction of physical conservation laws. *arXiv preprint arXiv :1909.09754*, 2019.

Lee, S., Yang, J., Forooghi, P., Stroh, A., et Bagheri, S. Predicting drag on rough surfaces by transfer learning of empirical correlations. *arXiv preprint arXiv :2106.05995*, 2021.

Lemarié-Rieusset, P. G. *The Navier-Stokes problem in the 21st century*. CRC Press, 2018.

Li, F., Ren, G., et Lee, J. Multi-step wind speed prediction based on turbulence intensity and hybrid deep neural networks. *Energy Conversion and Management*, 186 :306–322, 2019.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., et Talwalkar, A. Massively parallel hyperparameter tuning. 2018a.

Li, Q., Dietrich, F., Bollt, E. M., et Kevrekidis, I. G. Extended dynamic mode decomposition with dictionary learning : A data-driven adaptive spectral decomposition of the koopman operator. *Chaos : An Interdisciplinary Journal of Nonlinear Science*, 27(10) :103111, 2017.

Li, R., Wang, X., Lei, L., et Song, Y. *l_{21}*-norm based loss function and regularization extreme learning machine. *IEEE Access*, 7 :6575–6586, 2018b.

Liao, J. C., Beal, D. N., Lauder, G. V., et Triantafyllou, M. S. Fish exploiting vortices decrease muscle activity. *Science*, 302(5650) :1566–1569, 2003.

Lighthill, J. Artificial intelligence : A general survey. In *Artificial Intelligence : a paper symposium*, pages 1–21. Science Research Council London, 1973.

Likas, A., Vlassis, N., et Verbeek, J. J. The global k-means clustering algorithm. *Pattern recognition*, 36(2) :451–461, 2003.

Ling, J. et Templeton, J. Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty. *Physics of Fluids*, 27(8) :085103, 2015.

Ling, J., Kurzawski, A., et Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807 :155–166, 2016.

Lletí, R., Ortiz, M. C., Sarabia, L. A., et Sánchez, M. S. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes. *Analytica Chimica Acta*, 515(1) :87–100, 2004.

Loiseau, J.-C. Data-driven modeling of the chaotic thermal convection in an annular thermosyphon. *Theoretical and Computational Fluid Dynamics*, 34(4) :339–365, 2020.

Loiseau, J.-C. et Brunton, S. L. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838 :42–67, 2018a.

Loiseau, J.-C. et Brunton, S. L. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838 :42–67, 2018b.

Loiseau, J.-C., Noack, B. R., et Brunton, S. L. Sparse reduced-order modelling : sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844 : 459–490, 2018.

Lui, H. F. et Wolf, W. R. Construction of reduced order models for fluid flows using deep feedforward neural networks. *arXiv preprint arXiv :1903.05206*, 2019.

Lumley, J. L. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, 1967.

Luo, S., Chew, Y., et Ng, Y. Characteristics of square cylinder wake transition flows. *Physics of Fluids*, 15(9) :2549–2559, 2003.

Lusch, B., Kutz, J. N., et Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1) :1–10, 2018.

Lusseyran, F., Guéniat, F., Basley, J., Douay, C., Pastur, L., Faure, T., et Schmid, P. Flow coherent structures and frequency signature : application of the dynamic modes decomposition to open cavity flow. In *Journal of Physics : Conference Series*, volume 318, page 042036. IOP Publishing, 2011.

Ma, P., Tian, Y., Pan, Z., Ren, B., et Manocha, D. Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37 (4) :1–11, 2018.

Manohar, K., Brunton, B. W., Kutz, J. N., et Brunton, S. L. Data-driven sparse sensor placement for reconstruction : Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, 38(3) :63–86, 2018a.

Manohar, K., Kutz, J. N., et Brunton, S. L. Optimal sensor and actuator selection using balanced model reduction. *arXiv preprint arXiv :1812.01574*, 2018b.

Mansour, N., Hussain, F., et Buelll, J. Subharmonic resonance in a mixing layer. *Studying Turbulence Using Numerical Simulation Databases-I1*, page 57, 1988.

Martinuzzi, R. et Tropea, C. The flow around surface-mounted, prismatic obstacles placed in a fully developed channel flow (data bank contribution). 1993.

Mathey, F., Cokljat, D., Bertoglio, J. P., et Sergent, E. Assessment of the vortex method for large eddy simulation inlet conditions. *Progress in Computational Fluid Dynamics, An International Journal*, 6(1-3) :58–67, 2006.

Matsuo, M., Nakamura, T., Morimoto, M., Fukami, K., et Fukagata, K. Supervised convolutional network for three-dimensional fluid data reconstruction from sectional flow fields with adaptive super-resolution assistance. *arXiv preprint arXiv :2103.09020*, 2021.

Mattos, B., Macedo, A., et Silva Filho, D. Considerations about winglet design. In *21st AIAA Applied Aerodynamics Conference*, page 3502, 2003.

Maulik, R., San, O., Rasheed, A., et Vedula, P. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858 : 122–144, 2019.

McClean, J. et Sumner, D. An experimental investigation of aspect ratio and incidence angle effects for the flow around surface-mounted finite-height square prisms. *Journal of Fluids Engineering*, 136(8), 2014.

Mendible, A., Brunton, S. L., Aravkin, A. Y., Lowrie, W., et Kutz, J. N. Dimensionality reduction and reduced-order modeling for traveling wave physics. *Theoretical and Computational Fluid Dynamics*, 34(4) :385–400, 2020.

Menter, F. R. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, 32(8) :1598–1605, 1994.

Menter, F. R., Kuntz, M., et Langtry, R. Ten years of industrial experience with the sst turbulence model. *Turbulence, heat and mass transfer*, 4(1) :625–632, 2003.

Mercer, J. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209 (441-458) :415–446, 1909.

Mezić, I. et Banaszuk, A. Comparison of systems with complex behavior. *Physica D : Nonlinear Phenomena*, 197(1-2) :101–133, 2004.

Michalke, A. On the inviscid instability of the hyperbolictangent velocity profile. *Journal of Fluid Mechanics*, 19(4) :543–556, 1964.

Milano, M. et Koumoutsakos, P. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1) :1–26, 2002.

Minsky, M. et Papert, S. A. *Perceptrons : An introduction to computational geometry*. MIT press, 1969.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., *et al.* Human-level control through deep reinforcement learning. *nature*, 518(7540) :529–533, 2015.

Mohan, A. T. et Gaitonde, D. V. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *arXiv preprint arXiv :1804.09269*, 2018.

Mohan, P., Fitzsimmons, N., et Moser, R. D. Scaling of lyapunov exponents in homogeneous isotropic turbulence. *Physical Review Fluids*, 2(11) :114606, 2017.

Mons, V., Chassaing, J.-C., Gomez, T., et Sagaut, P. Reconstruction of unsteady viscous flows using data assimilation schemes. *Journal of Computational Physics*, 316 :255–280, 2016.

Morton, J., Kochenderfer, M. J., et Witherden, F. D. Parameter-conditioned sequential generative modeling of fluid flows. *AIAA Journal*, pages 1–17, 2021.

Murray, N. E. et Ukeiley, L. S. Estimation of the flowfield from surface pressure measurements in an open cavity. *AIAA journal*, 41(5) :969–972, 2003.

Murray, N. E. et Ukeiley, L. S. An application of gappy pod. *Experiments in Fluids*, 42(1) :79–91, 2007.

Ncrgaard, M., Jorgensen, C. C., et Ross, J. C. Neural network prediction of new aircraft design coefficients. 1997.

Noack, B. R. et Eckelmann, H. A global stability analysis of the steady and periodic cylinder wake. *Journal of Fluid Mechanics*, 270 :297–330, 1994.

Noack, B. R., Afanasiev, K., Morzynski, M., Tadmor, G., et Thiele, F. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497 :335, 2003.

Novati, G., de Laroussilhe, H. L., et Koumoutsakos, P. Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, pages 1–10, 2021.

Odaibo, S. Tutorial : Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv :1907.08956*, 2019.

Ott, C., Gallas, Q., Delva, J., Lippert, M., et Keirsbulck, L. High frequency characterization of a sweeping jet actuator. *Sensors and Actuators A : Physical*, 291 :39–47, 2019.

Ott, C., Pivot, C., Dubois, P., Gallas, Q., Delva, J., Lippert, M., et Keirsbulck, L. Pulsed jet phase-averaged flow field estimation based on neural network approach. *Experiments in Fluids*, 62(4) :1–16, 2021.

Paris, R., Beneddine, S., et Dandois, J. Robust flow control and optimal sensor placement using deep reinforcement learning. *Journal of Fluid Mechanics*, 913, 2021.

Perlman, E., Burns, R., Li, Y., et Meneveau, C. Data exploration of turbulence simulations using a database cluster. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–11, 2007.

Perozzi, G., Efimov, D., Biannic, J.-M., Planckaert, L., et Coton, P. Wind estimation algorithm for quadrotors using detailed aerodynamic coefficients. In *2018 Annual American Control Conference (ACC)*, pages 1921–1926. IEEE, 2018.

Pfeiffer, J. Closed-loop active flow control for road vehicles under unsteady cross-wind conditions. 2016.

Plaut, E. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv :1804.10253*, 2018.

Podvin, B., Fraigneau, Y., Lusseyran, F., et Gougat, P. A reconstruction method for the flow past an open cavity. 2006.

Pollard, A., Hacker, T. J., et Dyke, S. Whither turbulence and big data for the twenty-first century. In *Whither Turbulence and Big Data in the 21st Century ?*, pages 551–574. Springer, 2017.

Pope, S. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2) :331–340, 1975.

Pope, S. B. Turbulent flows, 2001.

Proctor, J. L., Brunton, S. L., et Kutz, J. N. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1) :142–161, 2016.

Qiu, X., Meyerson, E., et Miikkulainen, R. Quantifying point-prediction uncertainty in neural networks via residual estimation with an i/o kernel. *arXiv preprint arXiv :1906.00588*, 2019.

Rabault, J., Kuchta, M., Jensen, A., Réglade, U., et Cerardi, N. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of fluid mechanics*, 865 :281–302, 2019.

Raissi, M., Perdikaris, P., et Karniadakis, G. E. Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378 : 686–707, 2019.

Rajaee, M., Karlsson, S. K., et Sirovich, L. Low-dimensional description of free-shear-flow coherent structures and their dynamical behaviour. *Journal of Fluid Mechanics*, 258 :1–29, 1994.

Rechenberg, I. *Kybernetische lösungsansteuerung einer experimentellen forschungsaufgabe*. 1964.

Reddy, G., Wong-Ng, J., Celani, A., Sejnowski, T. J., et Vergassola, M. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726) :236–239, 2018.

Richardson, L. F. *Weather prediction by numerical process*. Cambridge university press, 1922.

Rosenblatt, F. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958.

Rowley, C. W. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03) :997–1013, 2005.

Rowley, C. W. et Dawson, S. T. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49 :387–417, 2017.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv :1609.04747*, 2016.

Rudy, S. H., Brunton, S. L., Proctor, J. L., et Kutz, J. N. Data-driven discovery of partial differential equations. *Science Advances*, 3(4) :e1602614, 2017.

Rumelhart, D. E., Hinton, G. E., et Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088) :533–536, 1986.

Sagaut, P. et Deck, S. Large eddy simulation for aerodynamics : status and perspectives. *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 367(1899) :2849–2860, 2009.

Sakamoto, H. et Arie, M. Vortex shedding from a rectangular prism and a circular cylinder placed vertically in a turbulent boundary layer. *Journal of Fluid Mechanics*, 126 :147–165, 1983.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., et Valaee, S. Recent advances in recurrent neural networks. *arXiv preprint arXiv :1801.01078*, 2017.

Sankaran, S., Moghadam, M. E., Kahn, A. M., Tseng, E. E., Guccione, J. M., et Marsden, A. L. Patient-specific multiscale modeling of blood flow for coronary artery bypass graft surgery. *Annals of biomedical engineering*, 40(10) :2228–2242, 2012.

Scherl, I., Strom, B., Shang, J. K., Williams, O., Polagye, B. L., et Brunton, S. L. Robust principal component analysis for modal decomposition of corrupt fluid flows. *Physical Review Fluids*, 5(5) :054401, 2020.

Schmid, P. J. Application of the dynamic mode decomposition to experimental data. *Experiments in fluids*, 50 :1123–1130, 2011.

Semeraro, O., Bellani, G., et Lundell, F. Analysis of time-resolved piv measurements of a confined turbulent jet using pod and koopman modes. *Experiments in fluids*, 53(5) :1203–1220, 2012.

Sheng, R., Perret, L., Calmet, I., Demouge, F., et Guilhot, J. Wind tunnel study of wind effects on a high-rise building at a scale of 1 : 300. *Journal of Wind Engineering and Industrial Aerodynamics*, 174 :391–403, 2018.

Sheng, R. *Application de l'approche de simulation des grandes échelles à l'évaluation des charges de vent sur les structures.* PhD thesis, École centrale de Nantes, 2017.

Shlens, J. A tutorial on principal component analysis. *arXiv preprint arXiv :1404.1100*, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., *et al.* Mastering the game of go without human knowledge. *nature*, 550(7676) :354–359, 2017.

Singh, A. P., Medida, S., et Duraisamy, K. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA journal*, 55(7) :2215–2227, 2017.

Sirovich, L. Turbulence and the dynamics of coherent structures. parts i-iii. *Quarterly of applied mathematics*, 45(3) :561–590, 1987.

Sirovich, L. et Kirby, M. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3) :519–524, 1987.

Smagorinsky, J. General circulation experiments with the primitive equations : I. the basic experiment. *Monthly weather review*, 91(3) :99–164, 1963.

Smola, A. J. et Schölkopf, B. A tutorial on support vector regression. *Statistics and computing*, 14(3) :199–222, 2004.

Snyder, W. H. et Castro, I. P. The critical reynolds number for rough-wall boundary layers. *Journal of wind engineering and industrial aerodynamics*, 90(1) :41–54, 2002.

Sohankar, A. Flow over a bluff body from moderate to high reynolds numbers using large eddy simulation. *Computers & Fluids*, 35(10) :1154–1168, 2006.

Sousa, J., García-Sánchez, C., et Gorlé, C. Improving urban flow predictions through data assimilation. *Building and Environment*, 132 :282–290, 2018.

Spielman, D. A., Wang, H., et Wright, J. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, pages 37–1. JMLR Workshop and Conference Proceedings, 2012.

Sreenivasan, K., Strykowski, P., et Olinger, D. Hopf bifurcation, landau equation, and vortex shedding behind circular cylinders. In *Forum on unsteady flow separation*, volume 1, pages 1–13, 1987.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., et Salakhutdinov, R. Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1) :1929–1958, 2014.

Sun, L. et Wang, J.-X. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3) :161–169, 2020.

Suzuki, T. Reduced-order kalman-filtered hybrid simulation combining particle tracking velocimetry and direct numerical simulation. *Journal of Fluid Mechanics*, 709 :249–288, 2012.

Symon, S., Dovetta, N., McKeon, B. J., Sipp, D., et Schmid, P. J. Data assimilation of mean velocity from 2d piv measurements of flow over an idealized airfoil. *Experiments in fluids*, 58(5) :61, 2017.

Symon, S., Sipp, D., Schmid, P. J., et McKeon, B. J. Mean and unsteady flow reconstruction using data-assimilation and resolvent analysis. *AIAA Journal*, 58 (2) :575–588, 2020.

Taira, K., Brunton, S. L., Dawson, S. T., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V., et Ukeiley, L. S. Modal analysis of fluid flows : An overview. *Aiaa Journal*, 55(12) :4013–4041, 2017.

Takens, F. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

Tauro, F., Grimaldi, S., et Porfiri, M. Unraveling flow patterns through nonlinear manifold learning. *PloS one*, 9(3) :e91131, 2014.

Tedrake, R., Jackowski, Z., Cory, R., Roberts, J. W., et Hoburg, W. Learning to fly like a bird. In *14th International symposium on robotics research. Lucerne, Switzerland*. Citeseer, 2009.

Teo, C., Lim, K., Hong, G., et Yeo, M. A neural net approach in analyzing photograph in piv. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1535–1538. IEEE, 1991.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1) :267–288, 1996.

Toda, H. Y. et Phillips, P. C. Vector autoregression and causality : a theoretical overview and simulation study. *Econometric reviews*, 13(2) :259–285, 1994.

Towne, A., Schmidt, O. T., et Colonius, T. Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, 847 :821–867, 2018.

Tu, J., Yeoh, G. H., et Liu, C. *Computational fluid dynamics : a practical approach*. Butterworth-Heinemann, 2018.

Tu, J., Rowley, C., Aram, E., et Mittal, R. Koopman spectral analysis of separated flow over a finite-thickness flat plate with elliptical leading edge. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 38, 2011.

Tu, J. H., Griffin, J., Hart, A., Rowley, C. W., Cattafesta, L. N., et Ukeiley, L. S. Integration of non-time-resolved piv and time-resolved velocity point sensors for dynamic estimation of velocity fields. *Experiments in fluids*, 54(2) :1429, 2013.

Vapnik, V. N. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5) :988–999, 1999.

Varoquaux, G., Raamana, P. R., Engemann, D. A., Hoyos-Idrobo, A., Schwartz, Y., et Thirion, B. Assessing and tuning brain decoders : cross-validation, caveats, and guidelines. *NeuroImage*, 145 :166–179, 2017.

Vassilicos, J. C. Dissipation in turbulent flows. *Annual Review of Fluid Mechanics*, 47 :95–114, 2015.

Venturi, D. et Karniadakis, G. E. Gappy data and reconstruction procedures for flow past a cylinder. *Journal of Fluid Mechanics*, 519 :315, 2004.

Verma, S., Novati, G., et Koumoutsakos, P. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23) :5849–5854, 2018.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., et Koumoutsakos, P. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 474(2213) :20170844, 2018.

Vuillemin, P., Poussot-Vassal, C., et Alazard, D. H2 optimal and frequency limited approximation methods for large-scale lti dynamical systems. *IFAC Proceedings Volumes*, 46(2) :719–724, 2013.

Wang, H. et Zhou, Y. The finite-length square cylinder near wake. *Journal of Fluid Mechanics*, 638 :453, 2009.

Wang, J.-X., Wu, J.-L., et Xiao, H. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3) :034603, 2017.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., et Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

Wiener, N. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 1948.

Wikle, C. K. et Berliner, L. M. A bayesian tutorial for data assimilation. *Physica D : Nonlinear Phenomena*, 230(1-2) :1–16, 2007.

Willcox, K. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & fluids*, 35(2) :208–226, 2006.

Williams, M. O., Kevrekidis, I. G., et Rowley, C. W. A data–driven approximation of the koopman operator : Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6) :1307–1346, 2015.

Winant, C. D. et Browand, F. K. Vortex pairing : the mechanism of turbulent mixing-layer growth at moderate reynolds number. *Journal of Fluid Mechanics*, 63(2) :237–255, 1974.

Xiao, D., Fang, F., Heaney, C. E., Navon, I., et Pain, C. A domain decomposition method for the non-intrusive reduced order modelling of fluid flow. *Computer Methods in Applied Mechanics and Engineering*, 354 :307–330, 2019.

Xie, B. *Improved vortex method for LES inflow generation and applications to channel and flat-plate flows*. PhD thesis, Lyon, 2016.

Xu, J. et Duraisamy, K. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372 :113379, 2020.

Xu, J., Du, T., Foshey, M., Li, B., Zhu, B., Schulz, A., et Matusik, W. Learning to fly : computational controller design for hybrid uavs with reinforcement learning. *ACM Transactions on Graphics (TOG)*, 38(4) :1–12, 2019.

Yildirim, B., Chryssostomidis, C., et Karniadakis, G. Efficient sensor placement for ocean measurements using low-dimensional concepts. *Ocean Modelling*, 27 (3-4) :160–173, 2009.

Yoder, D. A., DeBonis, J. R., et Georgiadis, N. J. Modeling of turbulent free shear flows. *Computers & fluids*, 117 :212–232, 2015.

Zebib, A. Stability of viscous flow past a circular cylinder. *Journal of Engineering Mathematics*, 21(2) :155–165, 1987.

Zhang, H.-Q., Fey, U., Noack, B. R., König, M., et Eckelmann, H. On the transition of the cylinder wake. *Physics of Fluids*, 7(4) :779–794, 1995.

Zheng, P., Askham, T., Brunton, S. L., Kutz, J. N., et Aravkin, A. Y. A unified framework for sparse relaxed regularized regression : Sr3. *IEEE Access*, 7 : 1404–1423, 2018.

Zhiyin, Y. Large-eddy simulation : Past, present and the future. *Chinese Journal of Aeronautics*, 28(1) :11–24, 2015. ISSN 1000-9361.

Résumé

Utilisation de l'apprentissage automatique en mécanique des fluides pour la réduction, la reconstruction et la prédiction orientée données du champ de vitesse fluctuante d'un écoulement

La mécanique des fluides est présente dans de nombreuses thématiques industrielles telles que la santé, le transport et l'énergie. Pour modéliser, contrôler et réduire les écoulements d'intérêt, il faut résoudre des problèmes d'optimisation non linéaires et multi-échelles. Face aux difficultés de résolution analytique et dans l'âge du *Big Data*, on cherche à utiliser la disponibilité des données numériques et expérimentales pour faire l'optimisation à partir des données. Dans ce contexte, la thèse s'intéresse aux outils de l'apprentissage automatique pour l'estimation orientée données d'un écoulement. En particulier, on cherche à réduire, reconstruire et prédire le champ de vitesse d'écoulements de complexité croissante : le sillage laminaire d'un cylindre, une couche de mélange spatiale, le sillage turbulent d'un cylindre rectangulaire et l'écoulement autour d'une tour isolée. Pour ce faire, on commence par reformuler les problèmes de réduction, de reconstruction et de prédiction. Pour la réduction, on s'intéresse à l'utilisation d'auto-encodeurs pour déterminer un espace latent de dimension réduite dans lequel on peut réécrire le film de l'écoulement. Concernant la reconstruction, on utilise l'apprentissage supervisé pour estimer l'état latent à un instant donné à partir de mesures du champ de vitesse fluctuante au même instant. Pour la prédiction, on cherche une approximation finie de l'opérateur de Koopman pour avancer linéairement dans le temps des observations choisies de l'état latent. Les résultats mettent en évidence que l'apprentissage automatique est une piste pertinente pour établir des modèles d'estimation d'écoulements. Toutefois, les développements restent académiques et le déploiement de modèles sur des configurations très turbulentes reste peu envisageable, notamment pour des questions de robustesses (à de nouvelles configurations, au bruit, etc.)
mots clés : *apprentissage automatique, réduction, reconstruction, prédiction, auto-encodeurs, opérateur de Koopman*

Abstract

Use of machine learning tools in fluid mechanics for the data-driven reduction, reconstruction and prediction of a fluid flow fluctuating velocity field

Fluid mechanics is an important part in industrial questions. The modelisation, reduction and control of fluid flows require to solve a nonlinear and multiscale optimisation problem. In the age of artificial intelligence, there is a craze to solve these optimisation problems using the wealth of experimental and numerical data. In this context, the objective of the manuscript is to present how machine learning tools can be used for the data-driven estimation of fluid flow velocity fields. In particular, we aim at reducing, reconstructing and predicting four increasing complexity flows : the laminar wake of a cylinder, a spatial mixing layer, the turbulent wake of a square cylinder and the flow around an isolated tower. To do so, we start by reformulating the reduction, the reconstruction and the prediction problems. For the reduction, we use autoencoders to find a latent space and rewrite the flow with with a low rank approximation. For the reconstruction, we use supervised learning tools to estimate the latent state from limited measurements of the fluctuating velocity field. For the prediction, we search for a finite approximation of the Koopman operator to linearly advance in time observations of the latent state. The results show that machine learning is a promising direction to establish fluid flow estimation models. However, the deployment of models for highly turbulent flows remains questionable mainly because of robustness issues.
keywords : *machine learning, reduction, reconstruction, prediction, Koopman operator, autoencoders*