Notes bibliographiques :
Utilisation de ML pour la turbulence

Auteur	Travaux
Ling and Templeton (2015)	Used Random Forests (RF) to predict regions of high model from uncertainty in RANS results.
Ling et al. (2016a)	Use of RF with Gini Impurity Score, max_depth = 15 and 500 DT whose output are averaged, and NNwith sigmoid activation function. In the previous work, Ling et *al.* predict the Reynolds Stress Anisotropy $\mathbf{A} = \frac{1}{2k}\mathbf{u} \bar{\otimes} \mathbf{u} - \frac{1}{3}\mathbf{Id}$ en un point à partir des images de $\mathbf{S}$ et $\mathbf{R}$ par les vecteurs d'une base d'invariants bien choisie, en tout point du domaine, ou aux points dont l'incertitude est la plus grande. Les entrées peuvent être aussi bien les invariants de la base que les 9 composantes non nulles cumulées de $\mathbf{S}$ et $\mathbf{R}$.
Ling et al. (2016b)	Use of NN embbeded with an invariant Tensor Layer element-wise mutliplied with the last HL of the Network. Use of leakyReLu act function. Trained with BPGD, the output is post-processed (see part.2, p.157). It has 8 HL, with each 30 nodes except for the last HL that has 10 nodes and $learning\_rate = 2.5 \times 10^{-7}$. ≪*Cross Validation ensures robustness of the TBNN*≫.
Milano and Koumoutsakos (2002)	Used DNS results for a turbulent channel flow to train a NN to reconstruct the near wall flow
Tracey et al. (2013)	ML algorithms (??) to model the Reynolds stress anisotopy
Tracey et al. (2015)	NN to mimic the source terms from the SA turbulence model
Duraisamy et al. (2015)	NN and GP to model intermittency in transitional turbulence
Zhang and Duraisamy (2015)	Used NN and GP to model turbulence Production in channel flow

Singh et al. (2017)	Use NN to augment SA model to predict turbulent flow over airfoils (strong adverse pressure gradient). First of all use adjoint-based full field inference to infer β the correction needed between SA and DNS or Exp. The NN proposed here has 2 HL with 100 nodes each, using sigmoid act function. The training used BPGD minimizing an Sum Squarred Error problem.
Wu et al. (2018)	Use of RF to predict $(\Delta \log k, \Delta\xi, \Delta\eta, h_1, h_2, h_3)$, 6 components thanks to 50 invariant inputs whose 47 build from Pope (1975) and Ling et al. (2016b) similar ideas but widened for taking into account changes in TKE or Pressure & three more added to gain physical interpretability and help the RF to construct new leafs. The number of trees of the Forest was set at 300.

Singh et al. (2017) :

L'idée de cet article peut être schématisée par la figure suivante (extraite de l'article) :

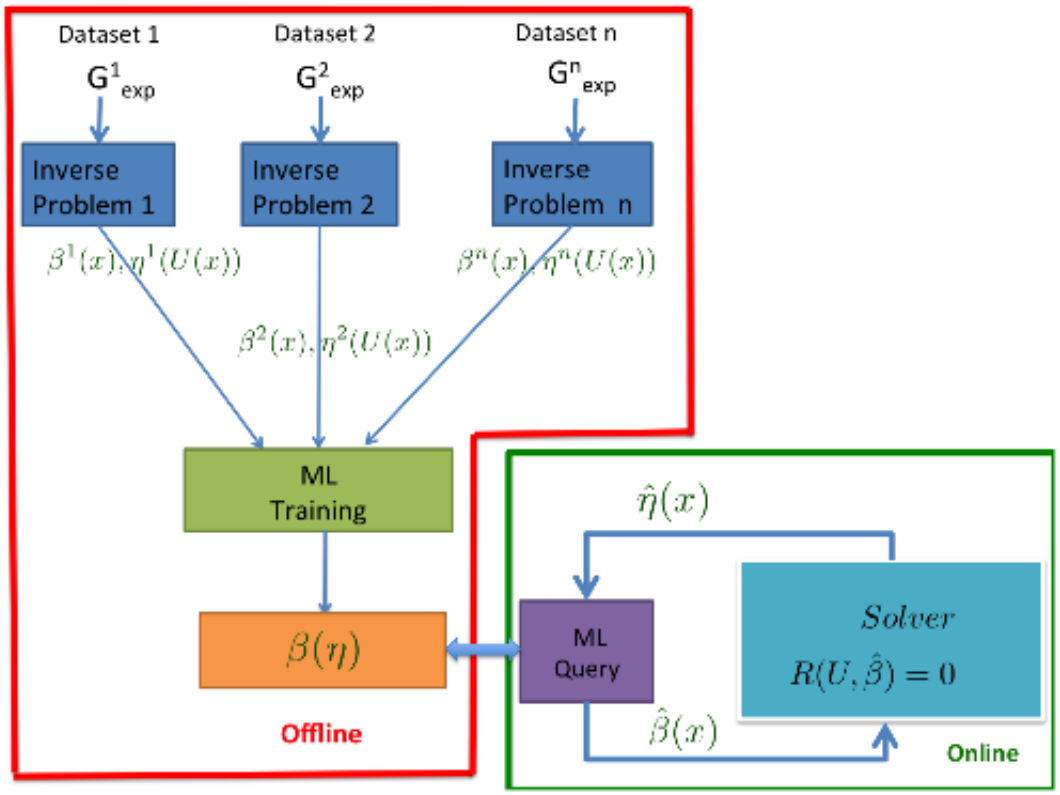


FIGURE 1 – Schéma des trois étapes pour augmenter les modèles de la turbulence

Le modèle SA peut être représenté par

$$\frac{D\tilde{\nu}}{Dt} = P\left(\tilde{\nu}, \mathbf{U}\right) - D\left(\tilde{\nu}, \mathbf{U}\right) + T\left(\tilde{\nu}, \mathbf{U}\right) \tag{0.0.1}$$

La grandeur $\tilde{\nu}$ est liée avec la viscosité turbulente ν_t de Boussinesq (Voir annexe 3 de l'article). L'idée de l'article et des méthodes développées par Parish and Duraisamy (2016) est d'inférer un terme de correction. En effet, les modèles (SA, $k-\varepsilon$) simplifient certains termes de Navier-Stokes (DNS). Du coup, en rajoutant un terme dynamiquement adaptable au cours du processus d'inférence Baysienne, on serait à priori capable de retrouver la véritable physique. Le grand défi réside dans la généralisation de cette correction, et c'est là où l'injection d'une prédiction via les MLA(lgorithmes) devient inévitable.
Ce terme est injecté dans le terme de production. Le modèle s'écrit alors :

$$\frac{D\tilde{\nu}}{Dt} = \beta\left(\mathbf{x}\right) P\left(\tilde{\nu}, \mathbf{U}\right) - D\left(\tilde{\nu}, \mathbf{U}\right) + T\left(\tilde{\nu}, \mathbf{U}\right) \tag{0.0.2}$$

Leur inversion peut sembler différente de celle effectuer dans Parish and Duraisamy (2016) ou Tarantola (2005) mais finalement la régression de Tikhonov (L1 et L2) avec un facteur de régression $\lambda = 4 \times 10^{-4}$ (similaire dans le sens au learning rate dans le NN) peut être ramener à ce qui est décrit dans les articles précédemment cités.
Dans notre article, l'inférence se fait sur le coefficient de portance C_l, la fonction de coût s'écrit alors :

$$\mathcal{J} = \min_{\beta} \left[\sum_{j=1}^{N_d} \left[C_{l,\text{exp}} - C_l\left(\beta\right)\right]^2 + \lambda \sum_{n=1}^{N_m} \left[\beta\left(x_n\right) - 1\right]^2 \right]$$

On peut expliquer les termes de cette équation :

- N_d : le nombre de points où sont réalisées les mesures expérimentales. Le coefficient de portance étant un champ, ces valeurs scalaires dépendent dre la position sur l'aile (par exemple)
- N_m : le nombre de volumes de contrôle, c'est-à-dire le nombre de points de *maillage* sur l'aile mais également autour, puisque dans ce genre de cas le contexte à son importance.
- λ : ampleur de la régression L1. Par rapport à l'article Parish and Duraisamy (2016) λ peut être vu comme le rapport des covariances observées sur *prior* :

$$\lambda = \frac{C_{\text{pri}}^{-1}}{C_{\text{obs}}^{-1}} = \frac{C_{\text{obs}}}{C_{\text{pri}}}$$

  Sachant cela, les carrés représentent alors le produit scalaire et on peut faire un analogie avec l'expression de la fonction de coût Parish and Duraisamy (2016).
  D'un point de vue plus algorithmique, augmenter λ revient à augmenter la *sparsity* des données inférées *i.e* ne pas temnir compte des actions faibles de certains éléments.

Il est également mentionné qu'en utilisant le coefficient de pression de surface C_p à la place de C_l on retrouvait le même β. Ce ci souligne le fait que l'inférence permet de déduire le contexte manquant pour retrouver la bonne physique. On aurait pu également chercher à minimiser la différence d'autres termes, le β aurait été le même.
Si l'inférence permet de retrouver le β tenant compte des informations manquantes de la simulations RANS par rapport aux DNS ou aux expériences, ceci reste propre à la simulation étudiée. Pour généraliser ces *informations* en *connaissance*, l'utilisation des MLA est ici prônée.
De manière générale, le training d'un réseau est très important car il conditionne la capacité de généralisation et la justesse des prédictions. En s'assurant que les entrées soient les plus *générales possibles* et qu'elles sont le plus adaptées à l'entité que l'on veut simuler, on augmente les chances de créer un réseau applicable sur plusieurs considérations.
Tracey et al. (2015) partie 3 B explique l'importance de l'adimensionnement et propose des entrées pour le modèle SA.
Tout d'abord le terme de production dépend des quantités locales

$$\nu,\ \tilde{\nu},\ \Omega,\ d,\ N = \frac{\partial \tilde{\nu}}{\partial x_i}\frac{\partial \tilde{\nu}}{\partial x_i}$$

Cependant ces grandeurs sont dimensionnées et peuvent donner des valeurs différentes pour deux écoulement dynamiquement similaires. D'un point de vue ML, utiliser ces quantités empêcherait une généralisation optimale.
On expose également deux types d'adimensionnalisation (1) globale (2) en utilisant des quantités locales. On présente ici la deuxième façon de faire. On définit $\tilde{\nu} + \nu$ et d deux échelles locales, on écrit :

$$\overline{\Omega} = \frac{d^2}{\tilde{\nu} + \nu}\,\Omega, \qquad \overline{N} = \frac{d^2}{(\tilde{\nu} + \nu)^2}\,N,$$

$$\overline{P} = \frac{d^2}{(\tilde{\nu} + \nu)^2}\,s_p \qquad \overline{D} = \frac{d^2}{(\tilde{\nu} + \nu)^2}\,s_d = \left(\frac{\chi}{\chi + 1}\right)^2 c_{w1} f_w,$$

$$= c_{b1}\,(\,1 - f_{t2}\,)\left(\frac{\chi}{\chi + 1}\right)\left(\bar{\Omega} + \frac{1}{\kappa^2}\frac{\chi}{\chi + 1} f_{t2}\right) \qquad \overline{s}_{cp} = \frac{d^2}{(\tilde{\nu} + \nu)^2}\,s_{cp} = \frac{c_{b2}}{\sigma}\,\overline{N}$$

Avec ν la viscosité cinématique, $\tilde{\nu}$ la variable traitée par le modèle S-A et d la distance par rapport au mur, Ω la magnitude de la vorticity, $\chi = \tilde{\nu}/\nu$, f_{t2} une fonction de χ, f_w une fonction de $\overline{\Omega}$ et de χ, et enfin c_{b1} et c_{w1} deux constantes.
Ces adimensionnalisations permettent alors de définir des entrées adaptées :

$$\eta = \left\{ \overline{\Omega}, \chi, S/\Omega, \tau/\tau_{\text{wall}}, P/D \right\}$$

La méthode utilisée ici est un NN avec 3 HL de 100 HN et des activations de type sigmoid.
Les données utilisées pour l'inversion sont issues de simulations autour des profils d'ailes d'avion. Le facteur β est inféré pour des combinaisons
{Nombre de Reynolds, Angle d'attaque} différentes.
On calcule ensuite les différentes caractéristiques dicutées plus haut, et on entraîne notre réseaux avec les correspondances η/β.
On souligne ici que le but de cet article n'était pas de retrouver le bon coefficient de portance mais de corriger l'ensemble de la simulation, une preuve étant les graphes des coefficients de portées de pression de surface.

Duraisamy et al. (2015) :

Dans cet article on veut trouver un facteur correctif pour reconstruire les intermittences de la tubulence. La façon de faire passe par de l'inférence puis une généralisation de cette inférence en utilisant les réseaux de neurones et les GP.
Ils insistent sur le fait que les entrées doivent être des quantités locales et adimensionnés (on pourra consulter Tracey et al. (2015) partie 3 Méthodologie).
La raison de cette insistance est la suivante : on veut que la sortie $f(\mathbf{q})$ soit utilisable dès que $\mathbf{q}$ se réalise, peu importe le problème.
Les paramètres influant le plus sur la sortie de la ML sont choisis au travers le procédé *hill-climbing* Kohavi and John (1997) (à lire). La fonction d'erreur est la SSE qui compare la sortie prédite par rapport à la sortie atendue (lors de l'apprentissage).
Les hyperparamètres amenant à une IA optimale sont déterminés par 10-fold Cross Validation (voir Müller and Guido (2016), chapitre 5 et 6). Cette cross-validation est donc faite pour déterminer les valeurs des paramètres qui ont été sélectionnés par le *hill-climbing*.

À la recherche des invariants

D'après (Ling et al., 2016a) les *features* ainsi choisis ne respectent pas les invariances en rotation et que le modèle n'a été entraîné que pour une seule configuration d'écoulement.
Il s'est avéré qu'entraîner un NN avec la même base de données, en "orientant" les entrées tout en les associant à la même sortie, on forçait les invariances rotationnelles (voir Lefik and Schrefler (2003)).
Tracey et al. (2013) et Ling and Templeton (2015) utilisent des entrées qui sont des invariants Galiléens, le choix de ces entrées a été fait en suivant "le sens physique".

Invariance Galiléenne

Les invariances galiléennes traduisent l'invariance des lois de mouvement par translation ou rotation du référentiel. Dans le cas de la mécanique des fluides, toute variable scalaire de l'écoulement (pression, norme de la vitesse) sera invariant par translation, rotation ou réflexion du repère de référence.

Les propriétés de symétrie d'un système physique représentent des contraintes à partir desquelles on peut déterminer les lois de mouvement du système. On comprend alors l'importance de respecter les symétries d'un problème. Inversement, les lois de mouvement sont intrinsèquement liées avec des propriétés d'invariance (qui traduisent les symétries du problème).
Une fonction est invariante par rapport à une certaine transformation, si lorsque les entrées sont assujetties à ces transformations, leur image par cette fonction reste inchangée.
Prenons le groupe des matrices définies comme étant des rotations en 3D : $\mathcal{SO}(3)$. Une fonction scalaire $f(\vec{v}, \mathbf{A})$ est dite «rotationnellement invariante» si

$$f\left(\mathbf{Q}\,\vec{v},\, \mathbf{Q}\mathbf{A}\mathbf{Q}^{\mathrm{T}}\right) = f(\vec{v}, \mathbf{A}),\ \forall \mathbf{Q} \in \mathcal{SO}(3)$$

Le groupe $\mathcal{SO}(3)$ est d'importance capitale, en effet puisqu'il représente toutes les rotations possibles en être invariant est la définition de l'isotropie. Ainsi, une fonction f invariante par ce type de transformation sera dite isotropique.
On peut juxtaposer une deuxième définition à ce type de fonction : une fonction isotropique est une fonction qui peut être projetée dans une base d'invariants du groupe $\mathcal{SO}(3)$, par définition une combinaison linéaire d'invariants de $\mathcal{SO}(3)$ sera un invariant de ce groupe (une fonction isotropique) l'inverse moins évident reste vrai.
De façon générale, la base $(\mathrm{Tr}(\mathbf{A}), \mathrm{Tr}(\mathbf{A}^2), \mathrm{Tr}(\mathbf{A}^3))$ est une base d'invariant pour le groupe isotropique. On appelle cette base la base intègre ou *integrity basis* (IB).

Ling et al. (2016a)

Dans cette étude, les auteurs cherchent à déterminer si les random forest ou les NN peuvent conserver les lois de conservations durant la phase de prédiction.
Ils proposent deux méthodes pour répondre à cette question. Ces méthodes cherchent à prédire la valeur de l'invariant du tenseur d'anisotropie dans en tout point. La comparaison de ces prédictions avec les véritables valeurs de cette invariant pourra nous donner une première réponse à la problématique considérée. Le tenseur d'anisotropie, $\mathbf{A}$, s'écrit

$$\mathbf{A} = \frac{1}{2k}\overline{\mathbf{u} \otimes \mathbf{u}} - \frac{1}{3}\mathbf{Id}$$

L'invariant considéré est le deuxième (issu du théorème de Cayley-Hamilton) :

$$\mathrm{II}_a = \mathrm{Tr}\left(\mathbf{A}\right)^2 - \mathrm{Tr}\left(\mathbf{A}^2\right)$$

Ces deux méthodes diffèrent cependant en deux points : la façon dont l'invariance par rotation est introduite qui impact le type et le nombre d'entrée des IA.
On résume les deux méthodes en quelques points

Injection des invariants :

- Construction d'une base intègre d'invariants ;
- Injection pour chaque point des valeurs de ses invariants et les faire correspondre avec un invariant du tenseur d'anisotropie ;

Induction de l'invariance :

- Rotation des éléments de la base données selon un angle en 2D et 3 angles en 3D (angles d'Euler) ;
- Injection des composantes de $\mathbf{S}$ et $\mathbf{R}$ pour l'ensemble de la nouvelle base de données en chaque point ;

· Prédiction de II_a et évaluation de l'erreur avec la métrique suivante :

$$E = \frac{\sum |\mathrm{II}_{a,\mathrm{DNS}} - \mathrm{II}_{a,\mathrm{ML}}|}{\sum \mathrm{II}_{a,\mathrm{DNS}}}$$

On a donc trois cas : un dans lequel la base d'invariants est sélectionnée en entrée, un cas d'induction d'invariance par rotation 2D, et un troisième dans lequel les rotations sont en 3D.

Dans les cas des *inductions* par rotation 3D, le Random Forest fournit des résultats moins convainquant que le NN, traduisant le fait que ce dernier semble être capable d'apprendre les invariances des données, à l'inverse du premier. Même constat (nuancé tout de même) pour les rotations 2D.
Les évaluations d'erreur dans le cas d'injection de l'invariance sont toujours les meilleures pour le RF, idem pour le NN si ce n'est pour les rotations 2D.
Les résultats étant quand même similaires (12% d'erreur entre les deux méthodes), les auteurs comparent les temps de calculs et la mémoire utilisée lors des différents cas :

Algorithm	Compute time (CPU Hours)	**Memory usage**
Invariant **RF** / **NN**	0.03 / 7	0.4 **GB** / 0.1 **MB**
10 - 2D Rotations : **RF** / **NN**	0.3 / 105	0.9 **GB** / 2 **MB**
10 - 3D Rotations : **RF**/ **NN**	46 / 1176	13 **GB** / 2 **MB**

TABLE 2 – Comparaisons des coûts de calcul et de mémoire pour RF et NN

Quelques remarques :

1) - RF : chaque branche se subdivise selon la logique «Si.. Alors... Sinon» en calculant le **gain en information** (Giny Impurity Score) de la possible branche. Les cas de régression avec cette logique nécessite logiquement beaucoup de mémoire.
On peut également limiter le nombre de nœuds maximal (on parlera de profondeur **tree's depth**) pour une branche. Dans l'article la profondeur est fixée à 15, entraînant 2^{15} soit $32,768$ possibilités, pour chaque arbre de la «forêt», d'où l'énorme taille de mémoire mobilisée.

2) - NN : La méthode d'optimisation utilisée est une version stochastique de la troncature de Newton. La fonction d'activation pour ce problème de régression fortement non linéaire est une sigmoïde, il semblerait que le relu ou le leakyrelu pourrait accélérer (**sparse**) l'apprentissage et améliorer les capacités de généralisation d'un tel réseau.
Le réseaux est de la forme $N_{in} - 6N_{in} - 12N_{in} - 6N_{in} - 2N_{in} - 1$ désigné comme étant la meilleure architecture obtenue au cours d'un processus de calibration.

3) - La base intègre citée dans le cas de l'injection de l'invariance est :

$$\mathrm{Tr}\left(\mathbf{S}^2\right),\ \mathrm{Tr}\left(\mathbf{S}^3\right),\ \mathrm{Tr}\left(\mathbf{R}^2\right),\ \mathrm{Tr}\left(\mathbf{R}^2\mathbf{S}\right),\ \mathrm{Tr}\left(\mathbf{R}^2\mathbf{S}^2\right),\ \mathrm{Tr}\left(\mathbf{R}^2\mathbf{S}\mathbf{R}\mathbf{S}^2\right)$$

Finalement ce papier a exploré la capacité du RF et du NN d'apprendre les proriétés d'invariance.
D'après eux construire une base d'invariants fournit systématiquement des caractéristiques d'entrée «d'utilisation immédiate» pour l'application des ML sur des modèles physiques. De plus, comme montrer dans l'article, utilisé ce genre de base permet l'utilisation de modèles d'IA moins complexes tout en gardant des performances proches du NN, le meilleur modèle de l'article.
La nécessité de l'utilisation d'une base intègre réside également dans le gain en temps de training, pour des résultats très similaires.

Ling et al. (2016b)

Les auteurs continuent sur la lancé de l'article précédent et s'orientent plutôt vers une forme particulière de réseau de neurones. La particularité de ces réseaux résisde dans leur flexibilité qui leur permet d'apprendre quasiment n'importe quoi.
Ils testent deux types de réseaux de neurones : le MLP classique et le TBNN qui est un réseau inédit représenté par la figure suivante :

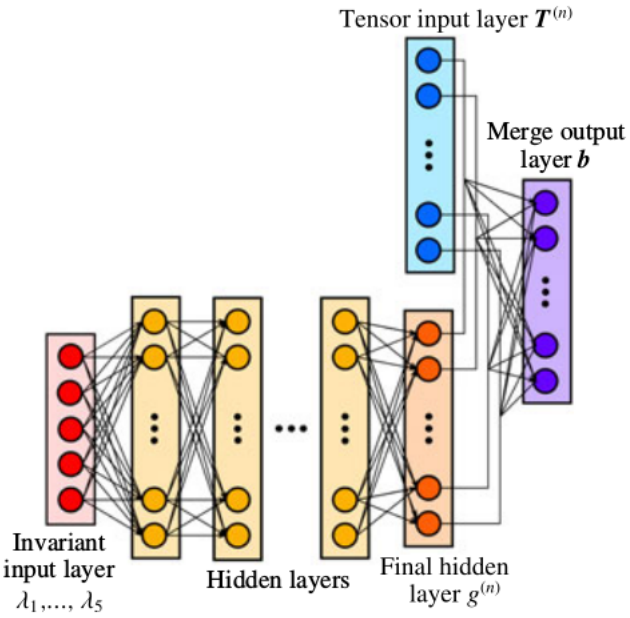


FIGURE 2 – Tensor embedded NN from (Kutz, 2017) et (Ling et al., 2016b)

Cette forme de neurones à pour but l'écriture du tenseur anisotropique comme une combinaison linéaire d'une base de tenseur bien particulière. Les coefficients de cette combinaison sont différentes fonctions des invariants du problème. L'article se base sur les travaux de Pope (1975) et de Smith (1965).
On alors propose de modifier l'hypothèse de Boussinesq qui consistait à écrire

$$\overline{u_i' u_j'} = \frac{2}{3} k \delta_{ij} - 2\nu_t \bar{S_{ij}}$$

Il propose plutôt de calculer

$$\frac{\bar{!} u_i' u_j'}{2k} = \frac{1}{3}\delta_{ij} + \sum_{\lambda} G^{\lambda} T_{ij}^{\lambda}$$

Avec une forme bien particulière pour G et T. La première est une fonction des invariants du problème qui sont dans le cas tridimensionnel :

$$\mathbf{\Omega} = \left\{ \{\mathbf{S}^2\}, \{\mathbf{R}^2\}, \{\mathbf{S}^3\}, \{\mathbf{R}^2\mathbf{S}\}, \{\mathbf{R}^2\mathbf{S}^2\} \right\}$$

La notation $\mathbf{\{\}}$ symbolise la trace du tenseur concerné. À noter, que l'invariant $\{\mathbf{S}\}$ n'est pas utilisé directement (puisque nul) mais indirectement au travers l'utilisation du $\{\mathbf{S}^3\}$.
Également, les tenseurs considérés $\mathbf{S}$ et $\mathbf{R}$ sont définis par

$$\mathbf{S} = \frac{k}{2\varepsilon}\left(\nabla_x \mathbf{U} + \nabla_x^T \mathbf{U}\right)$$

$$\mathbf{R} = \frac{k}{2\varepsilon}\left(\nabla_x \mathbf{U} - \nabla_x^T \mathbf{U}\right)$$

T est composée de 10 tenseurs dans le cas tridimensionnel. Les 10 tenseurs proposés par Pope (1975) et repris par Ling et al. (2016b) sont :

$$\mathbf{T}^{(1)} = \mathbf{S}$$
$$\mathbf{T}^{(2)} = \mathbf{SR} - \mathbf{RS}$$
$$\mathbf{T}^{(3)} = \mathbf{S}^2 - \frac{1}{3}\mathbf{I} \cdot \mathrm{Tr}\left(\mathbf{S}^2\right)$$
$$\mathbf{T}^{(4)} = \mathbf{R}^2 - \frac{1}{3}\mathbf{I} \cdot \mathrm{Tr}\left(\mathbf{R}^2\right)$$
$$\mathbf{T}^{(5)} = \mathbf{RS}^2 - \mathbf{S}^2\mathbf{R}$$
$$\mathbf{T}^{(6)} = \mathbf{R}^2\mathbf{S} + \mathbf{S}^2\mathbf{R} - \frac{2}{3}\mathbf{I} \cdot \mathrm{Tr}\left(\mathbf{SR}^2\right)$$
$$\mathbf{T}^{(7)} = \mathbf{RSR}^2 - \mathbf{R}^2\mathbf{SR}$$
$$\mathbf{T}^{(8)} = \mathbf{SRS}^2 - \mathbf{S}^2\mathbf{RS}$$
$$\mathbf{T}^{(9)} = \mathbf{R}^2\mathbf{S}^2 + \mathbf{S}^2\mathbf{R}^2 - \frac{2}{3}\mathbf{I} \cdot \mathrm{Tr}\left(\mathbf{S}^2\mathbf{R}^2\right)$$
$$\mathbf{T}^{(10)} = \mathbf{RS}^2\mathbf{R}^2 - \mathbf{R}^2\mathbf{S}^2\mathbf{R}$$

Si T peut être calculé en cours de processus, le calcul de G reste encore obscur et justement les coefficients par rapport aux invariants sont définis par le training du NN utilisé ici avec 8 HL [1], 30 nœuds chacune et $learning\_rate = 2.5 \times 10^{-6}$.
Comme on peut le voir Fig.(2), les invariants sont injectés en entrée du réseau, suivent ensuite un processus classique propre au Feed-Forward NN. La dernière HL représentera les combinaisons linéaires de coefficients et de la base $\mathbf{\Omega}$.
L'output s'écrira alors :

$$b_{ij} = \sum_{n=1}^{10} g^{(n)}(\lambda_1, .., \lambda_5) T_{ij}^{(n)}$$

Il semblerait que ce réseau soit fait pour déduire l'anisotropie après calcul du modèle RANS. La prédiction à chaque itération fait partie d'un projet qu'ils ont évoqué dans leur article. Les auteurs proposent de comparer les résultats de leurs méthodes avec les DNS de plusieurs cas et avec d'autres méthodes de reconstruction de $\boldsymbol{b}$.
Le calcul de l'erreur sur l'anisotropie est donné par la formule :

$$\mathrm{RMSE} = \sqrt{\frac{1}{6N_{data}} \sum_{m=1}^{N_{data}} \sum_{i=1}^{3} \sum_{j=1}^{3} \left(b_{ij,m,\mathrm{predicted}} - b_{ij,m,\mathrm{DNS}}\right)^2}$$

On rapporte les erreurs calculés pour MLP et TBNN dans les cas d'un écoulement en conduite et d'un écoulement au dessus d'une surface présentant une cavité :

Modèle	RMSE on Duct Flow	RMSE on Wavy flow
MLP	0. 31	0.09
TBNN	0.14	0.08

TABLE 3 – Comparaisons des erreurs par rapport à la DNS.

Ils ont également mis en exergue le fait que les erreurs ne provenaient pas seulement de la modélisation du terme anisotrope dans l'hypothèse de Boussinesq ; mais également des équations de transports et des différentes approximations mises aux points dans le cadre de l'obtention de ces équations.
Un fait soulignable : le training set n'avait de cas semblable à celui du *wavy wall* et malgré tout le TBNN a réussi à fournir une amélioration par rapport aux modèles existants.
Ceci prouve que le TBNN est un outil adapté pour la tâche engagée par les auteurs.

1. La dernière HL en aura 10.

Wu et al. (2018)

Dans cette article, l'auteur cherche à reconstruire le tenseur d'anisotropie obtenu à partir d'une simulation RANS, à l'aide du Machine learning. On détaille les différentes méthodes et notions mentionnées. On explicitera les différentes régressions et l'expression des différents termes.
En se base sur Pope (1975) , il écrit également le tenseur déviatorique $\mathbf{b}$ comme :

$$\mathbf{b} = \sum_{n=1}^{10} G^{(n)} \mathcal{T}^{(n)}$$

De plus, on lie le Tenseur de Reynolds τ avec $\mathbf{b}$ par la relation

$$\tau = \mathbf{b} + \mathrm{Tr}\,(\tau)$$

On décompose ensuite le tenseur déviatorique en une partie linéaire et une partie non linéaire :

$$\mathbf{b} = \nu_t^L \mathbf{S} + \mathbf{b}^\perp$$

Ce qui nous permet de réécrire

$$\begin{aligned} \tau &= \nu_t^L \mathbf{S} + \mathbf{b}^\perp + \mathrm{Tr}\,(\tau) \\ &= \nu_t^L \mathbf{S} + \mathrm{Tr}\,(\tau) + \mathbf{b}^\perp \\ &= \tau^L + \mathbf{b}^\perp \end{aligned}$$

On exprime à présent ν_t^L, la viscosité turbulente opitmale, en minimisant l'écart entre $\mathbf{b}$ et sa partie linéaire :

$$\nu_t^L = \underset{\nu_t}{\mathrm{argmin}}\, ||\mathbf{b} - \nu_t \mathbf{S}||$$

Avec $||\cdot||$ la norme au sens de Froebenius : $||\mathbf{S}|| = \sqrt{S_{ij} S_{ij}}$. La minimisation précédente revient en fait à calculer

$$\nu_t^L = 2\, \frac{\mathbf{b} : \mathbf{S}}{||\mathbf{S}||\, ||\mathbf{S}||}$$

Où $\mathbf{b} : \mathbf{S} = b_{ij} S_{ij}$. On rappelle enfin l'équation dite de RANS :

$$\frac{D\overline{U}}{Dt} + \nabla \overline{P} - \nu \nabla^2 \overline{U} = \nabla \cdot \tau$$

Toutes les grandeurs ayant été introduites, on construit les features $\mathbf{Q}$.

Features q

Ling et al. (2016b) en suivant Pope (1975) écrivent τ comme une fonctionnelle $\tau\,(\mathbf{S}, \mathbf{\Omega})$ du tenseur de déformations $\mathbf{S}$ et du tenseur de rotations Ω, considérant que les effets des gradients de pressions n'agissaient pas sur ce tenseur.
De plus, ils ont supposé une turbulence à l'équilibre avec $\mathcal{P} = \varepsilon$. Dans ce cas, on pourrait simplement écrire $\tau \sim \nabla \overline{U}\,(\mathbf{x})$, négligeant ainsi les effets de convection et de diffusion de la turbulence (le non-équilibre de la production et de la dissipation finalement), rendant incomplète la correction cherchée.
Il propose une écriture de cette fonctionnelle prenant en compte les grandeurs mentionnées au travers leur gradient :

$$\tau = \mathcal{F}\,(\mathbf{S}, \mathbf{\Omega}, \nabla p, \nabla k)$$

Avec $\mathcal{F}$ une fonction invariante par rotation, pour chacun de ses arguments.
L'invariance rotationnelle peut être assurée en choisissant comme entrées la base intègre minimale de l'ensemble $\hat{\mathcal{Q}} = \{\mathbf{S}, \mathbf{\Omega}, \nabla p, \nabla k\}$ le chapeau signifiant que les entrées sont adimensionnalisées (on y reviendra), et les invariants de τ en sortie.
La base intègre contient ici 47 éléments listées dans l'annexe B de l'article. L'invariance Galiléenne ne se limitant pas à l'invariance par rotation. Les invariants choisis vérifient l'invariance *Galiléenne* exceptés ceux qui comprennent le ∇p adimensionnalisés par $\rho \left| \overline{U} \cdot \nabla \overline{U} \right|$.
Trois autres caractéristiques ont été sélectionnés pour leur sens physique et pour aider la décision de l'IA. Ces trois features sont issus d'un article antérieur [2] et sont :

q_1 : indique la distance entre le point en question et le mur. Sert à déterminer si des effets visqueux sont à prendre en compte ou non.

$$q_1 = \min\left(\frac{\sqrt{k}\, d}{50\nu}, 2\right), \text{ Déjà normalisé}$$

q_2 : fournit des informations sur l'échelle spatiale de la turbulence. Cette l'intensité turbulente k. On le normalise par

$$\frac{1}{2}\overline{U_i}\ \overline{U_i}$$

q_3 : fournit des informations sur l'échelle temporelle de la turbulence. C'est le rapport de l'échelle spatiale de la turbulence sur l'échelle temporelle moyenne du taux de féformation.

$$q_3 = \frac{k}{\varepsilon}, \qquad \text{Normalisé par } \frac{1}{||\mathbf{S}||}$$

Output du ML

On cherche ici à faire apprendre à la machine l'écart entre les résultat de la simulation RANS en cours par rapport à une véritable DNS. L'apprentissage se fait alors en indiquant que pour une simulation RANS donnée, dont sont extraites les entrées discutées plus haut, et dont nous disposons la DNS, les sorties doivent être les différences entre 6 grandeurs particulières qui sont le thème de cette section.

L'output de l'algorithme sera l'ensemble noté $\Delta\tau$. Cet ensemble contient des éléments invariants par rotation (comme les entrées) et veut reconstruire le champ anisotropique en renseignant sur l'écart prédit entre le tenseur de Reynolds de la simulation, et le tenseur de Reynolds de la DNS (si on l'avait lancée).
On écrit cet écart comme une fonctionnelle de six variables *physiquement interprétables* construites à partir de la décomposition suivante du tenseur de Reynolds :

$$\tau = 2k\left(\frac{1}{3}\mathbf{I} + \mathbf{b}\right) = 2k\left(\frac{1}{3}\mathbf{I} + \mathbf{V}\Lambda\mathbf{V}^T\right)$$

Avec $\mathbf{V} = [v_1, v_2, v_3]$ la matrice des vecteurs propres et $\mathbf{\Lambda} = \text{diag}\,[\lambda_1, \lambda_2, \lambda_3]$ la matrice ayant pour diagonale les valeurs propres de $\mathbf{b}$.

2. Les points importants de cet article en question étant similaires à celui là, nous aggrémentons les deux articles

Dans l'optique de gagner en sens physique, on va exprimer les valeurs propres dans une base de coordonnées barycentriques pour ensuite pouvoir quantifier le caractère de l'écoulement à partir d'un triangle d'état de l'écoulement. Cette base vérifie :

$$C_1 = \lambda_1 - \lambda_2$$
$$C_2 = 2(\lambda_2 - \lambda_3)$$
$$C_3 = 3\lambda_3 + 1$$

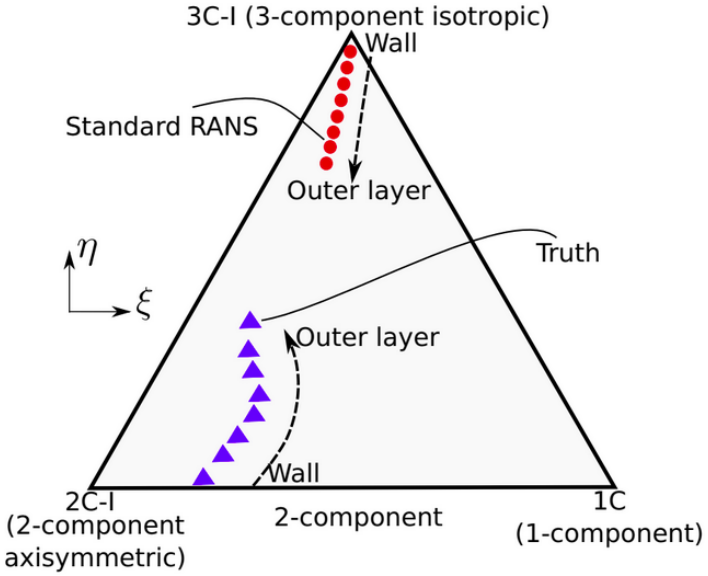
Tel que

$$C_1 + C_2 + C_3 = 1$$

Ensuite pour projeter l'écoulement (au travers ses valeurs propres) dans le triangle d'état, on pondère chacune de ces trois coordonnées par celles des sommets du triangle dont les coordonnées sont

$$[(\xi_1 c, \eta_1); (\xi_2, \eta_2); (\xi_3, \eta_3)] = [\xi_{1c}, \xi_{2c}, \xi_{3c}]$$

Le triangle d'état ou de *dimensionnalité de la turbulence* est représenté figure ci-dessous. On peut voir par exemple que le tenseur de Reynolds obtenu dans la résolution RANS standard est proche d'une turbulence isotropique.



On écrira alors

$$\vec{\xi} = \xi_{1c} C_1 + \xi_{2c} C_2 + \xi_{3c} C_3$$

Et on placera le cas étudié dans ce triangle grâce aux coordonnées ainsi calculées. L'avantage de ce triange encore une fois est le fait qu'il soit facile à comprendre et qu'il donne beaucoup d'informations sur la nature de la turbulence dans le cas considéré.
Les composantes d'une telle projection sont alors évaluées dans ce modèle et l'écart entre les composantes ξ et η sont sélectionnés pour être deux des output de l'IA :

$$\Delta\xi \text{ et } \Delta\eta$$

Une troisième quantité est la différence logarithmique entre l'énergie cinétique turbulente (TKE) de la simulation RANS et de la DNS. Dans l'apprentissage, on fournit les inputs de la simulation et on y fait correspondre L'exposant * faisant allusion à la TKE ciblée.

$$\Delta \log(k) = \log\left(\frac{k^*}{k^{\text{RANS}}}\right)$$

On construit ensuite les trois dernières sorties de l'algorithme en utilisant cette fois les vecteurs propres du tenseur anisotropique $\mathbf{V}$. On sait qu'il existe une base dans laquelle les vecteurs $\mathbf{V}^{\text{RANS}}$ peuvent être projetés pour retrouver les bons vecteurs propres notés $\mathbf{V}^*$. Cette base était intialement obtenu en utilisant les angles d'Euler, mais ici on propose l'utilisation d'unité quaternion dont la représentation est plus générale puisque indépendant du référentiel.
On consuit alors une base de vecteur $\mathbf{n} = [n_1, n_2, n_3]$ dont pour laquelle il existe un angle θ projetant $\mathbf{V}$ sur $\mathbf{V}^*$. On construit ensuite une unité quaternion $\mathbf{h}$ contractant la matrice de rotation liant les deux bases de vecteurs propres tel que :

$$\mathbf{h} = \left[\cos\frac{\theta}{2},\ n_1 \sin\frac{\theta}{2},\ n_2 \sin\frac{\theta}{2},\ n_3 \sin\frac{\theta}{2}\right]^T$$

L'existence de cet angle est le résultat du théorème d'Euler qui enstipule l'existence parant du principe que les ensembles de vecteurs propres sont orthonormés et qu'ils partagent une même origine (Wang et al. (2017)).
Pour plus de généralités, et puisque nous cherchons des sorties invariantes, on préfère prédire ou apprendre selon, les vecteurs h_1, h_2 et h_3 (soit les trois derniers vecteurs de $\mathbf{h}$) plutôt que n_1, n_2 et n_3 directement.
Finalement les six sorties choisies invariantes choisies sont :

$$(\Delta \log k, \Delta\xi, \Delta\eta, h_1, h_2, h_3)$$

Retour sur la normalisation des entrées

On résume les normalisation des entrées dans le tableau suivant, les 47 invariants de $\mathcal{F}(\mathbf{S}, \mathbf{\Omega}, \nabla p, \nabla k)$ sont disponibles dans l'annexe **B** de l'article Wu et al. (2018)

Entrée normalisée $\hat{\mathbf{q}}$	Desciption	Entrée de base	Facteur de Normalisation
$\hat{\mathbf{S}}$	Tenseur de déf.	$\mathbf{S}$	$\frac{\varepsilon}{k}$
$\hat{\mathbf{\Omega}}$	Tenseur de rot.	$\mathbf{\Omega}$	$\|\|\mathbf{\Omega}\|\|$
$\widehat{\nabla p}$	Grad. pression	∇p	$\rho\,\|\mathbf{U}\cdot\nabla\mathbf{U}\|$
$\widehat{k}$	Grad. TKE	k	$\frac{\varepsilon}{\sqrt{k}}$
q_1	Distance vav. paroi	$\min\left(\frac{\sqrt{k}\,d}{50\nu}, 2\right)$	Déjà adimensionné
q_2	Intensité turbulente (échelle spatiale)	k	$\frac{1}{2}U_iU_i$
$\widehat{q_3}$	$\frac{\text{Éechelle tempo. Turb}}{\text{Échelle tempo. def. moyenne}}$	$\frac{k}{\varepsilon}$	$\frac{1}{\|\|\mathbf{S}\|\|}$

Bibliographie

Duraisamy, K., Zhang, Z. J., and Singh, A. P. (2015). New approaches in turbulence and transition modeling using data-driven techniques. In *53rd AIAA Aerospace Sciences Meeting*, page 1284.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2) :273–324.

Kutz, J. N. (2017). Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814 :1–4.

Lefik, M. and Schrefler, B. (2003). Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Computer Methods in Applied Mechanics and Engineering*, 192(28) :3265 – 3283. Multiscale Computational Mechanics for Materials and Structures.

Ling, J., Jones, R., and Templeton, J. (2016a). Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318 :22–35.

Ling, J., Kurzawski, A., and Templeton, J. (2016b). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807 :155–166.

Ling, J. and Templeton, J. (2015). Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty. *Physics of Fluids*, 27(8) :085103.

Milano, M. and Koumoutsakos, P. (2002). Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1) :1–26.

Müller, A. C. and Guido, S. (2016). Introduction to machine learning with python : a guide for data scientists.

Parish, E. J. and Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305 :758–774.

Pope, S. (1975). A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2) :331–340.

Singh, A. P., Medida, S., and Duraisamy, K. (2017). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*.

Smith, G. (1965). On isotropic integrity bases. *Archive for rational mechanics and analysis*, 18(4) :282–292.

Tarantola, A. (2005). *Inverse problem theory and methods for model parameter estimation*. SIAM.

Tracey, B., Duraisamy, K., and Alonso, J. (2013). Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 259.

Tracey, B. D., Duraisamy, K., and Alonso, J. J. (2015). A machine learning strategy to assist turbulence model development. In *53rd AIAA Aerospace Sciences Meeting*, page 1287.

Wang, J.-X., Wu, J.-L., and Xiao, H. (2017). Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3) :034603.

Wu, J.-L., Xiao, H., and Paterson, E. (2018). Data-driven augmentation of turbulence models with physics-informed machine learning. *arXiv preprint arXiv :1801.02762*.

Zhang, Z. J. and Duraisamy, K. (2015). Machine learning methods for data-driven turbulence modeling. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2460.