Stage de Master 1 Calcul Scientifique et Modélisation de l'Université de Rennes

Machine Learning pour la reconstruction d'écoulements turbulents

Laboratoire d'accueil :	Institut de Recherche en Génie Civil et Mécanique, École Centrale Nantes
Encadrant :	Lucas LESTANDI (GEM)
Tuteur universitaire :	Eric DARRIGRAND (Université de Rennes)
Étudiant(e) :	Ilona RICHARD
Période :	Du 19 mai 2025 au 29 août 2025
Lieu :	Nantes, France

Université de Rennes
UFR Mathématiques

Table des matières

Chapitre 1

Présentation du laboratoire et du problème

1.1 Cadre du stage

Mon stage s'est déroulé au sein du laboratoire GEM (Institut de Recherche en Génie Civil et Mécanique) situé dans l'École Centrale Nantes (ECN). Il s'agit d'un laboratoire avec des équipes articulées autour de 3 sous-domaines de la physique qui sont le génie civil, mécanique des matériaux et procédés, et modélisation et simulation mécanique. Il est rattaché à l'École Centrale Nantes, Nantes Université et le CNRS et est implanté à Nantes et Saint-Nazaire. Les collaborations avec d'autres établissements et entreprises sont nombreuses.

Le stage s'inscrit dans une collaboration internationale nommée "Réduction de données et modélisation surrogate de la transition vers la turbulence des données d'instabilité de Rayleigh-Taylor obtenues par DNS" entre l'ECN et l'Indian Institute of Technology (Indian School of Mines) de Dhanbad, dirigée par Lucas Lestandi côté ECN et Aditi Sengupta pour l'IIT-ISM Dhanbad, et financée par le Centre Franco-Indien pour la Promotion de la Recherche Avancée.

Le laboratoire indien dispose de beaucoup de données de simulation obtenues par simulation directe (DNS) et c'est au laboratoire français de les analyser désormais. J'ai travaillé sous la supervision de Lucas Lestandi, enseignant-chercheur à l'ECN et membre du GEM, et en collaboration avec Yassin Ajanif, doctorant de 2^{me} année, qui a travaillé sur un modèle de réduction des données (ROM) utilisant la Décomposition Orthogonale aux valeurs Propres (POD).

1.2 Présentation du stage

Le but du stage est d'appliquer une ou plusieurs méthodes de Machine Learning afin de résoudre un problème de modélisation d'écoulements turbulents. La finalité est d'appliquer les méthodes trouvées aux données collectées par l'IIT-ISM.
L'une des problématiques en mécanique des fluides est le besoin de beaucoup de points dans le maillage et donc de grandes données au départ et à la fin, ce qui augmente la complexité des algorithmes de résolution des équations différentielles. La conséquence est un

temps de plusieurs jours pour l'exécution du code avec supercalculateur pour la méthode de résolution directe des équations de Navier-Stokes (DNS).
Depuis quelques années, les méthodes de Machine Learning sont de plus en plus développées pour le traitement de grandes données et ce sont donc des méthodes de plus en plus utilisées en CFD (mécanique des fluides numérique), bien que ce ne soit pas l'une des principales applications.

Le stage s'est déroulé en deux parties avec tout d'abord une partie exploration des méthodes de Machine Learning utilisées dans la mécanique des fluides et ensuite une partie compréhension et utilisation d'un article sur une méthode appliquée à la modélisation d'un écoulement turbulent.

Chapitre 2

Exploration du ML pour la CFD

L'IA est le sujet du moment et est de plus en plus utilisée notamment dans le traitement de grosses données, le traitement du langage naturel (NLP) et les IA génératives. Depuis quelques années, certains chercheurs de la sphère de mécanique des fluides numérique s'intéressent donc tout naturellement à l'utilisation de l'IA pour faire face aux grandes données et diminuer le temps de calcul.

2.1 Turbulence

Il existe 2 régimes pour qualifier un écoulement, périodique ou turbulent, qui dépendent du nombre de Reynolds.

$$Re = \frac{u \times d}{\nu}$$

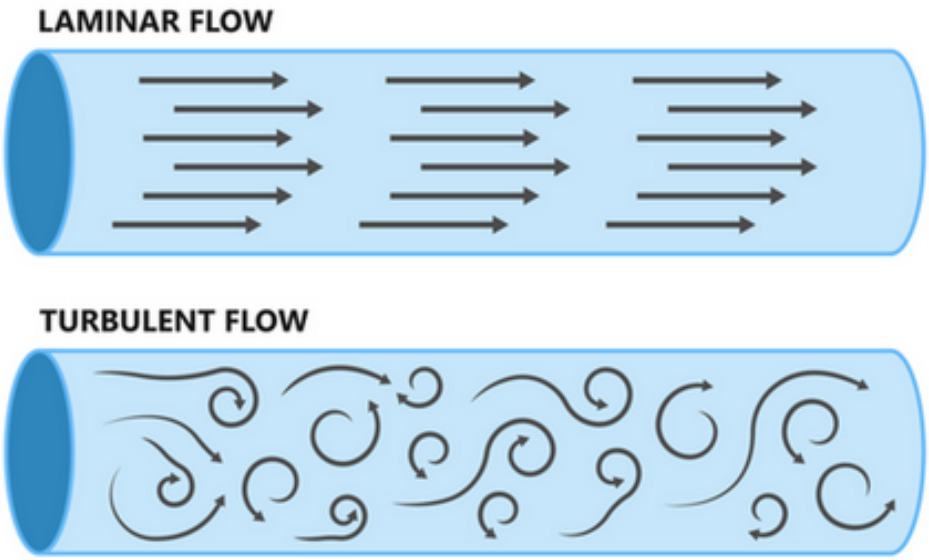
avec v=vitesse, d=diamètre et ν=viscosité cinématique (mesure l'étalement du fluide).
Lorsque Re est petit, l'écoulement est périodique et donc prévisible à partir des données de la première période.
Lorsque Re est grand, l'écoulement est dit turbulent, il devient chaotique et difficilement prédictible. Un maillage fin et beaucoup de données sont nécessaires pour résoudre numériquement les équations de Navier-Stokes.
C'est donc pour résoudre ce problème que certains scientifiques se sont intéressés au Machine Learning pour la CFD dont les écoulements turbulents.

Pour modéliser les écoulements turbulents, il existe 3 principales méthodes d'approximation afin d'éviter d'utiliser la DNS.

- LES (Large Eddy Simulation) : simulation à grandes échelles
- RANS (Reynolds Averaged Navier–Stokes) : simulation avec la moyenne
- hybride : RANS sur les parois et LES sur le reste du domaine

2.2 Méthodes utilisées pour la CFD

Dans un premier temps, j'ai donc fait une recherche bibliographique pour connaître et comprendre les méthodes d'IA dans la mécanique des fluides, et je me suis ensuite concentrée sur la modélisation d'écoulements turbulents.
Je suis partie de 3 articles que mon responsable de stage m'a recommandés, puis j'ai cherché dans la littérature scientifique à partir des co-auteurs des trois premiers articles



Ecoulement laminaire et turbulent

et avec les noms des méthodes qui revenaient souvent.
Il y a par exemple des méthodes de ML qui résolvent directement le système d'équations de Navier-Stokes mais étant donné que nous avions déjà les données par simulation directe, je me suis redirigée vers les articles d'analyse de données avec réduction de modèle et les méthodes de super-reproduction.

2.3 Recherche d'un article avec code

Dans un deuxième temps, j'ai cherché un article avec code pour modéliser un écoulement turbulent.
Il s'agit d'un article écrit par Alberto Solera-Rico, Carlos Sanmiguel Vila, Miguel Gómez-López, Yuning Wang, Abdulrahman Almashjary, Scott T. M. Dawson et Ricardo Vinuesa, publié en janvier 2024 et qui utilise 2 méthodes afin de construire un modèle réduit (ROM) de l'écoulement.
Le premier exemple est un écoulement périodique passant entre 2 plaques avec inclinaison à 90° et Re=40 (calculé avec la longueur de la plaque c).
Le deuxième exemple est un écoulement avec une inclinaison à 80° et Re=100, il s'agit d'un régime transitoire entre périodique et turbulent.

Les données lues et enregistrées par le code sont au format hdf5 qui n'est pas ouvrable donc j'ai créé un script python qui permet d'en savoir plus sur les données. Il y a tout d'abord les 2 composantes de la vitesse U et V, la moyenne mean et l'écart-type std. J'ai ensuite tracé U et V séparément et la norme de la vitesse.

Chapitre 3

Compréhension et exploitation du code

3.1 Explication des méthodes

3.1.1 POD

La SVD (Décomposition en Valeurs Singulières) fait partie des ROM et permet de décomposer une matrice de grande taille m×n en multiplication de matrices.

$$A = U\Sigma V^T$$

avec U=modes spatiaux, V^T=modes temporels et $\Sigma = diag(\sigma_i)_{1\leq i\leq m}$ valeurs propres.
La POD consiste à réaliser une troncature de la SVD qui permet de réduire la dimension.

$$A_r = \sum_{i=1}^{r} \sigma_i V_i^T \times U_i = \sum_{i=1}^{r} a_k(t) \times \phi_k(x) = f(x,t)$$

avec r=rg(A).
C'est une méthode linéaire donc elle est peut adaptée aux problèmes non linéaires, d'où le besoin de rechercher d'autres méthodes.

3.1.2 β-VAE

Pour la méthode β-VAE, il s'agit d'un algorithme qui prend en départ de grandes données, les compresse (Encoder) à faible dimension dans un espace latent et reconstruit les données à partir de l'espace latent (Decoder). L'encodeur calcule la moyenne μ_i et l'écart-type σ_i et c'est à partir de ces 2 valeurs que l'espace latent z est calculé, d'où le nom de Variationnal Auto Encoder.
Le β de la méthode correspond à un paramètre à estimer en plus dans la fonction de perte

$$\mathcal{L}(x) = \mathcal{L}_{\text{rec}} - \frac{\beta}{2} \sum_{i=1}^{d} \left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

3.1.3 Transformer et Attention (Attn)

Le Transformer est une méthode complexe, très récente et avec peu de sources (surtout pour la CFD) qui sert à faire de la prédiction temporelle. Dans le code, c'est le MultiHeadAttn surnommé easyAttn qui est la méthode novatrice utilisée et elle est comparée au

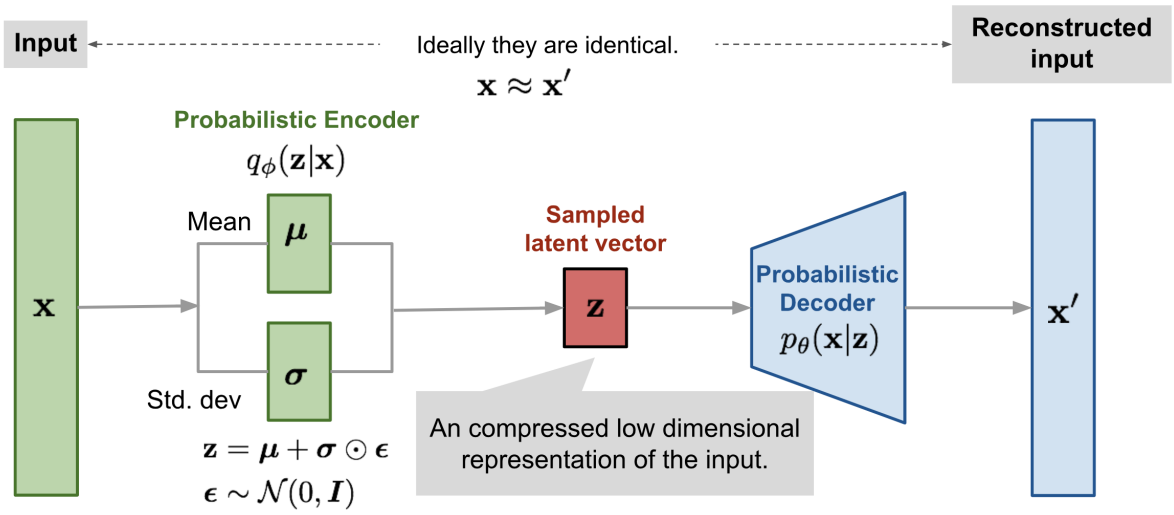


FIGURE 3.1 – Schéma d'un β-VAE

SelfAttn et Long Short Term Memory (LSTM) qui sont méthodes déjà connues dans la littérature scientifique. Le Transformer est l'algorithme qui utilise le mécanisme d'Attn. L'Attn est une catégorie de réseau de neurones qui consiste à parcourir les données, à les subdiviser en 3 catégories et à calculer les corrélations. La fonction Softmax est ensuite appliquée aux corrélations obtenues afin de sélectionner les paramètres les plus influents pour prédire l'espace latent. Le MultiHeadAttn utilise plusieurs têtes en même temps (4 dans le code) afin de parcourir plusieurs fois en parallèle les données, le SelfAttn utilise une seule tête et le LSTM parcourt également une seule fois les données tout en gardant la mémoire de ce qu'il a parcouru. La méthode MultiHeadAttn utilisée dans un Transformer est une évolution des autres méthodes afin de régler les difficultés de ces dernières en temps et en stockage.

3.1.4 Architecture globale

Le code est composé de 3 parties indépendantes : la POD, la β-VAE et le mécanisme d'attention (Attn).
Le but de l'algorithme est d'obtenir les 64 premiers espaces latents avec la partie encodeur, puis de prédire l'espace latent aux temps suivants avec le mécanisme d'attention, le tout afin de reconstruire les données de départ à partir du plus petit espace latent possible.

3.2 Exploitation du code

Le fichier main.py est composé de plusieurs paramètres : le nombre de Reynolds, ce qu'on demande à l'algorithme (run, train ou infer) et le réseau de neurones (easyAttn, selfAttn ou LSTM).

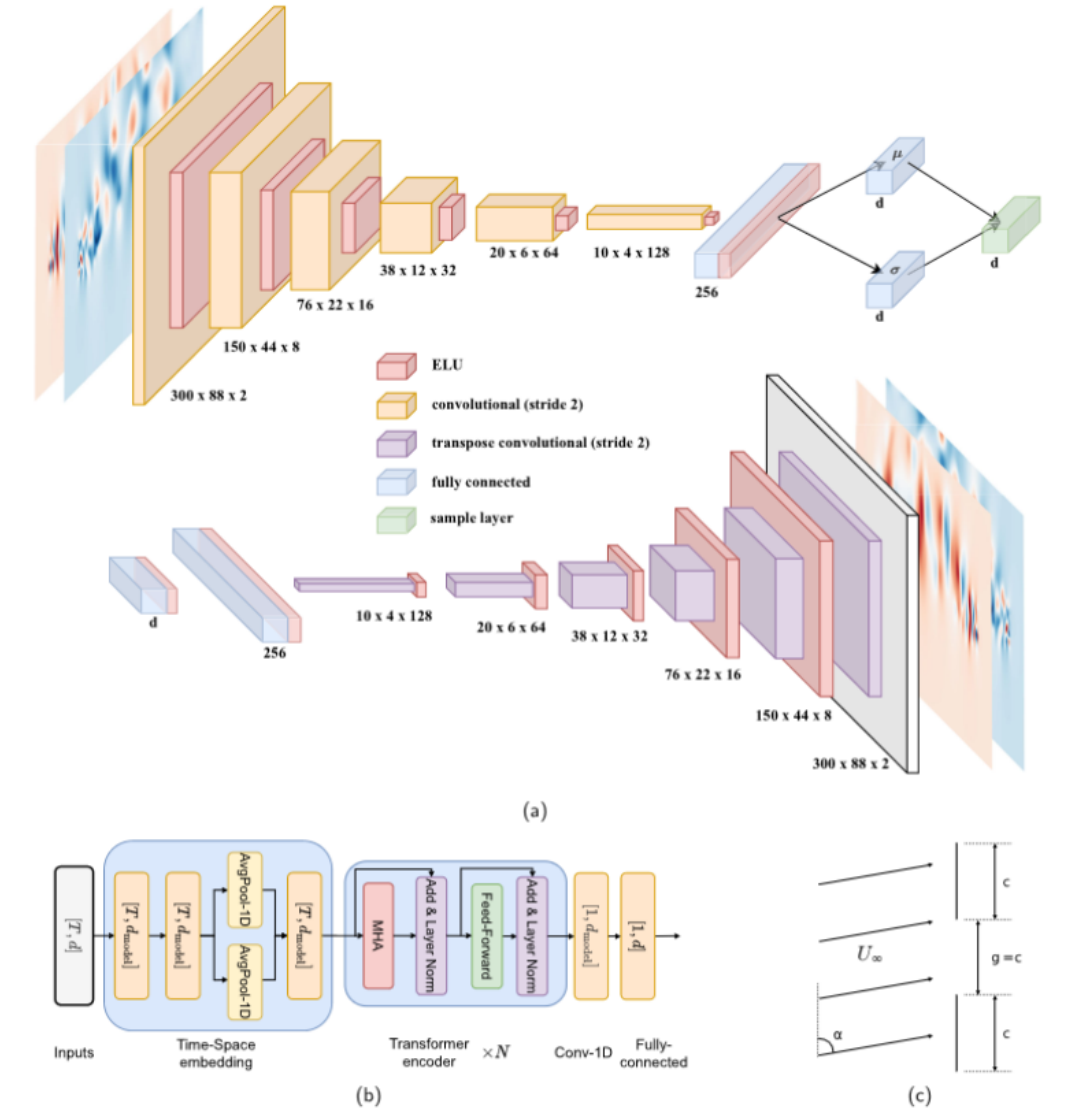


Fig. 9 | Model architectures and case schematic. a β-VAE encoder and decoder, (**b**) transformer. The dimension of the output for each layer has been indicated in each block. The symbols T and d_{model} denote the time-delay, MHA denotes multi-head attention. **c** Schematic representation of the numerical setup.

FIGURE 3.2 – Schéma de l'architecture du code

3.3 Modification du code

3.3.1 Debug

La partie POD ne s'exécutait pas donc j'ai perdu beaucoup de temps à la faire fonctionner sachant que les données étaient en 4D mais le code n'était pas adapté pour ce format. J'ai fini par conclure que ce n'était pas important puisqu'il s'agit juste de l'ancienne méthode employée et qu'elle sert juste de comparaison pour les nouvelles méthodes trouvées. De plus, Yassin Ajanif a déjà travaillé sur la POD pendant 1 an et demi et dispose donc d'un code que je peux utiliser si besoin.
Pour le reste du code, il y a juste eu quelques adaptations de versions de fonctions ou bouts de code.
Pour le 2^{me} exemple, le fichier de données de départs et le nombre de Reynolds étaient à changer.

3.3.2 Modifications faites

J'ai créé un fichier testStochastique.py afin de tester 2 réalisations du code à partir d'un même entraînement mais aussi pour comparer les prédictions et les données.

3.3.3 Utilisation du supercalculateur

J'ai tout d'abord exécuté le code via le terminal ou l'interface Spyder pour le 1er exemple afin d'obtenir les différents résultats du code.
Ensuite, pour le 2^{me} écoulement qui est plus complexe, le fichier de données est trop volumineux (30 Go) donc j'ai dû basculer mes calculs sur le supercalculateur Nautilus de l'école afin de programmer sur GPU. J'ai été formée par Yassin Ajanif pour cette partie car il y a des réglages à faire sur un fichier job.sh qui est comme un script qui s'exécute avec la commande 'sbatch' (comme le langage C), toutes les bibliothèques nécessaires à re-télécharger et on ne peut pas faire ce qu'on veut ni aller partout, il y a des restrictions de mémoire et de téléchargements. Les fichiers sont à envoyer du local ou de Git et ensuite les résultats sont à rebasculer en local pour les récupérer. Le calcul de la POD a fonctionné sur le supercalculateur pour le 1er exemple et tous les calculs du 2^{me} exemple.

3.4 Résultats

3.4.1 Re=40

Pour le VAE il n'y a pas vraiment de modes, mais pour visualiser l'effet de l'algorithme, un multiple d'un vecteur issu de la base canonique (de taille de l'espace latent, ici 2) est décodé (Figure 3.3.1).

La Figure 3.3.2 correspond à l'évolution temporelle des 2 variables latentes. Celle-ci semble périodique, ce qui est en accord avec le régime périodique de l'écoulement.

A partir d'un même entraînement de l'algorithme, on regarde la différence entre deux prédictions. L'erreur est comprise entre 10^{-7} et 10^{-9} et peut donc être considérée comme nulle. Donc à partir de deux espaces latents différents, mêmes $\mu et \sigma$ mais deux z différents, la reconstruction est la même.

Nous avons ensuite regardé l'erreur entre une prédiction issue de l'algorithme entier (β-VAE + Transformer), données prédites, avec une prédiction obtenue uniquement par β-VAE, données test.

La norme de l'erreur (Figure 3.4.3) semble elle aussi périodique et de l'ordre de 10^{-3} donc la différence entre les 2 méthodes pour cet écoulement périodique n'est pas très grande. Si l'on regarde le détail de l'erreur (Figure 3.4.1), elle est plus importante autour des 2 obstacles, ce qui est logique puisque l'écoulement commence à faire de petits tourbillons à cet endroit.
La Figure 3.4.2 montre l'allure de (u,v) à un temps arbitraire dans le domaine étudié et

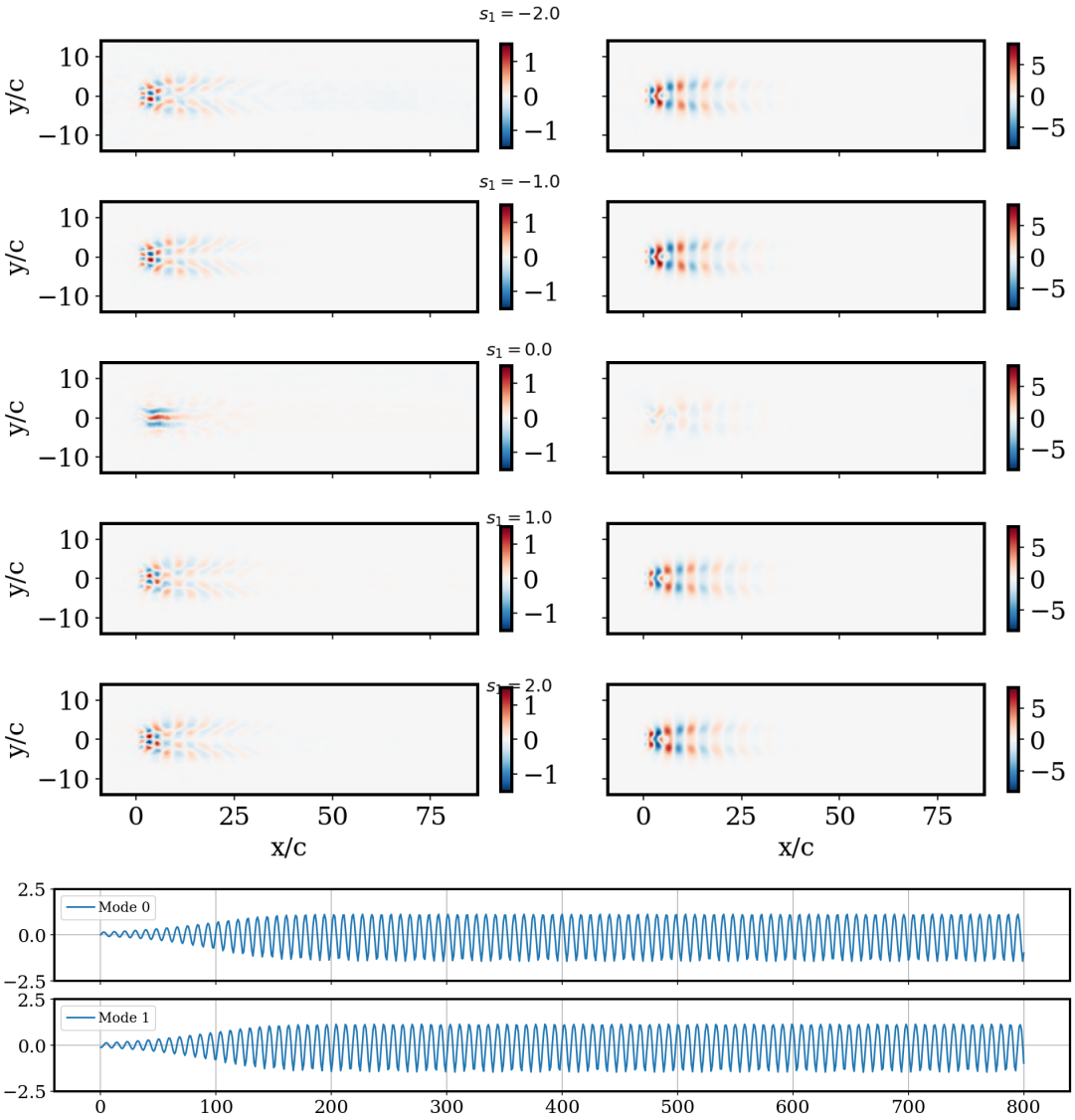


FIGURE 3.3 – Modes VAE et évolution temporelle de l'espace latent

permet de mieux comprendre les endroits où l'erreur entre les deux algorithmes augmente soit juste après le franchissement des deux obstacles.

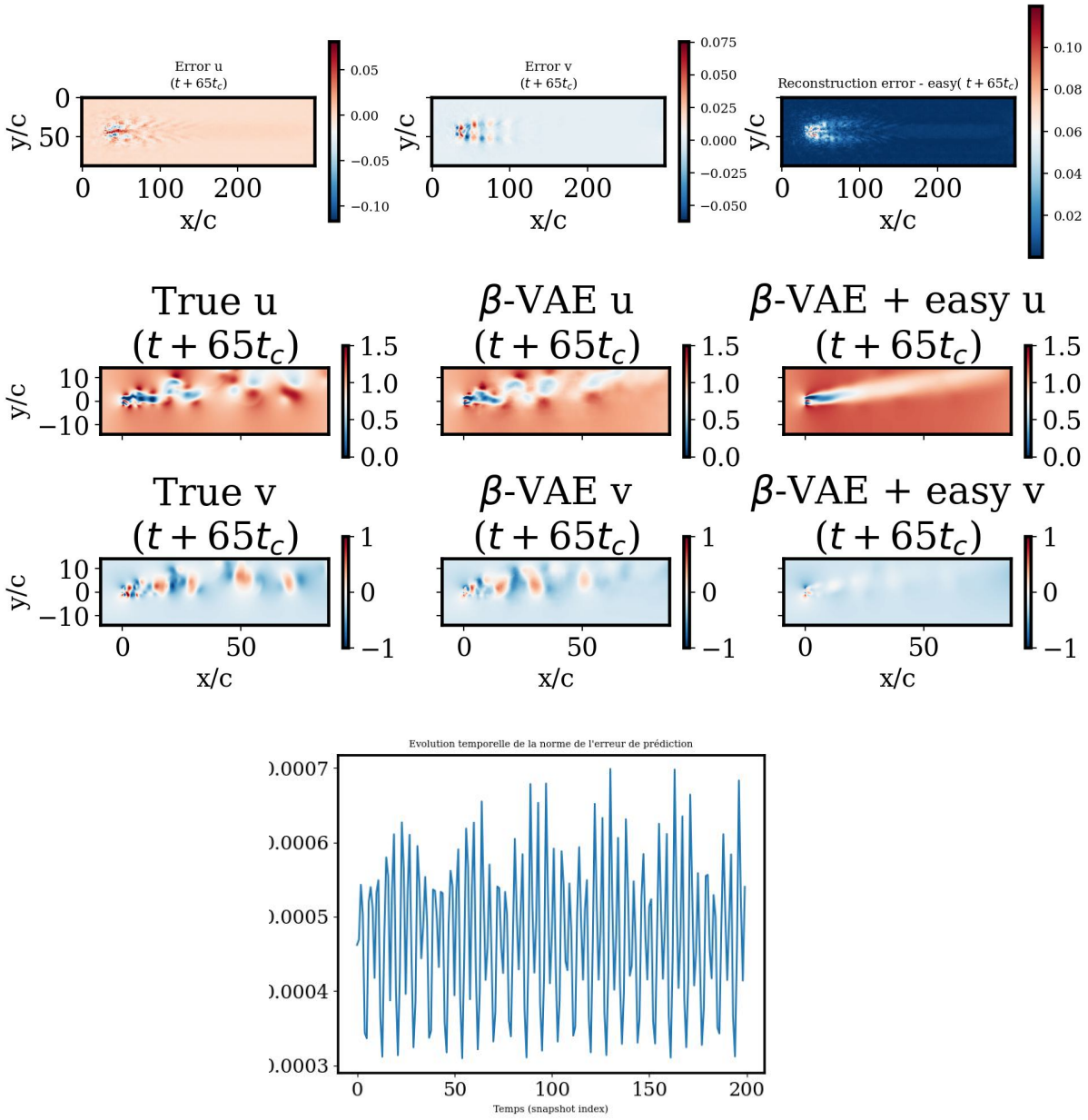


FIGURE 3.4 – Comparaison Transformer vs VAE

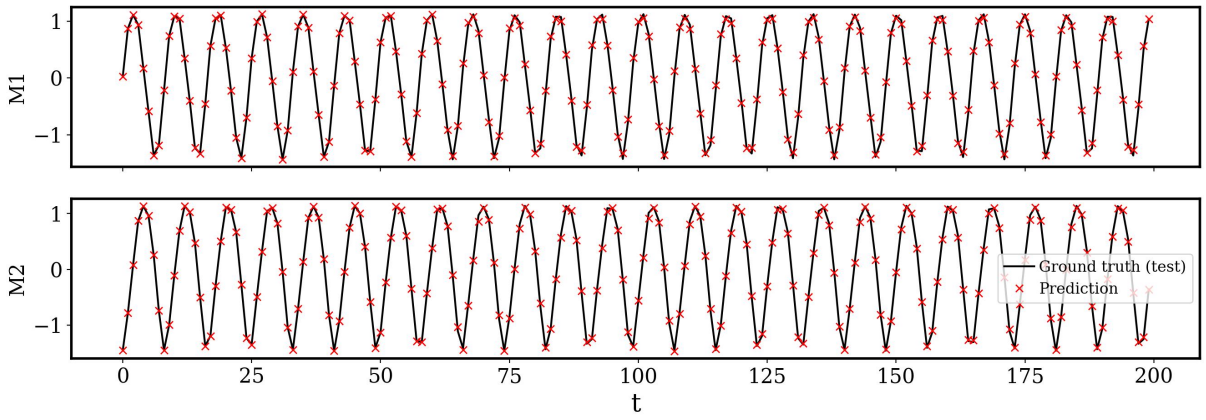


FIGURE 3.5 – Comparaison de l'évolution temporelle des 2 premiers modes de l'espace latent

La comparaison de l'évolution temporelle des deux premiers modes de l'espace latent (Figure 3.5) montre qu'il n'y a pas de différences entre la prédiction et le test.

3.4.2 Re=100

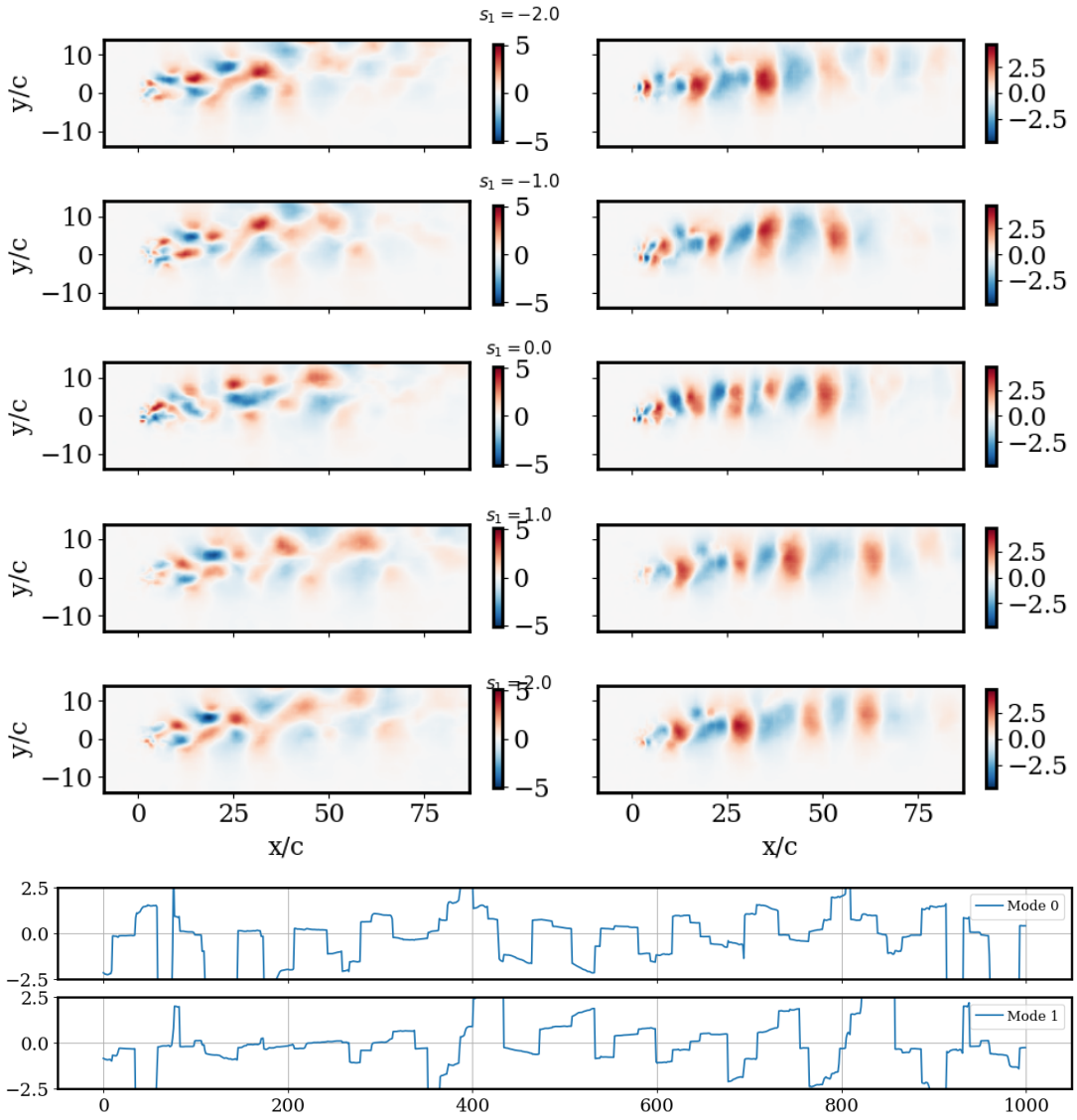


FIGURE 3.6 – Modes VAE et évolution temporelle de l'espace latent

La Figure 3.6.1 montre les premiers "modes" du VAE, ce qui montre que même avec des vecteurs de la base canonique, l'algorithme rend une reconstruction complexe et non linéaire (en opposition à la POD).

La Figure 3.6.2 correspond à l'évolution temporelle des 2 variables latentes. Celle-ci n'est plus périodique et montre que l'écoulement tend vers un régime turbulent.

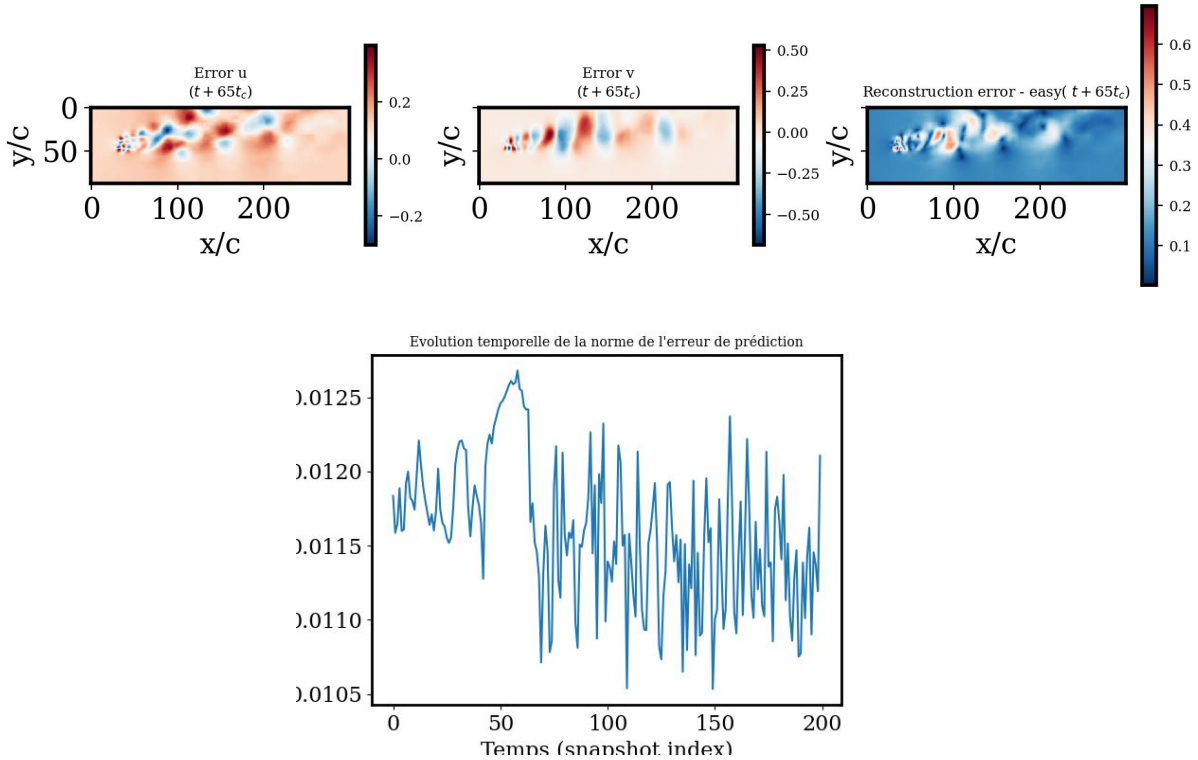


FIGURE 3.7 – Erreur entre deux prédictions issues d'un même entraînement

A partir d'un même entraînement de l'algorithme, on regarde la différence entre deux prédictions. La norme de l'erreur est faible mais beaucoup plus importante que pour l'exemple à Re=40 (Figure 3.7.2). Le détail avec chaque composante montre qu'à partir du même entraînement, les deux prédictions sont complètement différentes (Figure 3.7.1).

Nous avons de nouveau regardé l'erreur entre une prédiction issue de l'algorithme entier (β-VAE + Transformer), données prédites, avec une prédiction obtenue uniquement par β-VAE, données test.

La norme de l'erreur reste faible mais ne semble plus périodique (Figure 3.8.3 et 3.8.1).

L'algorithme VAE prédit mieux que l'algorithme avec Transformer pour cet exemple d'après la Figure 3.8.2 .

L'algorithme utilise les 64 premiers espaces latents afin de prédire celui d'après. On voit très bien sur la Figure 3.9 que pour $t \leq 64$, la prédiction Transformer coïncide avec les données test VAE mais après les 2 méthodes ne donnent pas du tout le même résultat.

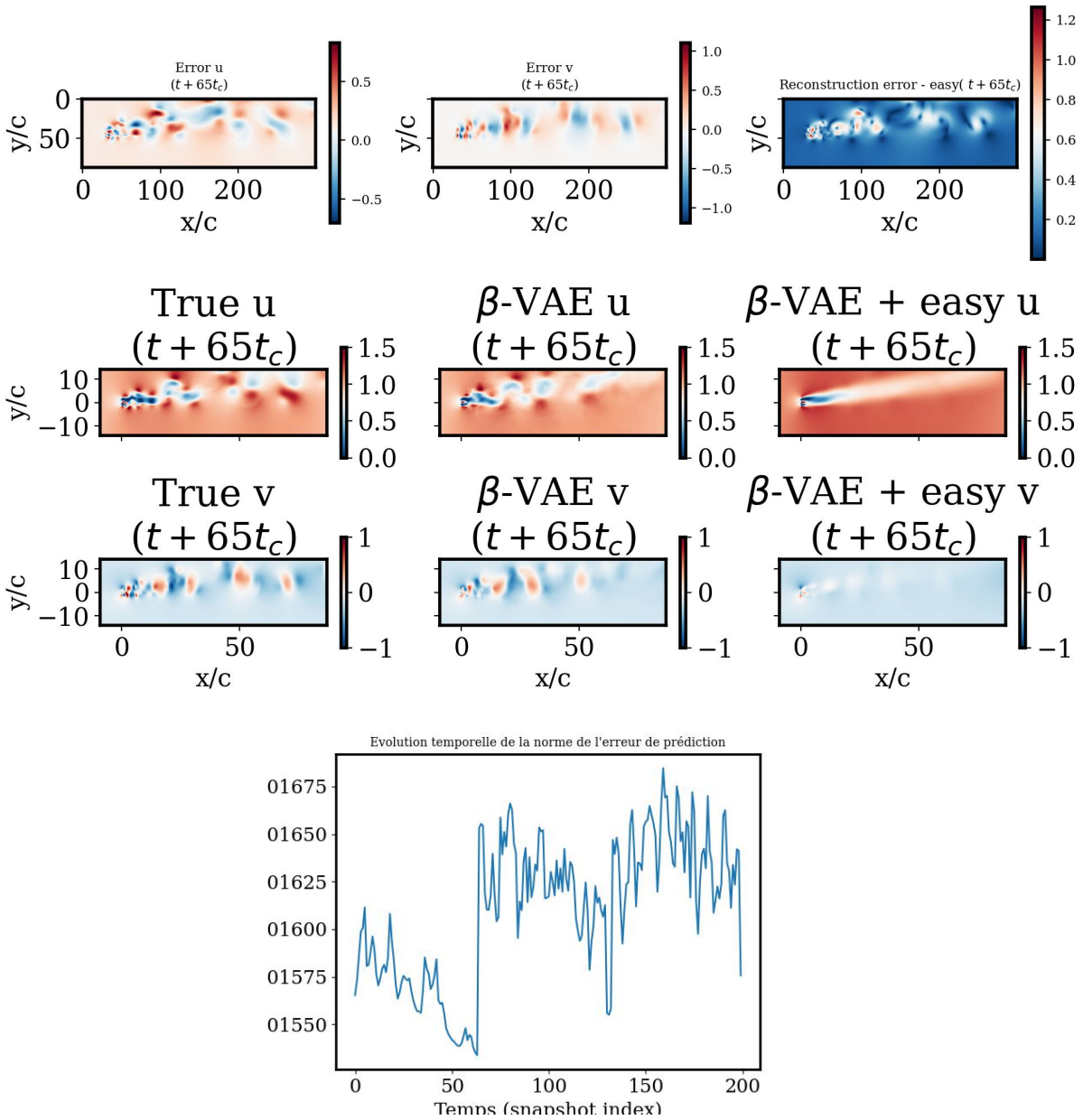


FIGURE 3.8 – Comparaison Transformer vs VAE

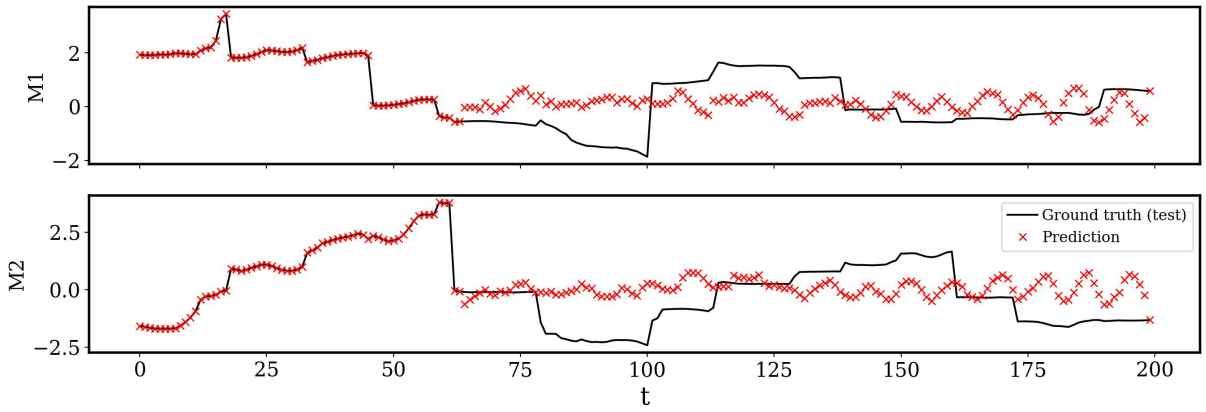


FIGURE 3.9 – Comparaison de l'évolution temporelle des 2 premiers modes de l'espace latent

Chapitre 4

Conclusion

L'algorithme Transformer montre de bons résultats pour la prédiction d'écoulements périodiques mais l'algorithme VAE reste plus performant pour l'écoulement transitoire avec les paramètres choisis. La dimension de l'espace latent a été fixée à 2 et pourrait être augmentée lorsque le nombre de Reynolds augmente.

Perspectives :
Je n'ai pas modifié tous les paramètres et certains peuvent être changés (dimension de l'espace latent, nombre de modes, réseau de neurones...).
Maintenant que la méthode a été testée et rend un résultat, elle est prête a être appliquée aux données envoyées par le laboratoire indien.
Seulement 2 méthodes ont été utilisées mais d'après la recherche bibliographique effectuée au début du stage, il existe d'autres méthodes à tester.

GitHub disponible au lien suivant : `https://github.com/ilona35000/stageM1_2025`

Sources

Burgess, Christopher (2018). "Understanding disentangling in -VAE". In.

Coscia, Dario (2024). "Generative adversial reduced order modelling". In.

Dubois, Pierre (2022). "Utilisation de l'apprentissage automatique en mécanique des fluides". Thèse de doct. Lille.

Eivazi, Hamidreza (2022). "Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows". In : *Elsevier*.

Fehli, Ghazi (2023). "Interpretable Sentence Representation with Variational Autoencoders and Attention". Thèse de doct. Sorbonne Paris Nord.

Fukami, Kai (2019). "Super-resolution reconstruction of turbulent flows with machine learning". In.

Geneva, Nicholas (2021). "Transformers for modeling physical systems". In : *Neural Networks*.

Heiland, Jan (2024). "Convolutional autoencoders, clustering, and POD for low-dimensional parametrization of flow equations". In : *Elsevier*.

Kingma, Diederik (2019). "An Introduction to Variational Autoencoders". In : *Foundations and Trends in Machine Learning*.

Lestandi, Lucas (2018). "Approximations de rang faible et modèles d'ordre réduit appliqués à quelques problèmes de la mécanique des fluides". Thèse de doct. Bordeaux.

López Carranza, Santiago Nicolás (2012). "Transition laminaire-turbulent en conduite cylindrique pour un fluide non Newtonien". Thèse de doct. Lorraine.

Manceau, Remi (2024). *Modelisation de la turbulence pour la CFD*. Rapp. tech.

Marchesse, Yann (2023). *Modélisation de la turbulence*. Rapp. tech. Lyon : ECAM.

Nicoud, Franck (s. d.). *1MKFLU - MI4 Mécanique des Fluides*. Cours. Montpellier.

Patil, Aakash (2023). "Deep Learning-Assisted Modelling of Turbulence in Fluids". Thèse de doct. Mines Paris - PSL.

Ren, Pu (2023). "PhySR : Physics-informed deep super-resolution for spatiotemporal data". In : *Elsevier*.

Sanchis-Agudo (2023). "Easy attention : A simple attention mechanism for temporal predictions with transformers". In : *arXiv preprint arXiv :2308.12874*.

Solera-Rico, Alberto (2024). "-Variational autoencoders and transformers for reduced-order modelling of fluid flows". In : *Nature communications*.

Tissot, Gilles (2013). "Model reduction using Dynamic Mode Decomposition". In : *Comptes rendus Mécanique*.

Turner, Richard (2024). "An Introduction to Transformers". In.

Vaswani, Ashish (2017). "Attention is all you need". In : *Google*.

Wang, Yuning (2024). "Towards optimal -variational autoencoders combined with transformers for reduced-order modelling of turbulent flows". In : *International Journal of Heat and Fluid Flow*.