Rapport de stage

pour l'obtention des grades de

Ingénieur en Sciences du Numérique de Toulouse INP-ENSEEIHT

et

Master 2 de l'Université de Toulouse

Spécialité : Informatique

par

Joël Roman KY

09 Septembre 2021

Analyse d'une base de données sur le vieillissement des piles à combustible

---

Stage réalisé chez Vitesco Technologies,
en collaboration avec ANITI[1], l'IMT[2] et le LAPLACE[3]

Encadrant école :
M. Serge GRATTON, Professeur, Toulouse INP-ENSEEIHT

Encadrants entreprise :
M. Ludovic LANDRY, Advanced Engineer, Vitesco Technologies
M. Lucian ALECU, Data Scientist, Continental Digital Services France
M. Fabrice GAMBOA, Professeur, ANITI-IMT
M. Christophe TURPIN, Directeur de Recherche, CNRS, LAPLACE

[1] Artificial and Natural Intelligence Toulouse Institute
[2] Institut de Mathématiques de Toulouse
[3] Laboratoire Plasma et Conversion d'Energie

Remerciements

Je tiens tout d'abord à remercier tous ceux qui ont contribué à rendre ce stage de fin d'études aussi enrichissant qu'intéressant. Je remercie tout d'abord M. Ludovic LANDRY pour l'opportunité et la confiance témoignée en me permettant de travailler sur les piles à combustible. En tant que maître de stage, ses discussions, ses conseils et la qualité de son encadrement m'ont été d'une très grande aide. Je remercie M. Lucian ALECU et M. Fabrice GAMBOA pour leurs conseils à la fois sur la partie mathématique et statistique de ces travaux lors des multiples points hebdomadaires qui ont jalonné ce stage. Je tiens à remercier de nouveau MM. Ludovic LANDRY et Lucian ALECU pour leurs relectures. Vos remarques constructives m'ont permis d'améliorer grandement la qualité de ce rapport.

Je remercie particulièrement M. Malik TOGNAN et M. Christophe TURPIN pour leur apport considérable et leurs conseils sur la base de données et sur la connaissance des piles à combustible sans lesquels une bonne partie des travaux auraient été difficiles à accomplir.

Je souhaite adresser des remerciements à mon responsable de département, M. Serge GRATTON d'avoir bien voulu être mon encadrant pédagogique sur ce stage, aux professeurs du parcours HPC et à toute l'équipe encadrante du département Sciences du Numérique de l'INP-ENSEEIHT pour la formation tout au long de ce cursus d'ingénieur.

J'aimerais aussi témoigner ma reconnaissance à toute l'équipe TI de Vitesco Technologies pour l'accueil et l'intégration au sein de l'équipe. Je remercie tous les différents collaborateurs de Vitesco, pour les différentes discussions et échanges d'idées sur la maintenance prédictive ou l'apprentissage automatique.

Un énorme merci à mon père, ma mère et mes frères, qui de loin n'ont cessé de me soutenir sur tous les plans afin que je puisse mener à bien mes études.

Je termine en remerciant particulièrement Axelle, pour sa relecture et ses conseils de rédaction qui m'ont été d'une grande aide.

Sommaire

Table des figures

Liste des tableaux

Liste des acronymes

AIC Aikake Information Criterion.

ANFIS Adaptive Neuro Fuzzy Inference System.

ANN Artificial Neural Network.

AWS Amazon Web Service.

BIC Bayesian Information Criterion.

CDSF Continental Digital Services France.

CNN Convolutional Neural Network.

CPU Central Processing Units.

DNN Deep Neural Network.

DWT Discrete Wavelet Transform.

EC2 Elastic Compute Cloud.

GPU Graphics Processing Units.

GRU Gated Recurrent Unit.

IA Intelligence Artificielle.

LOWESS Locally Weighted Scatterplot Smoothing.

LSSVM Least Square Support Vector Machine.

LSTM Long Short Term Memory.

LTU Linear Threshold Unit.

MEA Mind Evolutionary Algorithm.

NARNN Nonlinear Auto Regressive Neural Network.

PEMFC Proton Exchange Membrane Fuel Cell.

PF Particle Filter.

RFR Random Forests Regression.

RMSE Root Mean Square Error.

RNN Recurrent Neural Network.

RPF Regularized Particle Filter.

SSA-DGP Singular Spectrum Analysis-Deep Gaussian Processes.

SVM Support Vector Machine.

SVR Support Vector Regression.

TI Technology & Innovation.

UPF Unscented Particle Filter.

Chapitre 1

Introduction

1.1 Entreprise & Partenaires académiques

Ce stage intervient donc dans le cadre d'un projet de recherche sur les piles à combustible, impliquant Vitesco Technologies, ANITI et le laboratoire LAPLACE.

L'entreprise Vitesco Technologies (ancienne division Powertain de Continental) est un équipementier automobile pour une mobilité électrique durable et efficace. Ce stage de fin d'études a été réalisé au sein de l'équipe Technology et Innovation (TI) de Vitesco Technologies France. Cette équipe, qui regroupe une dizaine de chercheurs et d'ingénieurs, travaille sur la recherche et le développement de nouvelles technologies de propulsion et sur des solutions logicielles pour une optimisation intelligente et efficace de l'énergie. De nombreux échanges et discussions ont eu lieu avec des ingénieurs de Vitesco Technologies et d'autres équipes, notamment Lucian ALECU, Data Scientist chez Continental Digital Services France (CDSF).

ANITI (Artificial and Natural Intelligence Toulouse Institute) est un des quatre instituts interdisciplinaires d'intelligence artificielle (3IA) qui vise à développer une nouvelle génération d'intelligence artificielle hybride à travers une coopération forte entre chercheurs et industriels. Ce stage a impliqué la participation de la chaire de recherche *AI for physical models with geometric tools* de ANITI portée par le professeur Fabrice Gamboa.

Le LAPLACE (Laboratoire Plasma et Conversion d'Énergie) est une Unité Mixte de Recherche (UMR) du Centre National de la Recherche Scientifique (CNRS), de l'Institut National Polytechnique de Toulouse (INPT) et de l'Université Toulouse 3-Paul Sabatier (UPS). La collaboration avec le LAPLACE au cours de ce stage, est passée par la fourniture de la base de données de campagnes de vieillissement de piles à combustible et a impliqué la participation de Christophe TURPIN, chercheur CNRS au LAPLACE et responsable des activités hydrogène. Beaucoup d'échanges et réflexions ont été menés avec Malik TOGNAN, ingénieur R&D chez H2Pulse, entreprise spécialisée dans les technologies à base d'hydrogène.

1.2 Contexte du projet

La technologie "pile à combustible" est une technologie prometteuse pour la décarbonisation des transports terrestres. Le développement des piles à combustible a ainsi connu un grand intérêt afin de développer une nouvelle génération de moyens alternatifs de propulsion pour les véhicules. Cette technologie est de plus en plus adoptée dans les moyens de transport comme les bus et camions. Pour ce qui est des véhicules légers, la plupart des concepteurs automobiles sont encore au stade de recherche et développement, quoique le constructeur Toyota ait commercialisé son modèle *Mirai*. Cependant, malgré une telle avancée dans le domaine de l'hydrogène, son déploiement et son utilisation à plus grande échelle reste limitée. La technologie des piles à combustible reste confrontée à plusieurs verrous technologiques qui empêchent un déploiement industriel conséquent. Malgré sa grande efficacité énergétique et son émission faible, les piles à combustible possèdent une durée de vie limitée et un coût élevé. La production du H_2 suscite encore beaucoup de questions tant d'un point de vue écologique et aussi en termes d'efficacité. Les stations de recharge aussi sont très rares malgré la facilité de recharge des véhicules à hydrogène.

Dans le but de prolonger la durée de vie des piles, de nombreuses études ont été effectuées afin d'identifier les indicateurs de vieillissement des piles à combustible ainsi que de réaliser des pronostics sur la fin de vie de la pile ou sur le temps de vie restant de la pile. L'objectif des pronostics est de

pouvoir prédire les dégradations du fonctionnement de la pile afin d'opérer des opérations de maintenance prédictive et ainsi de prolonger la durée de vie.

Un des moyens de réaliser le pronostic du temps de vie de la pile est de modéliser le vieillissement de la pile à combustible en se basant sur les informations récupérées lors des campagnes de vieillissement, qui sont des études expérimentales sur la pile conduites en laboratoire. Les données récupérées pendant ces campagnes sont celles que l'on peut mesurer facilement à l'aide des capteurs. Il s'agit entre autres de la tension de la pile et des différentes cellules de la pile, l'intensité, la pression, les débits de charge, etc.

1.3 Problématique et Objectifs du stage

La problématique principale de ce stage est liée à la durée de vie des piles à combustible. Comment réaliser un pronostic de la fin de vie de la pile afin de réaliser des opérations de maintenance prédictive ? Pour réaliser ce pronostic, nous pouvons nous baser sur les indicateurs que l'on peut mesurer assez facilement comme la tension, l'intensité, la pression, les températures etc lors de campagnes expérimentales ciblées.

Il est assez difficile de réaliser un pronostic fiable sur le vieillissement des piles à combustible. Les différentes modélisations basées sur des observations expérimentales ou sur la connaissance physico-chimique des piles sont efficaces pour modéliser efficacement la dégradation, mais sont très coûteuses en termes de calcul. Une autre façon d'effectuer ce pronostic peut se faire à l'aide des méthodes statistiques et d'apprentissage automatique. Ces méthodes permettent de bien traiter les phénomènes de bruit ou la variabilité des expériences. Cependant, les méthodes d'apprentissage automatique (Intelligence Artificielle) sont confrontées aux problèmes de généralisabilité et d'explicabilité (*black box*). Comment résoudre les problèmes présentés par ces deux méthodes ?

Les objectifs de ce stage ainsi que les travaux réalisés se divisent comme suit :

- Réaliser un état de l'art des différents mécanismes de vieillissement des piles à combustible ainsi que des différentes méthodes de modélisations du vieillissement et de pronostic de fin de vie des piles à combustible.
- Étudier la base de données de vieillissement de piles à combustible afin d'y identifier les principaux facteurs de vieillissement de la pile.
- Utiliser des algorithmes d'intelligence artificielle (*machine learning, deep learning*) afin de modéliser le vieillissement de la pile et prédire les dégradations dans la pile.
- Se servir des connaissances physiques afin de rendre les modèles statistiques ou d'apprentissage plus interprétables et généralisables.

Chapitre 2

État de l'art

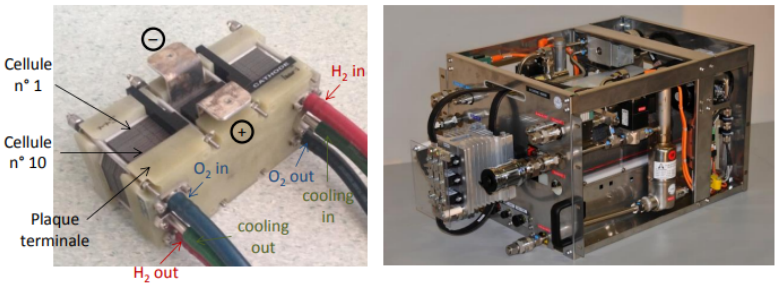
La pile à combustible est sujette à de nombreuses dégradations qui affectent son fonctionnement et contribuent à son vieillissement. Plusieurs méthodes ont été proposées afin de modéliser ce vieillissement. Ces méthodes se divisent en trois catégories : les méthodes physiques dites *model-based*, les méthodes basées sur des approches empiriques ou guidées par les données (encore appelées méthodes *data-driven*) et les méthodes hybrides (combinant les deux premières approches).

Nous présenterons dans ce chapitre la pile à combustible et son principe de fonctionnement. Par la suite, nous présenterons les différents mécanismes de dégradation de la pile à combustible, l'état de l'art des différentes méthodes de la littérature pour le vieillissement des piles à combustible ainsi qu'une critique de cet état de l'art.

2.1 Travaux précédents

2.1.1 Piles à combustible : histoire et fonctionnement

L'effet pile à combustible est découvert par l'Allemand Christian Schönbein en 1839. Le premier modèle de laboratoire de pile à combustible est réalisé par William R. Grove sur les trois années suivantes.



Source : [1]

FIGURE 2.1 – Pile à combustible

La pile à combustible est un moyen de convertir l'énergie chimique d'une oxydo-réduction en énergie électrique. Il existe six (06) types de piles à combustible :

- **PEMFC** : Proton Exchange Membrane Fuel Cell
- **DMFC** : Direct Methanol Fuel Cell
- **PAFC** : Phosphoric Acid Fuel Cell
- **AFC** : Alkaline Fuel Cell
- **SOFC** : Solid Oxyde Fuel Cell

- **MCFC** : Molten Carbonate Fuel Cell

Les piles les plus avancées pour une utilisation commerciale et celles auxquelles nous nous sommes intéressées dans ce projet sont les piles à membrane échangeuses de protons (PEMFC).

Le cœur des PEMFC est constitué d'une anode siège de l'oxydation et d'une cathode siège de la réduction séparée par un électrolyte constitué d'une membrane solide polymère qui fait l'échange des protons (H+).

Le cœur de la pile est le siège de deux réactions chimiques :

Anode : $$H_2 \rightarrow 2H^+ + 2e^-$$

Cathode : $$\frac{1}{2}O_2 + 2H^+ + 2e^- \rightarrow H_2O + Q$$

Equation globale : $$H_2 + \frac{1}{2}O_2 \rightarrow H_2O + Q$$

2.1.2 Mécanismes de dégradation de la pile à combustible

Nous présentons dans cette section les différents mécanismes de dégradations qui influent sur le fonctionnement de la pile. Elles se divisent en dégradations irréversibles et réversibles et affectent différents composants de la pile. Ces dégradations ont différentes causes et conséquences sur la pile à combustible et sont synthétisées par Wasterlain [2] dans le tableau ci-dessous :

Composants	Dégradations	Causes	Conséquences	Gravité
Plaque bipolaire	Déformations/ Fêlures	Couple de serrage	Fuites des réactifs	Irréversible
	Obstruction canaux	- Gel - Accumulation d'eau	- Accès gaz réactifs - Surpression locale	Réversible Réversible
		Corrosion	Fragilisation	Irréversible
Joint d'étanchéité	Perte d'étanchéité	- Exposition à H_2 - Milieu corrosif	- Dissolution du joint - ↓ Conductivité membrane - Mélange H_2/O_2	Irréversible
Couche de diffusion	Corrosion carbone	- Radicaux libres - Potentiel électrodes - Faible humidité relative	- ↑ Résistance électrique - ↓ - Conductivité membrane - Accumulation d'eau	Irréversible
	Dissolution PTFE			
	Rupture des fibres	- Couple de serrage	- ↑ Résistance électrique	Irréversible
Couche active	Corrosion du carbone	- Potentiel électrodes - Cyclage tension - Appauvrissement réactif	- ↓ Surface active - ↑ Résistance électrique	Irréversible
	- Dissolution - Agglomération du Pt		↓ Surface active	
	Empoisonnement	Réactifs impurs		Dépend des espèces
Membrane	Assèchement	Mauvaise gestion eau	- ↑ Résistance électrique - Fragilisation membrane	Réversible
	Empoisonnement	Réactifs impurs	↓ Conductivité membrane	Irréversible
	Perforations	- Mélange H_2/O_2 - Imprégnation catalyseur	Rupture de la membrane	

TABLE 2.1 – Mécanismes de dégradation de la pile à combustible

Comme le montre le tableau ci-dessus, ces dégradations sont nombreuses, affectent différemment de nombreux composants, ont des causes diverses et s'inter-influencent mutuellement. De ce fait, l'identification et la modélisation du vieillissement pose de nombreuses difficultés.

2.1.3 Méthodes de modélisation physique

Les méthodes de modélisation physique sont construites à partir des lois qui gouvernent les phénomènes physiques intrinsèques de la pile ou par des procédures expérimentales permettant de calibrer les

paramètres de ces modèles. Comme mentionné par Tognan et al. [3], ces modélisations se décomposent en modélisations "mécanistiques" qui reposent sur une forte connaissance des phénomènes physico-chimiques de la pile ; en modélisations semi-empiriques qui intègrent l'évolution temporelle des paramètres du modèle dans un modèle physique et en modélisations empiriques qui utilisent des méthodes de filtrage associées à un modèle de vieillissement.

Tognan et al. [3, 1] modélise la dégradation en fonction d'un taux de dégradation qui dépend du temps et du temps du dernier démarrage. Cette méthode permet de découpler les pertes réversibles et irréversibles. Ce modèle est associé aussi à une loi empirique qui permet de suivre l'étanchéité interne de la pile durant une campagne, qui est aussi un critère déterminant de la fin de vie de la pile.

Jouin et al. [4] se base sur la modélisation de la pile à combustible à partir de l'équation de polarisation. Cette équation modélise les pertes qui impactent la tension qui peuvent être divisées en 4 catégories (activation, ohmique, concentration et crossover). L'idée de Jouin et al. [4] est de modéliser l'évolution temporelle de chacune de ses pertes. Les paramètres de ce modèle sont estimés par marche aléatoire. Cette modélisation physique est couplée à un filtre particulaire afin d'effectuer un pronostic sur la durée de vie de la pile et aussi corriger l'estimation des paramètres du modèle. La validation de ce modèle montre qu'il arrive à suivre la tendance de vieillissement de la pile, mais présente une incertitude élevée (15W) dans le 25-ième et 75-ième centile.

Jouin et al. [5] proposa une autre modélisation physique empirique en utilisant le filtre particulaire pour estimer le temps de vie restant à une pile, mais en se basant sur un modèle de dégradation de la tension de la pile basé sur des distributions de probabilité. Le filtre particulaire, qui est une méthode de Monte-Carlo basée sur le théorème de Bayes, permet de répondre aux conditions imposées par Jouin et al. [5] dans leurs modélisations. Le modèle de dégradation, implanté en utilisant un modèle d'états empirique qui peut être linéaire, log-linéaire ou exponentiel, est validé sur deux piles différentes. Cette validation montre que les performances du modèle sont meilleures lorsque le modèle d'états est logarithmique, mais que les dispersions sont plus étendues.

Zhang et al. [6] propose une méthode d'ensemble basée sur des filtres particulaires qui consiste à combiner deux sources d'informations : la première étant l'information de dégradation fournie par la tension de la pile, mais aussi une information empirique indicatrice de la dégradation de la pile γ. Cet indicateur γ permet de compenser le manque d'information de la tension de la pile sur la dégradation interne qui affecte la résistance interne et la charge à vide. En effet, lors du déroulement d'une campagne de vieillissement de la pile, la tension ne traduit que les symptômes de dégradation de la pile, mais pas les causes de cette dégradation. Elle est alors insuffisante pour suivre le vieillissement de la pile. Zhang et al. combine les pronostics de fin de vie de ces deux méthodes par agrégation locale en assignant des poids proportionnels aux performances de chaque filtre particulaire sur les données d'entraînement.

2.1.4 Méthodes de modélisation data-driven

Les méthodes data-driven se basent sur des techniques d'apprentissage statistiques sur de grandes quantités de données récupérées sur des campagnes de vieillissement de piles à combustible.

Vichard et al. [7] utilise les *Echo State Network* qui est un type de réseau de neurones récurrents (RNN) efficace basé sur un concept de réservoir de neurones. Le réservoir est un système non-linéaire de RNNs à dynamique fixe, qui associe des signaux d'entrées à des espaces de grande dimension, ce qui permet de réduire la puissance de calcul. L'importance de la température ambiante, qui n'est pas toujours considéré, a été l'une des principales problématiques de cette étude. Des études expérimentales sous température ambiante variable ont donc été conduites afin d'avoir des données pour entraîner le modèle. Ce modèle proposé par Vichard et al. modélise ainsi le vieillissement à partir de la tension, la température ambiante ainsi que le mode opératoire d'une pile sous conditions de fonctionnement stationnaires. Les performances de cette méthode augmentent en fonction du ratio données d'entraînement/données de prédiction. Ainsi pour un ratio 60/40, on obtient une *RMSE* (Root Mean Square Error) de 0.098.

D'autres méthodes à base de LSTM (Long Short-Term Memory) et de RNN (Recurrent Neural Network) ont été employées par Liu et al. [8]. Les LSTM sont connues pour leur habilité à traiter les longues données temporelles et à compenser les défauts des RNN. Afin de réduire la complexité de calcul, les données sont reconstruites et lissées par la méthode LOWESS (Locally Weighted Scatterplot Smoothing) afin de supprimer les points aberrants et bruités. La méthode proposée permet de réduire le coût de calcul de la méthode et d'atteindre une RMSE de 0.003. Zuo et al. [9] a aussi utilisé des LSTM et des GRU (Gated Recurrent Unit) combiné à des mécanismes d'attention. Les GRU sont une simplification des LSTM mais, tout aussi efficaces pour le traitement des longues séquences de données. Les mécanismes

d'attention, qui ont prouvé leurs efficacités dans les méthodes de vision par ordinateur, permettent de concentrer "l'attention" du réseau de neurones sur certaines caractéristiques du signal en leur accordant plus de poids. Ils permettent ainsi d'améliorer la prédiction. La comparaison entre le modèle proposé et des GRU combinés à des mécanismes d'attention montrent que les mécanismes d'attention améliorent les performances des LSTM et des GRU. On obtient une RMSE de 0.008.

Liu et al. [10] a utilisé une méthode basée sur la technique des auto-encodeurs couplés à des réseaux de neurones profonds (DNN). Les données issues des campagnes expérimentales sont en quantités importantes et surtout très bruitées. L'idée de Liu et al. est de lisser les données brutes de tension d'une pile collectées dans une campagne par la méthode *Gaussian weighted average filter* et d'extraire les caractéristiques pertinentes à l'aide du réseau *Sparse Auto-Encoder* préalablement entraînés pour reconstruire une représentation de faible dimension des données de vieillissement de la pile. Les auto-encodeurs creux permettent d'obtenir une bonne représentation des données et de surmonter les inconvénients de l'initialisation aléatoire dans le réseau de neurones profonds. La prédiction est ensuite effectuée grâce aux réseaux DNNs et la validation donne une erreur absolue de 0.2035 sur des données de 1020 h avec les 500 premières heures utilisées comme données d'entraînement. Cette performance de prédiction varie avec la taille des données d'entraînement.

La méthode proposée par Chen et al. [11] consiste à améliorer la convergence des algorithmes à base de réseaux de neurones artificiels (ANN) à l'aide d'algorithmes évolutionnaires (optimisation par essaim de particules, algorithmes génétiques, MEA). L'algorithme de rétropropagation (*backpropagation*) a en effet une capacité de recherche, dans l'espace des paramètres, faible et est sensible aux initialisations. Les algorithmes évolutionnaires, contrairement, sont plus efficaces dans ce domaine et sont utilisés par Chen et al. pour initialiser les poids des réseaux de neurones. De plus, cette méthode, contrairement aux autres méthodes, tente de prédire le vieillissement des piles sous conditions opératoires réelles, qui sont un facteur influençant significativement la dégradation de la pile. Les données sont d'abord lissées par la méthode LOWESS. La validation montre que l'algorithme de backpropagation couplée à la méthode MEA (Mind Evolutionary Algorithm) obtient le taux d'erreur le plus bas, soit une erreur absolue en pourcentage de 0.218% lors de l'estimation de la fin de vie.

Xie et al. [12] utilise un processus de dégradation gaussien précédé d'une analyse singulière spectrale pour dé-bruiter les données afin d'améliorer les performances. Cette méthode a pour but de proposer une méthode plus généralisable et plus précise que les méthodes data-driven les plus souvent utilisées (statistiques, machine-learning, réseau de neurones). Les méthodes de machine learning ont des limitations avec des données brutes bruitées et sujettes au sur-apprentissage ; les méthodes statistiques sont sensibles aux bruits et aux pics présents dans les données. De plus, la plupart de ces méthodes ne fournissent pas de degré de confiance en leurs prédictions. La méthode proposée (SSA-DGP) supprime les bruits et les pics grâce à l'analyse spectrale singulière (SSA). L'analyse spectrale se base sur la décomposition en valeurs singulières (SVD)) pour décomposer et reconstruire les données de sorte à supprimer les pics et le bruit. La méthode DGP (*Deep Gaussian Process*) se base sur plusieurs modèles de processus gaussiens à variables latentes (GPLVM), couplée à des fonctions de noyau pour améliorer la performance. Ce modèle est ainsi plus efficace pour gérer les non-linéarités dans le processus de dégradation. Les moyennes des prédictions de ce modèle, ainsi que les matrices de covariances obtenues permettent l'élaboration d'un intervalle de confiance sur la prédiction. Une comparaison de cette méthode a été effectuée avec différents modèles (LSTM, Relevance Vector Machine, Gaussian Process State Model). Cette comparaison montre que la méthode SSA-DGP suit bien la dégradation comme les LSTM, contrairement aux autres méthodes, et obtient la meilleure performance, soit une RMSE de 0.0057.

2.1.5 Méthodes de modélisation hybrides

Les méthodes hybrides résultent d'une combinaison des deux précédentes méthodes. Les méthodes physiques permettent de modéliser la dégradation, mais comportent un coût de calcul élevé et utilisent des hypothèses très simplificatrices. Les méthodes data-driven, elles permettent un meilleur traitement des données bruitées et non-linéaires mais sont considérées comme des boîtes noires. Il est donc nécessaire d'utiliser à la fois ces deux méthodes par une approche hybride qui consiste à apporter la connaissance physico-chimique aux modèles data-driven.

Cheng et al. [13] propose une méthode combinant un modèle data-driven, *Least Square Support Vector Machine* (LSSVM) à une méthode basée sur le filtre particulaire régularisé (RPF). Le filtre parti-

culaire et la méthode SVM (Support Vector Machine) sont souvent employées dans la littérature pour la modélisation de la dégradation de la pile. Le filtre particulaire souffre du problème de dégénérescence des particules ainsi que de la perte de diversité, ce qui restreint son application. La méthode SVM est connue pour être plus généralisable que les réseaux de neurones, mais possède de faibles performances quand la quantité de données devient très importantes. La méthode LSSVM proposée par Cheng et al. améliore la complexité de la méthode SVM en convertissant le problème d'optimisation en équations linéaires ce qui permet en plus d'améliorer la précision de convergence. Le filtre particulaire régularisé est une amélioration du filtre particulaire classique qui permet de s'affranchir du problème de dégénérescence des particules ainsi que la perte de diversité issue du sur-échantillonnage. Le couplage de ces deux méthodes permet ainsi d'obtenir une méthode hybride, généralisable qui fournit en plus une caractérisation de l'incertitude de prédiction. Les données sont filtrées par une méthode de spline cubique pour supprimer les bruits et réduire le temps de calcul. Les prédictions obtenues par la méthode LSSVM sont utilisées comme entrées pour le modèle RPF, ce qui permet d'obtenir une quantification de l'incertitude du pronostic de fin de vie de la pile. Les résultats obtenus avec cette méthode sont une RMSE de 0.0072. Ces résultats sont 64 fois meilleurs que ceux obtenus avec la méthode du filtre particulaire.

La méthode proposée par Chen et al. [14] est basée sur un modèle de dégradation gaussien couplé à un filtre UPF (*Unscented Particle Filter*). Les travaux dans la littérature utilisent pour la plupart le même indicateur de l'état de la pile sans chercher à l'enrichir d'informations ou à l'améliorer. Chen et al. propose un nouvel indicateur résultant de la fusion de trois (03) indicateurs de santé (la tension, la puissance et la résistance). La méthode proposée utilise le calcul de la distance géodésique, qui décrit efficacement la structure spatiale des données, pour estimer l'état de santé actuel de la pile relativement à l'état de référence. Le temps de vie restant de la pile est estimé par le filtre UPF couplé à un modèle gaussien d'ordre 2 qui modélise l'évolution de l'état de santé. Ce couplage permet d'améliorer les faiblesses de la modélisation de la dégradation de la pile. Les données ont été prétraitées avec la méthode LOWESS afin de supprimer les pics et les bruits des données. Les résultats de la méthode sur l'estimation de la durée de vie restante sont de 99,51% quand la durée d'entraînement est de 500 h sur un dataset de 1200 h.

Zhou et al. [15] propose une méthode hybride pour prédire la dégradation en utilisant une technique de fenêtre glissante en combinant un filtre particulaire et un modèle de réseaux de neurones non-linéaire auto-régressif (NARNN). L'approche adoptée permet de fusionner les prédictions des méthodes physiques et data-driven, afin d'avoir une prédiction efficace sur la tendance de dégradation à long terme tout en conservant les caractéristiques de dégradation à court terme. Ce modèle arrive à capturer la tendance de dégradation de la pile grâce au filtre particulaire et à décrire efficacement les caractéristiques non-linéaires locales de la dégradation avec le réseau de neurones non-linéaire auto-régressif. La fenêtre glissante permet de subdiviser les données en mini-blocs regroupés par blocs de 3 (entraînement, validation, prédiction). Cette méthode permet de rendre le modèle robuste sur tout le dataset et permet aussi des prédictions incrémentales. Les prédictions des deux méthodes sont combinées proportionnellement à l'inverse des erreurs de chaque modèle sur les données de validation. L'évaluation des performances de ce modèle est réalisée sur différents datasets et montre que le modèle hybride est plus performant par rapport aux modèles data-driven et modèles physiques, ainsi que par rapport à la méthode ANFIS[16].

Xie et al. [17], comme dans les autres méthodes, combine un filtre particulaire à une architecture LSTM pour la prédiction du vieillissement. L'idée de Xie et al. est de bénéficier des bonnes performances du filtre particulaire sur les systèmes non-linéaires avec un bruit non gaussien et aussi des bonnes performances des LSTM sur les longues séries temporelles. Le filtre particulaire et l'architecture LSTM sont entraînés séparément et lors de la prédiction, les prédictions de ces deux méthodes sont fusionnées pour obtenir les résultats finaux. L'évaluation des performances de ce modèle, en le comparant au filtre particulaire, montre que les prédictions du modèle sont plus proches des valeurs réelles malgré la grande incertitude de la méthode.

Wang et al. [18], contrairement aux autres méthodes hybrides, combine plusieurs méthodes data-driven : les LSTM, une régression SVR (Support Vector Regression) et une régression à base de forêts aléatoires (RFR). L'objectif de cette méthode proposée est d'agréger, par une méthode d'ensemble, les prédictions de différents modèles pour améliorer les prédictions. Pour cette étude, les données utilisées proviennent de deux piles différentes sous des conditions opératoires stationnaires et dynamiques. Les paramètres sont estimés à l'aide de métaheuristiques comme l'évolution différentielle. Un des apports de la méthode hybride de Wang et al. a été d'utiliser une approche de Monte Carlo (Monte Carlo Dropout)

pour obtenir un intervalle de confiance pour les prédictions. Les prédictions du temps de vie restant à la pile sont effectuées à l'aide d'une technique de fenêtre glissante. Les résultats avec différents datasets montrent une erreur relative de moins de 2% en fonction de la taille de l'ensemble d'apprentissage.

2.1.6 Synthèse des méthodes de modélisation

Le tableau ci-dessous synthétise les différentes méthodes de la littérature pour le vieillissement des piles à combustibles.

Types	Noms	Principe	Références
Méthodes physiques	Taux de dégradation	Taux de dégradation de la tension en fonction des arrêts	[1, 3]
	Équation de polarisation	Modéliser la tension de la pile en fonction des pertes	[4]
	Filtres particulaires	Méthode de Monte-Carlo empirique basée sur le théorème de Bayes	[5]
Méthodes data-driven	Echo State Network (ESN)	Réservoirs de réseaux de neurones récurrents	[7]
	LSTM-RNN	Réseaux de neurones récurrents gérant les dépendances temporelles sur le long-terme	[8, 9]
	DNN & Auto Encoders	Réseaux de neurones couplés à des auto-encoders pour extraire les caractéristiques du vieillissement de la pile	[10]
	Algorithmes évolutionnaires	Amélioration de la convergence des réseaux de neurones par des métaheuristiques	[11]
	SSA-DGP	Processus de dégradation gaussien couplé à une analyse spectrale	[12]
Méthodes hybrides	LSSVM-RPF	Filtre particulaire régularisé couplé à des machines à vecteurs de supports	[13]
	Modèle gaussien-UPF	Modèle de dégradation gaussien couplé à un filtre particulaire	[14]
	NARNN-PF	Réseaux de neurones non-linéaires couplés à un filtre particulaire	[15]
	LSTM-RPF	Réseaux de neurones récurrents couplés à un filtre particulaire	[17]
	LSTM-SVR-RPF	Méthode d'ensemble basée sur une combinaison de réseaux récurrents, forêts aléatoires et de machines à vecteurs de supports	[18]

TABLE 2.2 – Synthèse des méthodes de l'état de l'art

2.2 Synthèse critique de l'état de l'art

Les approches physiques présentent l'intérêt de modéliser les aspects physico-chimiques en se basant sur des lois physiques et les mécanismes de dégradation. Le principal avantage de ces méthodes est qu'elles ne nécessitent pas beaucoup de données. Les phénomènes complexes liés au fonctionnement des

piles ne sont pas tous parfaitement connus et modélisés, ce qui limite les performances de ces méthodes. Aussi les méthodes physiques sont souvent coûteuses en calcul en raison de la complexité de certaines modélisations analytiques. Les approches utilisées par exemple par Jouin et al. [5] ont nécessité de modéliser linéairement le phénomène de dégradation de la pile. Cette hypothèse simplificatrice ne permet pas de traduire tout le phénomène de dégradation des tensions des piles à combustible qui est principalement non-linéaire et surtout de tirer profit du filtre particulaire.

Les approches data-driven que nous avons énoncées plus haut sont satisfaisantes pour la modélisation du vieillissement et possèdent toutes des erreurs de prédictions faibles. Il est assez délicat d'effectuer une comparaison entre ces méthodes car les conditions utilisées ne sont pas toujours les mêmes. Elles présentent l'avantage de mieux modéliser les non-linéarités que les méthodes physiques. Ces approches arrivent à effectuer de bonnes prédictions. Étant donné que tous les phénomènes et les systèmes liés aux fonctionnements de la pile ne sont pas maîtrisés, ces méthodes permettent d'apprendre le comportement de la pile en s'affranchissant de ces verrous et en se basant sur des données issues des campagnes. Les méthodes présentes dans la littérature se basent essentiellement sur des données issues de campagnes de vieillissement stationnaires. L'impact des arrêts/démarrages au cours des campagnes est rarement pris en compte ainsi que l'impact des différentes conditions opératoires pris en compte par Chen et al. [11].

Le plus gros inconvénient des méthodes data-driven reste la non-explicabilité des modèles fournis. Les méthodes data-driven, principalement celles à bases de réseaux de neurones, sont vues comme des boîtes noires (*black-box*), en raison de la difficulté d'expliquer les résultats issus de ces modèles. De plus, il est compliqué de savoir à quel point ces méthodes sont capables de se généraliser en dehors des données rencontrées, et aussi comment garantir proprement cette généralisabilité dans les modèles. Ces principaux inconvénients constituent des freins importants qui doivent faire l'objet de futurs travaux.

Afin de répondre aux problèmes liés à ces deux modélisations, les méthodes hybrides permettent de combiner les avantages liés aux deux modélisations. Celles que nous avons présentées plus haut permettent d'avoir de meilleures performances que celles obtenues avec les méthodes data-driven pures. Cependant, les principales méthodes hybrides de la littérature n'utilisent pas vraiment au mieux les avantages des méthodes physiques. Toutes les méthodes hybrides de la littérature utilisent des méthodes à base de filtres particulaires. L'avantage du filtre particulaire, qui est une méthode bayésienne, est de pouvoir assimiler les propriétés non-linéaires et de tenir compte de distributions non gaussiennes. Cependant, dans les modélisations utilisées pour le modèle d'état, censé représenter la tendance de dégradation de la pile, la modélisation linéaire est la plus souvent retenue pour des raisons de simplicité, les non-linéarités supposées captées par le modèle data-driven. L'hypothèse linéaire ne traduit pas correctement la tendance de dégradation de la tension de la pile et ne permet pas d'intégrer toute la connaissance physico-chimique traduite par les méthodes physiques.

Un autre inconvénient de la plupart des méthodes énoncées ici, est que les conditions étudiées sont celles stationnaires. Les conditions opératoires de fonctionnement de la pile à combustible dans un véhicule ne sont pas toujours étudiées. L'indicateur de santé le plus utilisé dans la littérature est la tension. L'utilisation d'autres indicateurs comme l'impédance et la résistance reste minoritaire. D'autres critères pouvant impacter la durée de vie de la pile, comme l'impact des arrêts démarrages ou des profils de route sont très rarement étudiées. Les données brutes issues des campagnes d'endurance sont généralement lissées avant d'appliquer les algorithmes d'apprentissage machine qui sont très sensibles aux points aberrants. Ce lissage conduit à négliger l'impact des arrêts démarrages qui conduisent à un recouvrement des capacités de la pile (dégradations réversibles). L'étude de ces facteurs permettrait d'avoir une meilleure modélisation de la durée de vie de la pile.

Une solution envisageable pour un meilleur pronostique de la fin de vie des piles à combustible serait l'utilisation de l'apprentissage automatique assisté par la physique, proposé par Karpatne et al. [19]. Cette approche utilisée dans la modélisation de plusieurs domaines scientifiques comme la modélisation des turbulences, la chimie quantique, les sciences du climat ou l'hydrologie permet d'intégrer la connaissance physique dans un modèle de data science sous différentes formes : contraintes de pénalisation en cas de solution non consistante au sens physique, augmentation des caractéristiques en utilisant des caractéristiques issues du modèle physique, insertion des équations physiques dans des réseaux de neurones... On obtient ainsi des modèles prédictifs robustes d'un point de vue physique, généralisables et interprétables. Cette approche est aussi appliquée par Bikmukhametov et al. [20], dans laquelle des techniques d'augmentation de l'espace des caractéristiques, par des caractéristiques "physiques", permettent d'obtenir des modèles plus performants et plus interprétables. Il apparaît donc intéressant d'étudier cette

piste pour le cas des piles à combustible.

2.3 Conclusion

L'utilisation à grande échelle des piles à combustible dans les véhicules demeure tributaire de sa durée de vie. Les différentes méthodes proposées dans la littérature pour modéliser le vieillissement d'une pile à combustible et estimer sa fin de vie, se divisent en trois principales approches : méthodes physiques, data-driven et hybrides. Les principales méthodes physiques se basent sur des modélisations des phénomènes physico-chimiques de la pile ou sur des modélisations empiriques couplées à des techniques de filtres particulaires. Les réseaux de neurones, notamment les réseaux récurrents, ainsi que des méthodes statistiques (machine learning) sont les méthodes data-driven qui fournissent les meilleures modélisations et estimations. Chacune de ces deux approches présentent des inconvénients, soit sur la prise en compte des différents phénomènes qui impactent le vieillissement de la pile, l'intégration des non-linéarités dans la modélisation, le besoin d'une grande quantité de données, l'interprétabilité du modèle etc.

Il ressort ainsi qu'une approche dite hybride pourrait permettre de combiner les avantages de chacune des deux méthodes. Seulement, malgré les performances des approches hybrides proposées dans la littérature, il est à noter que le caractère hybride de ces méthodes reste discutable, tant la présence d'une modélisation physique est très souvent simplifiée ou présente de façon empirique. On se retrouve de ce fait à négliger des facteurs qui influencent la dégradation de la pile comme les arrêts/démarrages. Les données d'endurance utilisées lors des expérimentations sont très souvent étudiées dans des conditions stationnaires. Ce point rend difficile la généralisation de ces approches à des conditions opératoires dynamiques, quoique les conditions opératoires dynamiques peuvent être interprétées comme une combinaison de plusieurs conditions stationnaires. Il apparaît quand même intéressant de disposer de données dynamiques afin d'avoir une meilleure modélisation. L'approche du machine learning assisté par la physique est une solution assez prometteuse pour réussir à proposer des approches hybrides et efficaces, en intégrant la connaissance physique de différentes manières dans des modèles de machine learning. Ces considérations ont guidé les travaux de ce stage qui seront présentés dans la suite.

Chapitre 3

Analyse de la base de données

3.1 Présentation de la base de données

Les bases de données de vieillissement que nous avons à disposition proviennent de la société HELION fournies par l'intermédiaire du LAPLACE.

Ces bases de données sont issues de campagnes de vieillissement réalisées sur des piles à combustible. On dispose de 3 campagnes de vieillissement réalisées sur des piles de 10 cellules et une campagne réalisée sur une pile de 100 cellules. La durée de ses campagnes varie entre 1700 h et 8000 h pour des données récupérées chaque 10 secondes. Les données récupérées dans cette campagne sont entre autres la tension de la pile, les tensions des cellules, la pression, les températures. Les informations détaillées relatives aux différents paramètres des piles, aux conditions opératoires nominales de fonctionnement et aux durées des plages sont disponibles en A.1.

L'indicateur de santé des piles couramment utilisé est la tension de la pile qui est facile à obtenir. Lors des campagnes, le principal critère de fin de vie de la pile utilisé pour mettre fin aux campagnes est que la tension de la pile subisse une perte de 10 ou 20% de sa valeur initiale. Le critère de fin de vie utilisé lors des campagnes de HELION était un autre. Les campagnes ont été arrêtées quand le taux de fuite interne dépasse un seuil limite [1]. Il existe donc différentes définitions de la "mort" d'une pile à combustible, mais dans cette étude, nous nous intéressons à la dégradation des performances de la pile.

La figure ci-dessous présente l'évolution des fuites dans la pile de la campagne 1. Les campagnes ont été arrêtées dès que les fuites étaient supérieures à $10\mu l/min/cm^2$. Un dépassement du seuil ne traduit pas forcément une perte de performance pour la pile. En effet, on peut remarquer que pour les campagnes 2, 3 et 4 la pile délivre toujours une bonne tension.

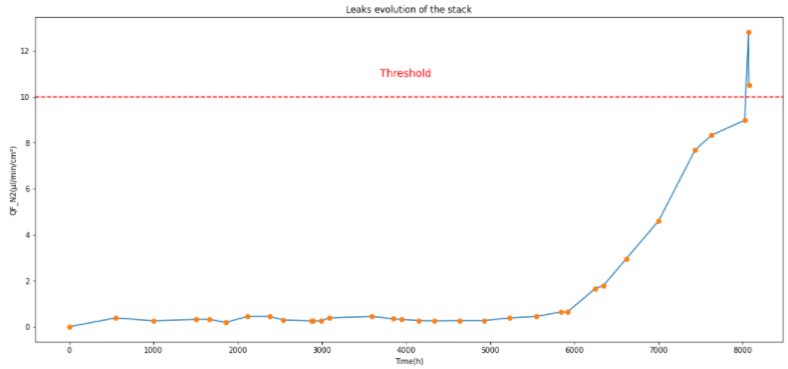
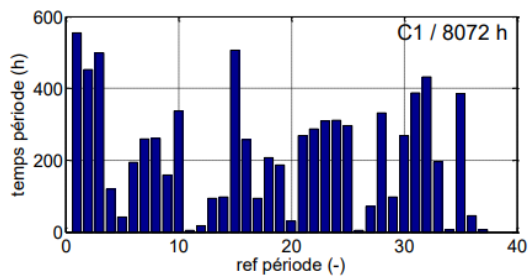


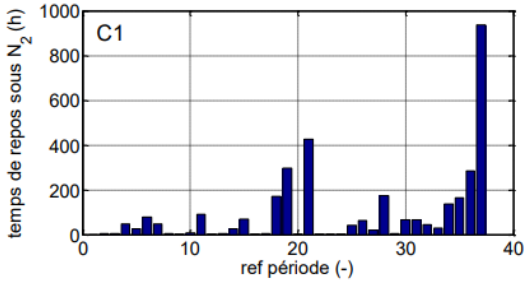
FIGURE 3.1 – Fuites de la campagne 1

Au cours des différentes campagnes, on a assisté à plusieurs phases d'arrêts/démarrages. On a ainsi des plages de fonctionnements, qui représente le temps entre un redémarrage et un arrêt. La campagne 1 possède 37 plages de fonctionnement de durée variable. La durée des temps d'arrêts ainsi que celle des plages de fonctionnement ne suit aucun plan d'expériences donné. La durée aléatoire des temps d'arrêts s'explique principalement par des contraintes pratiques et matérielles. La durée des différentes plages de la campagne 1 ainsi que la durée des phases de repos est présentée dans les figures 3.2a et 3.2b.



Source : [1]

(a) Durée des plages de fonctionnement de la campagne 1



Source : [1]

(b) Durée des phases de repos de la campagne 1

FIGURE 3.2 – Durée des plages et des phases de repos

On peut observer l'évolution des tensions de la campagne 1 constituée de 8000 h (environ 1 an) de données dans la figure ci-dessous.

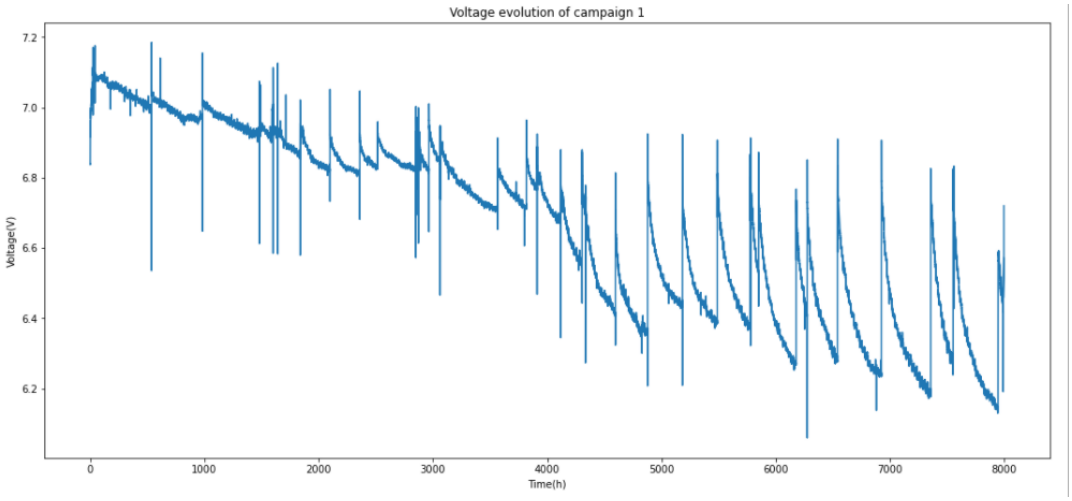


FIGURE 3.3 – Tension de la pile de la campagne 1

Cette évolution montre la dégradation irréversible que subit la pile (décroissance globale de la tension avec le temps) ainsi que le phénomène de dégradation réversible (traduit par les pics après chaque redémarrage).

La pile étant constituée de plusieurs cellules en série, la tension de la pile est définie comme étant la somme des tensions de chacune des cellules. On peut observer sur la figure 3.4, les tensions des différentes cellules de la pile de la campagne 1.

3.2 Analyse des corrélations entre les grandeurs

3.2.1 Mesure de corrélation de Chatterjee

Dans le cadre de l'identification des facteurs principaux, nous avons décidé d'utiliser la mesure de corrélation proposée par Chatterjee [21]. Cette nouvelle mesure propose un coefficient de corrélation aussi simple que les mesures classiques de Pearson et possède aussi une théorie asymptotique sous hypothèse d'indépendance comme les autres méthodes classiques. Elle offre en plus une mesure interprétable de la corrélation des données : elle est égale à 0 si et seulement si les variables sont indépendantes et à 1 si et seulement si une variable est une fonction mesurable de l'autre.

Notre choix s'est porté essentiellement sur la mesure de Chatterjee notamment pour sa capacité à bien caractériser la corrélation si une grandeur est une fonction *non-linéaire* d'une autre. Le processus de dégradation dans les piles à combustible faisant intervenir de nombreux phénomènes non-linéaires,

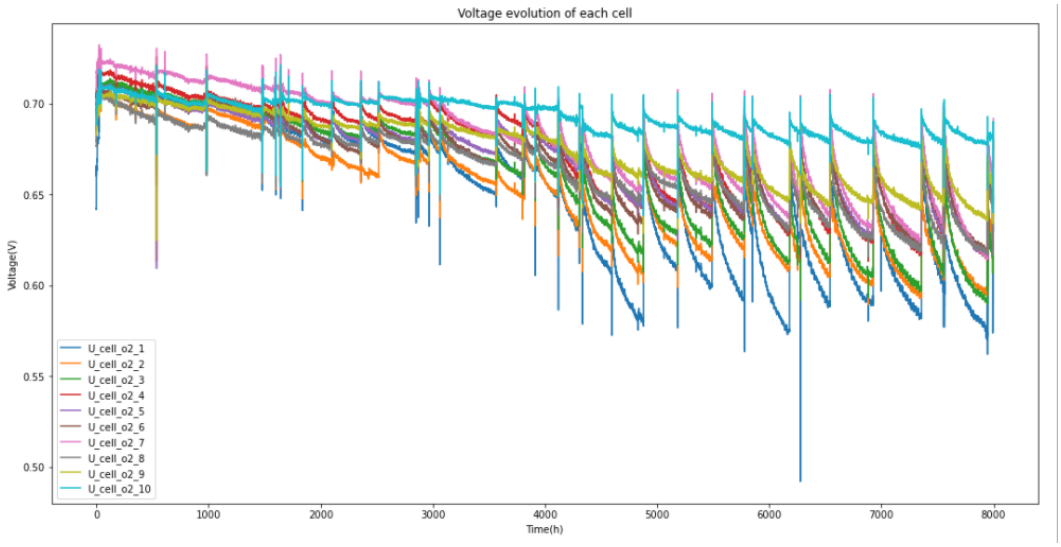


FIGURE 3.4 – Tension des cellules de la campagne 1

cette méthode nous paraît plus adaptée à notre cas d'usage, contrairement à la mesure de Pearson plus adaptée pour les relations *linéaires*.

Les détails concernant cette mesure sont donnés dans la section B.1.

3.2.2 Calcul des corrélations pour la campagne 1

Le calcul des corrélations à l'aide de cette mesure s'est fait grâce à l'implémentation R fournit par le package XICOR [22]. Nous présenterons ici les corrélations calculées sur la campagne 1. Les corrélations obtenues sur les autres campagnes sont disponibles en B.2.

Les corrélations ont été calculées relativement à la tension de la pile. On cherche donc à voir si une ou plusieurs autres des grandeurs présentes dans la base de données expliqueraient l'évolution de la tension de la pile. On utilise comme tension, la grandeur U_h2_stack qui caractérise la tension de la pile de la même façon que la grandeur U_o2_stack, les deux grandeurs étant sensiblement égales. Le tableau 3.5 présente les différentes corrélations obtenues sur la campagne 1 sous la forme d'une heatmap.

Comme le montre cette heatmap, on peut observer une forte corrélation entre les différentes tensions des cellules et la tension de la pile. En effet, la tension de la pile étant la somme des tensions des différentes cellules (cellules placées en série), observer de fortes corrélations à ce niveau n'est donc pas surprenant.

On observe des corrélations importantes entre la tension et les fuites et débits de fuites observées dans la pile, mais aussi entre la tension et certaines grandeurs comme la température monopolaire (T_M1_o2), la température bipolaire (T_B23_o2) ou la variation de charge (ΔP_o2).

L'analyse des corrélations effectuées ici a été réalisée de façon globale sur la totalité de la campagne (8000 h). Afin d'avoir une meilleure idée de l'influence des grandeurs de la campagne, nous avons analysé les corrélations calculées sur chacune des 37 plages de fonctionnement. Ces corrélations sont présentées dans la heatmap ci-dessous 3.6.

On peut noter que les corrélations suivent la même tendance que celles observées sur l'ensemble de la campagne. Toutefois, certains points sont à noter :

- Les corrélations avec les fuites et débits de fuite sont peu marquées au début de la campagne et deviennent plus importantes vers la fin de la campagne. Cette observation peut s'expliquer par le fait que les fuites deviennent plus importantes vers la fin de la campagne comme le montre la figure 3.1.
- Les corrélations avec les variations de pression et la température sont moins marquées sur les plages de fonctionnement que globalement.

Les résultats obtenus en observant les corrélations sur la campagne nous ont poussé à voir si ces corrélations se traduisaient dans les piles. Cependant, comme le mentionne Tognan [1] dans sa thèse, des mesures de fuite ont été réalisées sur la pile en fin de vie. Ces mesures montrent que les fuites ne

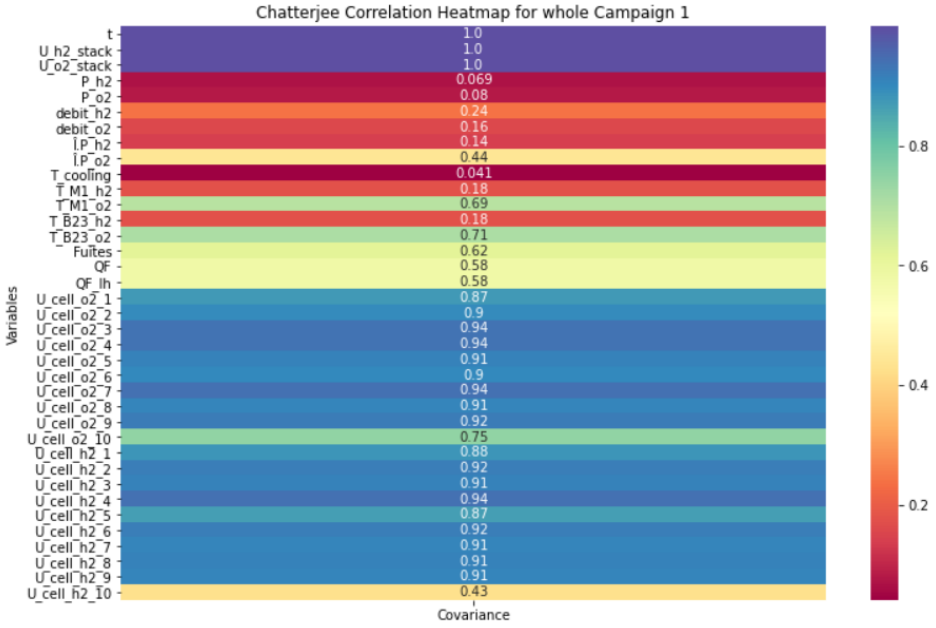


FIGURE 3.5 – Heatmap des corrélations globales de la campagne 1

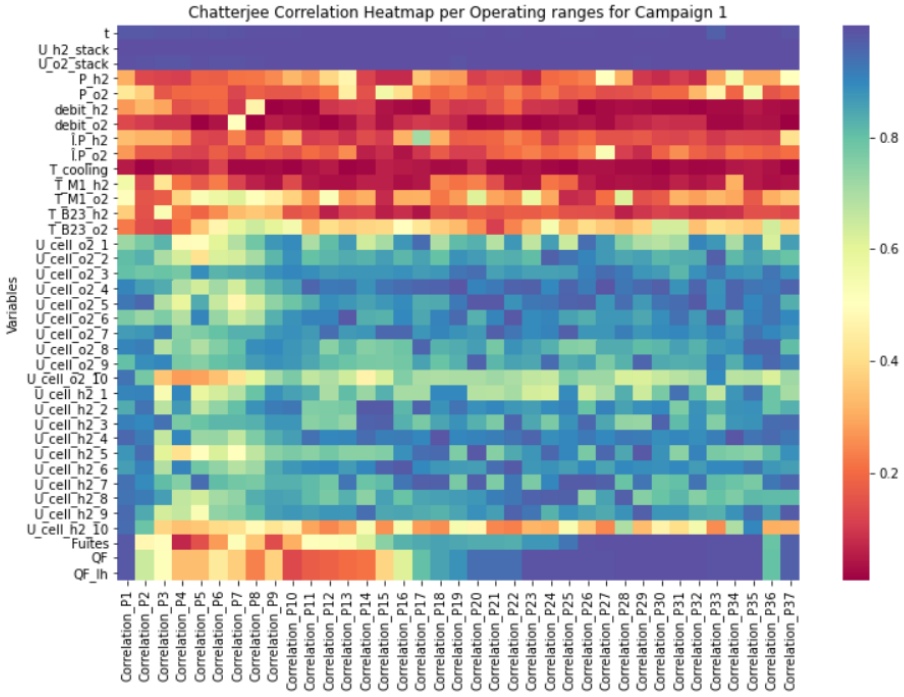
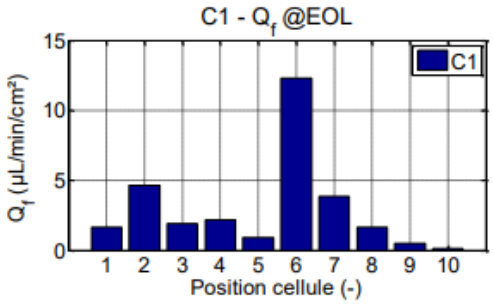


FIGURE 3.6 – Heatmap des corrélations par plage de fonctionnement de la campagne 1

sont pas uniformément reparties sur chaque cellule. Bien au contraire, on remarque que juste une partie des cellules (4 cellules pour la campagne 1) cumulent près de 80% des fuites dans la pile. Ce constat est confirmé par les analyses effectuées sur la campagne 4 où la pile est constituée de 100 cellules. On remarque ici que la fuite est plus importante et est essentiellement concentrée sur 4 cellules.

Ne disposant pas de mesures de la fuite sur chaque cellule à différents instants de la campagne, on peut supposer que la fuite est localisée et l'augmentation de cette fuite avec le temps n'est pas liée au nombre de cellules, mais à l'aggravation des premières fuites.

Afin de confirmer les analyses effectuées par Tognan, nous avons alors réalisé des calculs de corrélation entre les débits de fuites et les tensions des différentes cellules. Les cellules les plus fuyardes de la



Source : [1]

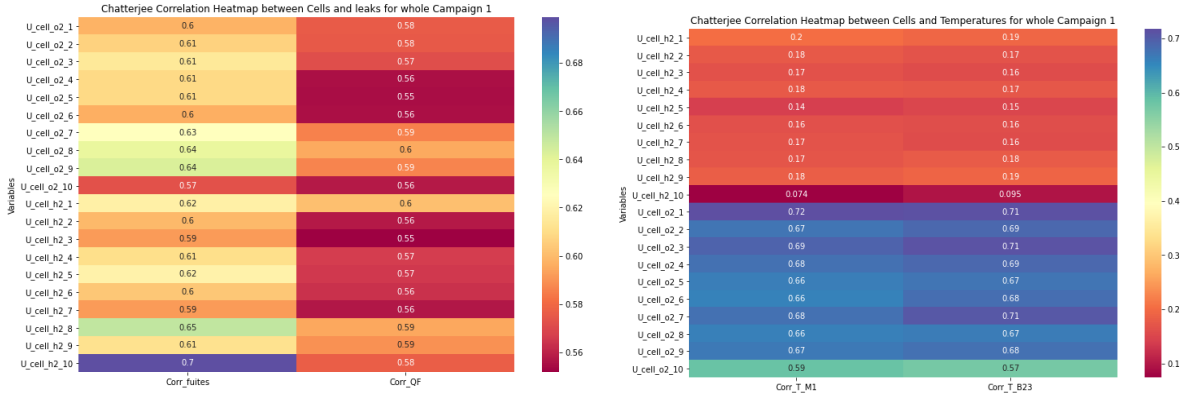
FIGURE 3.7 – Répartition des fuites dans les cellules de la pile

campagne 1 sont les cellules 6, 2 et 7 par ordre décroissant d'importance des fuites. En observant la figure 3.8a, on note que les cellules les plus corrélées avec les fuites et les débits de fuites sont les cellules 8 et 9 (parfois la 10). L'étude des corrélations montre des résultats assez contre-intuitifs, car on s'attendait à voir des corrélations plus importantes avec les cellules 6, 2 et 7.

Nous avons effectué une analyse similaire en étudiant les corrélations entre les tensions des cellules et les températures. La température T_M1 représente la température récupérée au niveau de la cellule 1 tandis que la température T_B23 représente quant à elle la température mesurée entre les cellules 2 et 3. On dispose dans la campagne de ces températures pour l'H_2 et l'O_2.

En observant les corrélations présentées dans la figure 3.8b, on remarque que les températures en H_2, sont faiblement corrélées avec les tensions des cellules tandis que celles en O_2 sont plus corrélées.

En observant ces corrélations, on s'attendrait à remarquer que les tensions de la cellule 1 soit nettement plus corrélées avec la température T_M1 que les autres cellules. On observe en effet que la cellule 1 possède la corrélation la plus marquée, mais l'écart avec les cellules 3 et 7 n'est pas très significatif. De même, en observant les corrélations avec T_B23, on devrait avoir les cellules 2 et 3 qui sont plus corrélées, mais on remarque ici que les cellules 1 et 7 sont aussi très corrélées avec cette température.



(a) Corrélations entres les tensions des cellules et les fuites de la campagne 1

(b) Corrélations entres les tensions des cellules et les températures de la campagne 1

FIGURE 3.8 – Analyses des corrélations des fuites et des températures

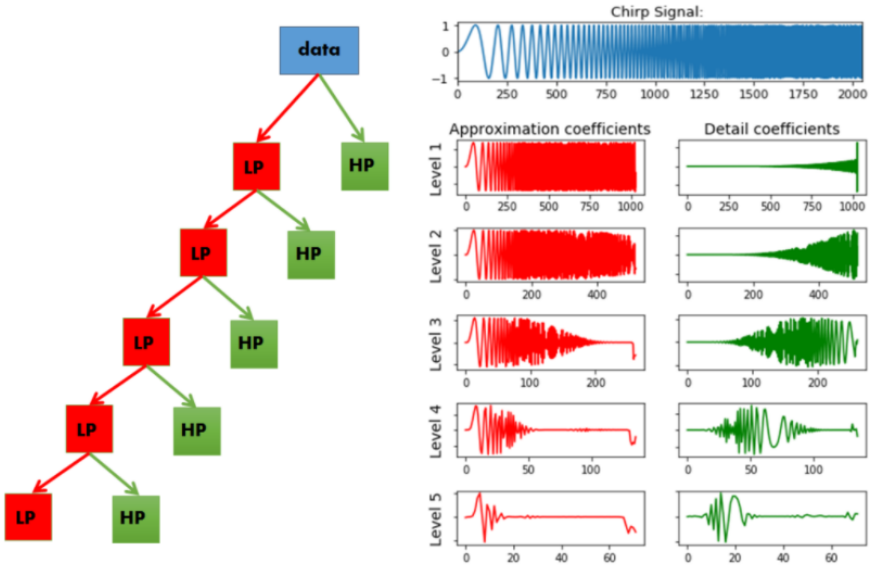
3.3 Analyse des tensions de la pile de la campagne 1

Après avoir effectué l'analyse des corrélations sur l'ensemble de la campagne 1, nous avons décidé de nous concentrer sur la tension de la pile afin de chercher à observer des phénomènes particuliers.

3.3.1 Transformée en ondelettes

Afin de savoir si certains phénomènes de la dégradation de la pile apparaissait dans la tension, nous avons effectué une analyse multi-résolution de celle-ci en se basant sur la transformée en ondelettes. Un des principaux avantages de la transformée en ondelettes, c'est qu'elle possède une grande résolution aussi bien dans le domaine fréquentiel que temporel. Cela permet de pouvoir identifier les différentes fréquences présentes dans le signal, mais aussi à quel instant elles apparaissent. Elle est de ce fait plus efficace que la transformée de Fourier qui elle, ne possède qu'une grande résolution dans le domaine fréquentiel.

Dans cette étude, nous avons utilisé une transformée en ondelettes discrète (DWT). Cette transformée permet d'analyser le signal à plusieurs échelles en commençant par les plus petits niveaux qui correspondent aux plus hautes fréquences et au niveau suivant on analyse les fréquences autour de la moitié de la fréquence maximale. On peut répéter le processus jusqu'au niveau maximal de décomposition. La figure 3.9 explique le fonctionnement de la DWT.



Source : https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

FIGURE 3.9 – Transformée en ondelettes discrète d'un signal sur plusieurs niveaux

Nous avons donc réalisé la transformée en ondelettes de la tension de la pile de la campagne 1. L'ondelette utilisée est l'ondelette de *Daubechies* via l'implémentation fournie par le package PyWavelets [23] sur Python. On peut observer le résultat sur la figure 3.10.

En analysant cette décomposition de la tension de la pile, on peut remarquer dans les coefficients de détails la présence de pics. Ces pics sont pour la plupart dus aux pics présents dans la tension de la pile. Ces pics sont causés par le phénomène des arrêts démarrages présents tout au long de la campagne. L'analyse à l'aide des ondelettes ne nous a pas permis de trouver la présence d'autres phénomènes pouvant influencer le vieillissement. Nous avons donc décidé d'effectuer une analyse plus fine à l'aide d'autres méthodes afin d'identifier d'autres phénomènes.

3.3.2 Analyse temporelle de la tension de la pile

Nous avons observé la tension de la pile en superposant les évolutions de la tension par plage de fonctionnement. Comme le montre la figure 3.11, on peut observer que l'évolution de la tension est linéaire pour les toutes premières plages de fonctionnement. Les dernières plages sont caractérisées par un pic suivi par une décroissance exponentielle. Un phénomène caractéristique de cette campagne est la présence d'oscillations traduisant un phénomène périodique dans le vieillissement de la pile. Selon Tognan [1], ce phénomène oscillatoire s'explique par des variations cycliques journalières de la pression d'entrée en O_2.

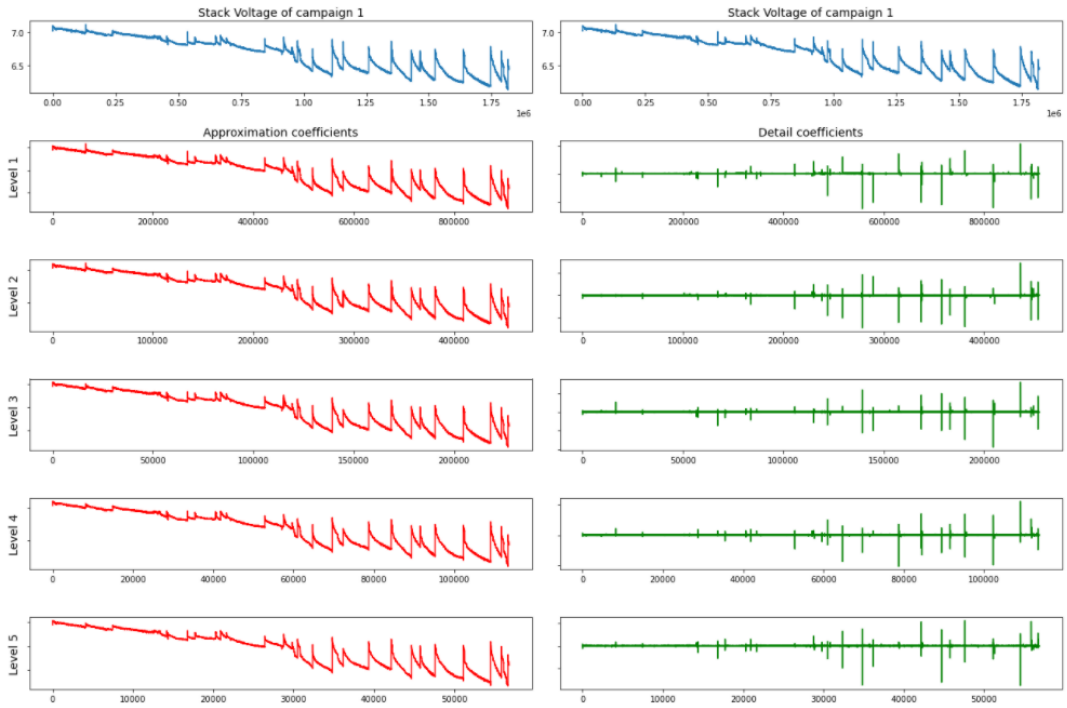


FIGURE 3.10 – Transformée en ondelettes en cascades de la tension

La présence de ce phénomène oscillatoire nous a poussé à chercher la présence de phénomènes périodiques ou saisonniers dans l'évolution de la tension. Une série temporelle peut se décomposer comme étant la somme ou le produit de trois (03) termes :

- un terme de tendance : la tendance renvoie à la direction de la série lors de son évolution dans le temps.
- un terme de saisonnalité : la saisonnalité renvoie aux évolutions cycliques de la série temporelle.
- un terme de résidu : le résidu renvoie au terme restant dans la série lorsque l'on retire le terme de tendance et le terme de saisonnalité.

Il existe plusieurs façons de décomposer un signal temporel. Nous avons décidé d'utiliser une méthode de décomposition naïve additive. On peut donc écrire :

$$Y(t) = T(t) + S(t) + R(t) \tag{3.1}$$

où Y désigne la série temporelle, T la tendance de la série, S le terme de saisonnalité et R le résidu.

Cette méthode présente des inconvénients notamment, car elle estime que le terme de saisonnalité reste constant tout au long de l'entière série. Cette décomposition a confirmé la présence de nombreux "patterns" de saisonnalité dans les données mais qui évoluent avec aussi bien en amplitude que dans la forme avec le temps. Il est donc important d'avoir des méthodes pouvant en tenir compte afin de fournir de meilleures estimations.

3.4 Conclusion

Il ressort de ces analyses de corrélation, que les fortes corrélations relevées entre la tension de la pile et certaines grandeurs comme les fuites ou la température ne traduisent pas forcément le comportement de la pile. La base de données à notre disposition ne nous permet pas d'identifier des grandeurs pouvant être considérées comme indicateurs de santé de la pile, autre que la tension de la pile.

Zhang et al [6] mentionne dans son article qu'utiliser la tension de la pile comme indicateur de santé est généralement un bon choix, mais la présence de phénomènes réversibles et irréversibles empêchent d'avoir une bonne estimation du vieillissement de la pile. Comme on peut le remarquer dans notre cas

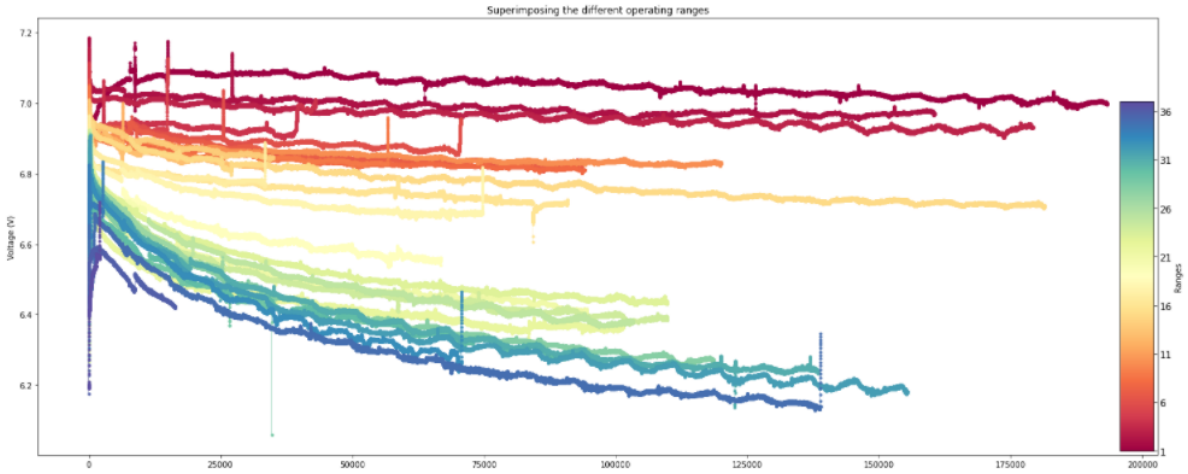


FIGURE 3.11 – Superposition des tensions de chaque plage de fonctionnement

d'études, et sur les tensions des autres campagnes (C2, C3 et C4), la pile peut délivrer toujours une bonne tension alors qu'elle subit des phénomènes de fuites dans ses cellules. Les informations permettant de mieux caractériser la dégradation au sein de la pile sont disponibles par des spectroscopies d'impédance électrochimique ou des courbes de polarisation. Cependant, ces mesures sont assez délicates à réaliser en raison de leur caractère intrusif qui nécessite souvent de détériorer la pile afin de les effectuer. Ces informations sont pourtant nécessaires afin de réussir à fournir une meilleure estimation du vieillissement de la pile à combustible.

Ainsi, pour la suite de nos travaux et pour nos différentes modélisations, seule la tension de la pile sera utilisée.

Chapitre 4

Description des modèles implémentés

Nous décrivons dans cette section les différents modèles et architectures que nous avons implémentés pour la modélisation du vieillissement de la pile à combustible. Les résultats des expérimentations seront présentés dans le chapitre suivant. Il ressort des sections précédentes qu'une bonne partie des données présentes dans notre base de données n'est pas utilisable, car n'ayant pas un impact considérable sur le vieillissement des piles. Nous allons donc utiliser la tension de la pile comme information principale pour les modélisations.

4.1 Modèle physique

4.1.1 Élaboration du modèle physique

Le premier type de modèle que nous décrivons est un modèle physique basé sur les lois de dégradations de la pile connues, couplés à des modélisations empiriques et semi-empiriques.

Dans la littérature, le modèle de dégradation le plus utilisé est celui construit à partir de l'équation de polarisation [3, 5]. Cette équation modélise les pertes qui affectent la tension réversible de la cellule E_{rev}, encore appelée *tension de Nernst*. Cette tension correspond à la tension électrique obtenue par la réaction chimique s'il n'existait aucune perte. Les différentes pertes qui peuvent affecter la pile sont :

- les pertes d'activation,
- les pertes de diffusion,
- les pertes ohmiques,
- les pertes de crossover.

Il existe différentes modélisations, mais pour une première approche, nous nous sommes basés sur celle utilisée par Tognan et al.

$$V_{cell}(J) = E_{rev} - \eta_{act} - \eta_{diff} - \eta_{ohm} \tag{4.1}$$

où V_{cell} désigne la tension de la cellule, E_{rev} la tension de Nernst, η_{act} les pertes d'activation, η_{diff} les pertes de diffusion, η_{ohm} les pertes ohmiques.

Dans le cadre de nos travaux, des hypothèses et simplifications ont été adoptées afin d'avoir un modèle capable de traduire les différentes pertes réversibles et irréversibles [24, 25, 26, 27]. L'ensemble de ces simplifications est détaillé en annexe C.1.

L'équation de modélisation de la tension de la cellule de la pile retenue à l'issue de ces modélisations est donnée comme suit :

$$V_{cell}(J,t) = E_{rev}(P,T) - \frac{RT}{\alpha nF} ln(\frac{J + J_n}{J_0(t_0) + \tau_{degr}(t)}) - (\frac{R^A_{diff}(t_0)}{\tau_{degr}(t)} + R^B_{diff}(t_0)) * (J + J_n) - R_{ohm}(t) * (J + J_n) \tag{4.2}$$

où

$$\tau_{degr}(t) = s_{min} + (1 - s_{min}) * e^{-(k_{irrev} * t + k_{rev}(t) * \Delta t)}$$
$$R_{ohm}(t) = R_{ohm}^{BOL} + \frac{R_{ohm}^{EOL} - R_{ohm}^{BOL}}{duree_{campagne}} * t$$

4.1.2 Paramétrage du modèle

Le modèle physique proposé doit être paramétré afin de pouvoir prédire l'évolution de la tension dans le temps. Ce modèle physique comporte de nombreux paramètres qui doivent être estimés à partir des données expérimentales. Les paramètres inconnus du modèle sont donnés dans le tableau suivant.

Paramètres	Borne min	Borne max	Unité
J_o	10^{-8}	10^{-3}	A/cm^2
$R_{diff}^A(t_0)$	0	0.5	Ω/cm^2
$R_{diff}^A(t_0)$	0	0.5	Ω/cm^2
k_{irrev}	0	0.1	h^{-1}
k_{rev}	0	0.1	h^{-1}

TABLE 4.1 – Paramètres à estimer du modèle physique

Nous avons choisi, pour l'estimation de ces paramètres, de passer par une méthode d'optimisation par moindres carrés. Il existe plusieurs autres méthodes d'optimisation, mais nous avons décidé d'utiliser celle-ci en première approximation. L'algorithme utilisé est celui de Levenberg-Marquardt [28, 29]. Cet algorithme se base sur l'algorithme de Gauss-Newton et la descente de gradient pour trouver la solution numérique d'une fonction de plusieurs paramètres. Pour un ensemble de points (t_i, U_i) $\forall i \in [|1; n|]$, pour une fonction f fonction de t et d'un ensemble de paramètres noté vectoriellement $\theta = \begin{pmatrix} \theta_1 \\ . \\ . \\ \theta_b \end{pmatrix}$, on cherche alors les paramètres θ qui minimisent la somme des carrés des erreurs :

$$S(\theta) = \sum_{i=0}^{N} [U_i - f(t_i|\theta)]^2$$

Dans le cadre de notre modèle physique, nous disposons de bornes minimum et maximum pour les paramètres à estimer. L'algorithme utilisé pour estimer les paramètres est couplé à une méthode de régions de confiance [30]. Cet algorithme recherche une approximation de la solution dans une région dite *"de confiance"* (généralement une boule). Si une approximation satisfaisante est trouvée dans cette région, elle est agrandie sinon elle est réduite.

L'implémentation du paramétrage du modèle sur les données s'est fait grâce à la librairie Python *LMFIT* [31]. Cette librairie permet d'encapsuler l'estimation des paramètres tout en offrant une flexibilité d'implémentation et de paramétrage. Elle fournit des informations statistiques sur le modèle physique comme l'incertitude sur les paramètres estimés ($1-\sigma$incertitude) ainsi que les corrélations entre les paramètres. On retrouve aussi d'autres informations comme le nombre d'itérations, l'information sur l'erreur du modèle physique comme le critère AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), ainsi que le χ-square ou le χ-square réduit.

Les deux critères AIC et BIC [32] sont des mesures d'information de la qualité d'un modèle prédictif. Le critère AIC pénalise les modèles en fonction du nombre de paramètres tandis que le modèle BIC, qui est une variante du AIC, dépend aussi de la taille de l'échantillon. Les deux critères permettent de satisfaire au principe de parcimonie (Rasoir d'Ockham). Les formules de ces critères sont données ci-après.

$$AIC = 2k - 2ln(L)$$
$$BIC = -2ln(L) + kln(n)$$

où k désigne le nombre de paramètres libres du modèle, L la vraisemblance du modèle estimé et n le nombre d'observations du modèle.

Pour plusieurs modèles $(M_1, ...M_d)$, le meilleur modèle sera celui qui minimise le critère AIC (ou BIC).

$$M_{best} = arg \min_{j \in [|1;d|]} AIC(M_j) \tag{4.3}$$

4.2 Réseaux de neurones

Il existe dans la littérature de nombreux modèles statistiques ou d'architectures de réseaux de neurones, qui ont été utilisés dans le cadre de la modélisation du vieillissement des piles à combustible. Nous nous sommes intéressés dans ce stage qu'aux architectures de réseaux de neurones profond (*Deep Learning*). Nous avons décidé de nous focaliser sur ces réseaux en raison de leurs capacités à modéliser les non-linéarités. Les autres modèles d'apprentissage (processus gaussien, machines à vecteurs de support etc) pourront faire l'objet d'autres travaux. Cette section présente brièvement les différentes architectures ainsi que les raisons qui ont justifié leurs choix.

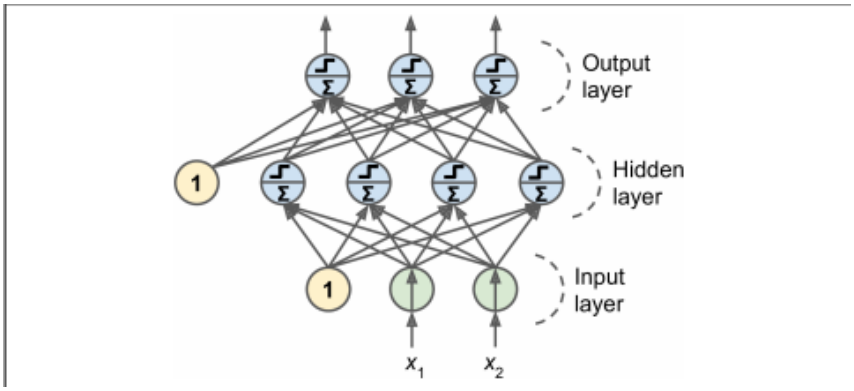
4.2.1 Réseaux de neurones artificiels (ANN)

La première architecture que nous avons décidé de tester est le réseau de neurones artificiels. L'observation de l'évolution de la tension de la campagne 1 3.3, nous a montré que l'évolution de la tension est linéaire sur les premières plages de fonctionnement puis exponentielle sur les dernières plages. L'architecture ANN est donc la plus simple pour effectuer des modélisations linéaires, mais aussi des modélisations non-linéaires (grâce aux fonctions d'activation).

L'architecture de base des réseaux de neurones artificiels est le perceptron. Le perceptron est constitué d'une couche de neurones LTU (*Linear Threshold Unit*). Le LTU effectue une combinaison linéaire des entrées et effectue une classification binaire.

Le perceptron est dans sa version simplifiée, monocouche, c'est-à-dire constitué d'une entrée et d'une sortie. En pratique, on utilise le perceptron multicouche (MLP), qui est plus efficace pour les plus grands problèmes non-linéaires. Il attribue une matrice de poids W et une matrice de biais b aux entrées X et calcule la sortie à laquelle sera appliquée une fonction d'activation non-linéaire ϕ. Dans le cas où les neurones de la couche précédente sont entièrement connectés à ceux de la couche supérieure (comme le montre la figure 4.1 qui présente un MLP), on parle alors de couche entièrement connectée (*Fully Connected Layer*).

$$Y = \phi(WX + b) \tag{4.4}$$



Source : [33]

FIGURE 4.1 – Fonctionnement d'un MLP

Dans notre étude, nous avons implémenté une couche de MLP sans activation pour avoir un comportement linéaire. Ce modèle de base n'est pas forcément adapté à nos données en raison de l'hypothèse linéaire, mais constituera l'architecture de référence de nos travaux.

4.2.2 Réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs sont une version régularisée du perceptron multicouche, inspirés du cortex visuel des vertébrés, qui tentent de limiter les entrées des neurones en ne conservant que les relations spatiales (ou temporelles) des entrées. Ils sont très utilisés pour le traitement d'image (2D), mais s'avèrent aussi très efficaces pour le traitement de données temporelles (1D). L'avantage des CNN sur les données temporelles est que l'on peut apprendre facilement les informations à court terme et effectuer des calculs plus rapides. Les prédictions sont effectuées en se basant sur le passé à court terme. Les CNNs sont, de plus, des modèles très résistants au bruit, et ils sont capables d'extraire des caractéristiques très informatives et profondes, qui sont indépendantes du temps.

Le fonctionnement des CNN peut s'expliquer comme suit :

Pour une série temporelle $x_{(t)} = (x_{(1)}, x_{(2)}, ..., x_{(N)})$, le réseau CNN qui est constitué d'un noyau (kernel) $K = (k_1, k_2, ..., k_m)$ de taille $m << N$, va effectuer un produit de convolution entre une fraction de l'entrée, de même taille que le noyau et le noyau. Cette opération sera appliquée en décalant le noyau d'un pas, jusqu'à ce qu'elles soient appliquées sur toutes les données. Ce fonctionnement est décrit dans la figure 4.2 avec un noyau de taille 3.

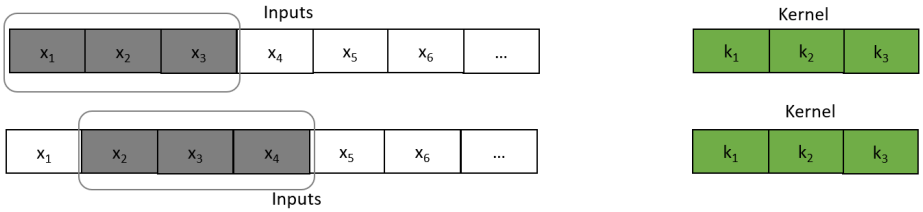
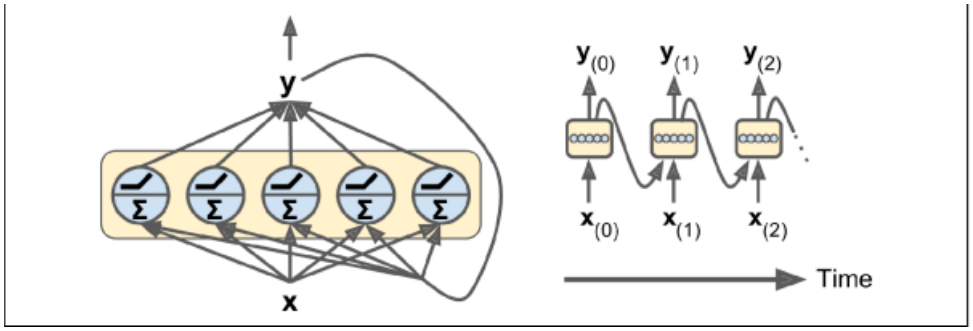


FIGURE 4.2 – Fonctionnement d'un CNN en dimension 1

4.2.3 Réseaux de neurones récurrents (RNN)

Les réseaux de neurones récurrents sont des architectures particulièrement adaptées pour les signaux temporels non-linéaires. Le réseau RNN le plus simple est composé d'un neurone qui reçoit une entrée, produit une sortie et se renvoie sa sortie. Dans le cas d'une couche de RNN, à chaque instant t, le neurone i reçoit l'entrée $x_{(t)}$ et la sortie du neurone précédent, $y_{(t-1)}$. Il transmet alors sa sortie $y_{(t)}$ au neurone $i+1$, comme le montre la figure 4.3.



Source : [33]

FIGURE 4.3 – Fonctionnement d'un RNN

Chaque neurone dispose de types de poids : les poids associés aux entrées $x_{(t)}$ que l'on note w_x et ceux associés aux entrées du neurone précédent $y_{(t-1)}$, que l'on note w_y. En considérant toute une couche de RNN et en passant à une notation vectorielle, on peut noter l'équation de la sortie d'une couche RNN par :

$$Y_{(t)} = \phi(X_{(t)}W_x + Y_{(t-1)}W_y + b) \tag{4.5}$$

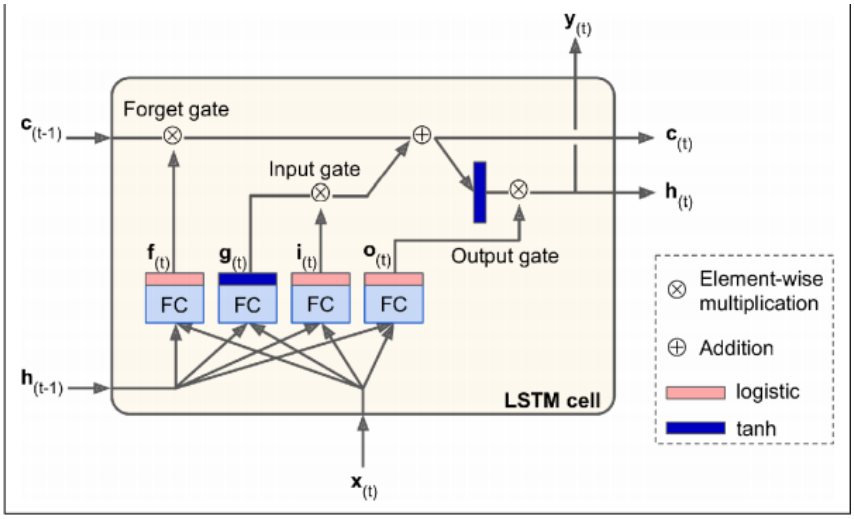
où b désigne le vecteur de biais et ϕ désigne la fonction d'activation non-linéaire utilisée. Dans le cas des RNN, la fonction d'activation couramment utilisée est la fonction *tanh* tangente hyperbolique. Cette fonction est très efficace pour éviter le problème des *exploding gradients* qui est beaucoup plus accentuée dans les RNN, notamment avec la fonction *ReLU*. Si la descente de gradients modifie les poids des neurones au point d'augmenter les sorties, sachant que les poids seront utilisés à chaque pas de temps, on aura une sortie qui ne fait qu'augmenter. Les fonctions d'activation comme la *ReLU* qui sont non saturantes n'empêchent pas ce phénomène ce qui conduit au phénomène des *exploding gradients*.

Dans le cas de la prédiction des séries temporelles, les RNN sont efficaces et ne nécessitent pas de retirer la saisonnalité et la tendance des données, comme c'est le cas dans les modèles ARIMA (weighted moving average models ou autoregressive integrated moving average).

4.2.4 Long Short Term Memory (LSTM)

Un des principaux inconvénients présentés par les RNN, est sa difficulté à gérer les longues séquences. Avec toutes les transformations que subissent les données lors de leur passage dans les RNN, une partie de l'information est perdue à chaque fois, si bien qu'au bout d'un moment, le réseau ne contient aucune trace des premières entrées.

Dans le but de résoudre ce problème l'architecture LSTM a été proposée par Sepp Hochreiter et Jürgen Schmidhuber en 1997 [34]. Cette architecture est plus efficace, converge plus rapidement et détecte les longues dépendances dans les données. Comme le montre la figure 4.4, la cellule LSTM prend en entrée trois vecteurs, l'état à court terme $h_{(t-1)}$, l'état à long terme $c_{(t-1)}$ et l'entrée $x_{(t)}$. La couche $g_{(t)}$ est la couche principale qui analyse les entrées et l'état court-terme précédent afin de produire la sortie $y_{(t)}$ et le nouvel état court-terme $h_{(t)}$. Les couches *Fully Connected (FC)* $f_{(t)}$, $i_{(t)}$ et $o_{(t)}$ contrôlent les portes d'oubli, d'entrée et de sortie. Ce sont elles qui permettent d'ajouter ou de retirer de "la mémoire" à l'état long-terme précédent afin de produire le nouvel état long-terme $c_{(t)}$.



Source : [33]

FIGURE 4.4 – Fonctionnement d'un LSTM

Les équations suivantes résument l'ensemble des calculs à l'intérieur d'une LSTM. On notera $W_?$ et $b_?$ les matrices de poids et de biais respectivement, et on indexera par x et h les matrices associées à l'entrée et à l'état court-terme respectivement.

$$
\begin{aligned}
i_{(t)} &= \sigma(W_{xi}^T x_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \\
f_{(t)} &= \sigma(W_{xf}^T x_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \\
o_{(t)} &= \sigma(W_{xo}^T x_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \\
g_{(t)} &= tanh(W_{xg}^T x_{(t)} + W_{hg}^T h_{(t-1)} + b_g) \\
c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \\
y_{(t)} &= h_{(t)} = o_{(t)} \otimes tanh(c_{(t)})
\end{aligned}
$$

4.2.5 Entraînement des réseaux de neurones

Dans le cadre de notre problème, nous cherchons à prédire l'évolution de la tension dans le temps. Nous utiliserons alors un réseau de neurones qui va apprendre à effectuer cette prédiction à partir d'un couple de données (X, Y) où X désigne les entrées et Y la sortie que l'on souhaite obtenir. Il s'agit là d'un problème de régression en apprentissage supervisé (*supervised learning*) que l'on peut formuler comme suit :

Pour un jeu de données $(x_1, y_1), ..., (x_n, y_n)$, où x_i désigne une entrée et y_i un algorithme d'apprentissage va chercher une fonction g capable pour un x_j inconnu de prédire la valeur y_j associée.

Pour entraîner des réseaux de neurones dans le cadre d'un problème d'apprentissage supervisé, l'algorithme utilisé est celui de la rétropropagation des gradients (*backpropagation*). Cet algorithme se présente comme suit :

- Faire passer à travers le réseau de neurones les entrées des données d'apprentissage par mini-lots (x_j). Chaque couche du réseau calcule la sortie et la transmet à la couche suivante jusqu'à la couche de sortie. C'est ce qu'on appelle la propagation vers l'avant (*forward pass*).
- À la fin de la (*forward pass*), on obtient les sorties $(\tilde{y_j})$. On calcule l'erreur entre la sortie du réseau $(\tilde{y_j})$ et la sortie attendue (y_j) en utilisant la fonction objectif (*loss function*).
- L'algorithme calcule à quel point chaque neurone du réseau à contribuer à cette erreur qui est ensuite propagée aux neurones du réseau qui va mettre à jour les poids de chaque neurone avec la méthode de la descente de gradient.

Dans notre problème d'apprentissage, il est important de bien choisir la fonction objectif à minimiser. Comme il s'agit d'un problème de régression, nous avons décidé de prendre comme fonction de coût la racine carrée de l'erreur quadratique moyenne (RMSE) (*Root Mean Square Error*). Cette erreur est définie comme suit :

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \tilde{y_i})^2}{N}} \tag{4.6}$$

où $y_i, \tilde{y_i}, N$ désignent respectivement la donnée à prédire, la donnée prédite et la taille de l'échantillon.

Les données que nous cherchons à prédire étant des données temporelles, il est courant d'utiliser comme métrique de la performance du modèle, l'erreur moyenne absolue MAE (*Mean Absolute Error*), qui est plus robuste aux déviations que la RMSE. Cette métrique, qui sera celle utilisée dans tous les travaux, est définie comme suit :

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \tilde{y_i}|}{N} \tag{4.7}$$

4.3 Modèle hybride

Comme mentionné dans l'état de l'art, les méthodes physiques et les méthodes d'apprentissage présentent toutes deux des inconvénients dans le cas des problèmes de prédiction du vieillissement des piles à combustible. En vue de résoudre les inconvénients de ces deux méthodes, nous avons décidé d'adopter une approche hybride. La méthode la plus utilisée dans la littérature pour l'élaboration de modèles hybrides reste les méthodes d'ensemble. Les méthodes d'ensemble permettent de combiner plusieurs modèles dit *weak learners*, possédant de faibles performances, afin d'obtenir un modèle plus robuste ayant de meilleures performances et mieux généralisable. Qiu [35] catégorise les méthodes d'ensemble en deux :

- Les méthodes d'ensemble concurrentes : qui consistent à construire plusieurs modèles (différents ou non) qui vont apprendre à effectuer la prédiction sur les données. Les prédictions de ces modèles seront ensuite agrégées afin d'obtenir les prédictions finales.
- Les méthodes d'ensemble coopératives : qui reposent sur le principe de "diviser pour régner". La tâche originale est décomposée en différentes tâches qui seront apprises par différents modèles.

Dans cette étude, nous nous intéresserons principalement aux méthodes d'ensemble dites concurrentes plutôt qu'à celles dites coopératives, mais celles-ci pourraient être envisagées pour de prochains travaux.

Le choix des méthodes d'ensemble se justifie particulièrement dans notre étude, pour des raisons d'interprétabilité de modèles. Combiner les prédictions d'un modèle dit "boite noire" et un autre dit "boite blanche" permettrait d'obtenir un modèle dit "boite grise" dont les prédictions sont plus explicables.

Un autre avantage à utiliser des méthodes d'ensemble est qu'elles permettent d'obtenir une erreur de prédiction moindre que si on utilisait chacun des modèles séparément. Cette erreur de prédiction peut se décomposer comme suit [36] :

$$
\begin{aligned}
E[\tilde{y} - y]^2 &= bias^2 + \frac{1}{N} var + (1 - \frac{1}{N}) covar \\
bias &= \frac{1}{N} \sum_{i=1}^{N} (E[\tilde{y_i} - y]) \\
var &= \frac{1}{N} \sum_{i=1}^{N} E[\tilde{y_i} - E[\tilde{y_i}]^2 \\
covar &= \frac{1}{N(N-1)} \sum_{i} \sum_{j \neq i} E[[\tilde{y_i} - E[\tilde{y_i}][\tilde{y_j} - E[\tilde{y_j}]]
\end{aligned}
$$

où $\tilde{y}$ désigne la prédiction du modèle d'ensemble, $\tilde{y_i}$ celle d'un modèle i, N le nombre de modèles indépendants. *bias* , *var* et *covar* désigne respectivement le biais moyen, la variance moyenne et le terme de covariance des modèles de la méthode d'ensemble.

Les méthodes d'ensemble sont d'autant plus efficaces que les modèles qui les composent sont indépendants. Les modèles que nous utiliserons (à savoir le modèle physique et notre approche neuronale) étant fondamentalement différents et donc indépendants, la méthode d'ensemble est censée fournir de meilleurs résultats. Il existe différentes méthodes d'ensemble parmi lesquelles on peut citer celles qui réduisent la variance parmi les différents *weak learners* (*bagging*) ou d'autres qui réduisent à la fois le biais et la variance (*boosting*). Dans nos travaux, nous allons nous intéresser trois types de méthodes d'ensemble :

- Le vote à majorité (*Majority Voting*)
- La moyenne pondérée des modèles (*Weighted Average Model*)
- la méthode *Stacked Generalization*

4.3.1 Majority Voting

L'approche du vote à majorité consiste, dans le cas d'une régression, à effectuer la moyenne des prédictions de chaque modèle. Cette approche assez intuitive consiste à donner le même poids à chacun des modèles et donc considérer que chaque modèle est aussi performant l'un que l'autre. En considérant donc m modèles et leurs prédictions respectives $y_1, y_2, ..., y_m$, la prédiction du modèle d'ensemble s'écrit :

$$
y_{voting} = \frac{1}{m} \sum_{i=1}^{m} y_i \tag{4.8}
$$

La figure 4.5 illustre le fonctionnement de cette méthode.

4.3.2 Weighted Average Model

Plutôt que de considérer que chaque modèle a sensiblement la même performance, une autre approche serait de donner plus d'importance aux modèles ayant les meilleures performances. Le principe de fonctionnement de cette approche est le même que celui du vote à majorité. La différence est que l'idée du Weighted Average Model est d'attribuer des poids plus importants aux modèles les plus performants.

Nous avons donc choisi d'attribuer des poids inversement proportionnels à la somme des valeurs absolues des résidus de ces modèles sur les données de validation. En considérant m modèles et leurs erreurs sur les données de validation $e_1, e_2, .., e_m$, le poids w_i d'un modèle s'écrit :

$$
w_i = \frac{1}{e_i} = \frac{1}{\sum_{j=1}^{n_{val}} |y_j^{val} - \tilde{y_{i,j}^{val}}|} \tag{4.9}
$$

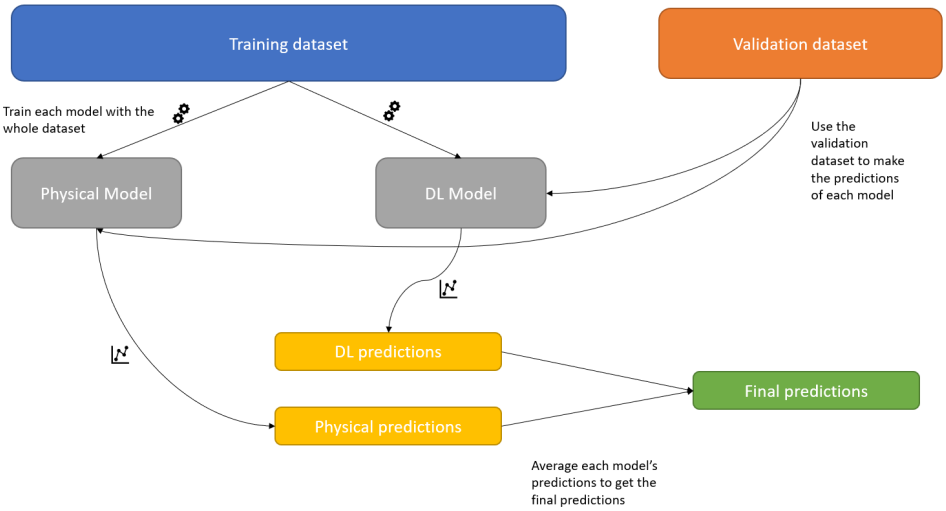


FIGURE 4.5 – Architecture de la méthode Majority Voting

où n_{val} désigne la taille des données de validation, y_j^{val} les labels dans les données de validation et $\tilde{y_{i,j}^{val}}$ la prédiction du modèle i sur les données de validation.

La prédiction finale de la méthode d'ensemble est donnée alors comme suit :

$$y_{weighted} = \sum_{i=1}^{m} w_{norm,i} y_i \tag{4.10}$$

où $w_{norm,i} = \frac{w_i}{\sum_{k=1}^{m} w_p}$ est le poids normalisé par rapport aux autres poids attribués et y_i la prédiction de chaque modèle de la méthode d'ensemble.

4.3.3 Stacked Generalization

La méthode Stacked Generalization (Stacking) a été introduite par David Wolpert. L'idée générale de cette approche est de construire un *meta-model* qui va apprendre à partir des prédictions des modèles dits "faibles" (*weak learners*) afin de réduire les biais des weak learners.

En considérant pour m modèles, leurs prédictions respectives $y_1, y_2, ..., y_m$, on cherche le méta-modèle qui fournira la combinaison optimale des poids w_i $\forall i \in [|1; m|]$. On définit la prédiction du modèle de stacking comme suit :

$$y_{stacking} = \sum_{i=1}^{m} w_i y_i \tag{4.11}$$

Dans sa formulation générale, cette approche se base sur l'approche de la validation croisée et un modèle aux moindres carrés. L'approche de la validation croisée, qui permet d'avoir des modèles plus robustes, consiste à :

- partitionner les données d'entraînement en quelques sous-ensembles,
- retenir un des sous-ensembles,
- entraîner le modèle sur les sous-ensembles restants,
- effectuer les prédictions sur le sous-ensemble retenu,
- réitérer jusqu'à ce que tous les sous-ensembles aient été utilisés pour la prédiction.

En raison du surcoût en temps de calcul de cette approche (induits par les multiples entraînements), nous avons préféré adopter l'approche dite *blending*. Cette approche consiste à diviser en deux les données d'entraînement, à entraîner sur le premier et effectuer les prédictions sur le deuxième comme cela est illustré sur la figure 4.6.

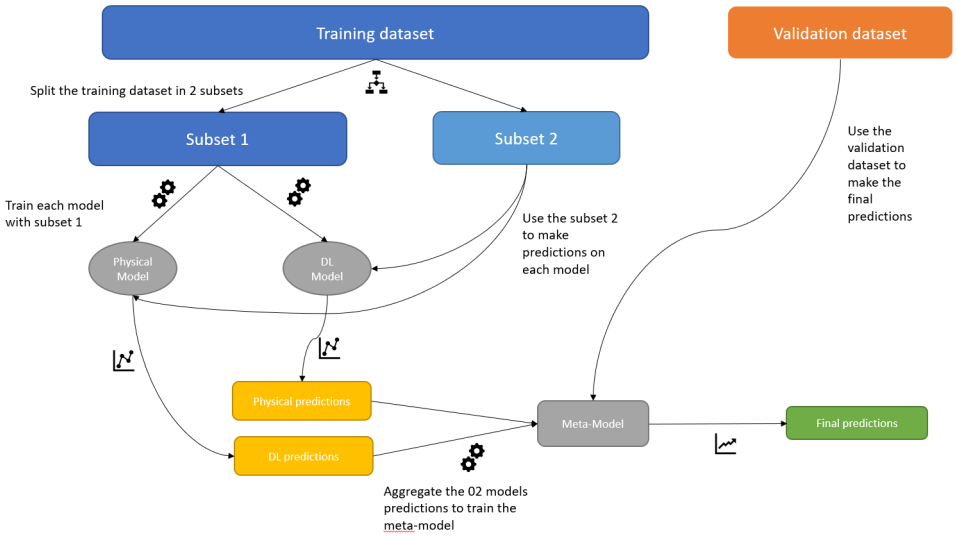


FIGURE 4.6 – Architecture de la méthode Stacked Generalization

Un point important à noter dans cette méthode est le choix de l'algorithme pour le meta-learner. Choisir un algorithme qui permet de maximiser les performances, mais qui n'est pas explicable ne nous permettrait pas de répondre à l'objectif premier de l'utilisation de la méthode d'ensemble. De ce fait, nous avons décidé de privilégier les méthodes les plus simples (et donc les plus explicables) pour le meta-learner. Nous avons ainsi implémenté une régression linéaire aux moindres carrés (à coefficients positifs), des régressions par descente de gradient stochastique ou des régressions régularisés (Ridge).

Chapitre 5

Expérimentations & Simulations

Les différentes expérimentations et tests sur les différents modèles que l'on présentera dans cette section, ont été réalisés en utilisant la base de données de la campagne 1. Les données initiales comportant beaucoup de bruit, il a été nécessaire de les filtrer. Tognan [1] mentionne dans sa thèse que les mesures lors de la campagne de vieillissement, n'ont pas toujours été effectuées dans les conditions optimales de fonctionnement (disponible en annexe A.1). Les données ont donc été filtrées afin de ne contenir que les points pris dans les conditions de fonctionnement.

D'autres méthodes de filtrages (LOWESS, Gaussian Moving Average Filter, etc.) étaient plus adaptées, mais faisaient perdre une partie de l'information que nous jugions importante, à savoir les pics observés lors des arrêts démarrages.

5.1 Modèle physique

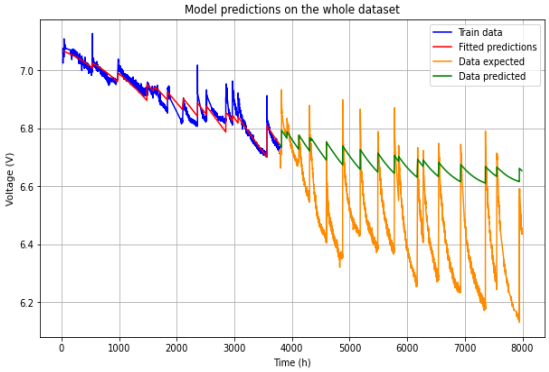
5.1.1 Expérimentations et résultats

L'implémentation de cette méthode pour paramétrer le modèle s'est fait en utilisant la base de données de la campagne 1. Nous avons utilisé 50% de la campagne pour le paramétrage ($\sim 4000h$) de l'équation. La validation du modèle s'est effectuée sur les données restantes.

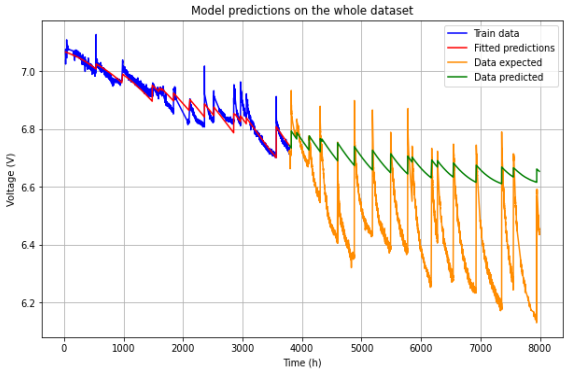
Dans le cadre de la modélisation physique, nous avons décidé de modéliser la dégradation physique dans le taux de dégradation τ_{degr} par un paramètre $k_{rev}(t)$. Dans le cadre de cette modélisation, quelques modélisations empiriques ont été choisi pour modéliser ce paramètre. Ces choix ont été retenus en observant la dégradation physique dans la tension des campagnes 3.3.

$$\begin{aligned}
\text{Modélisation linéaire : } & k_{rev}(t) = a_{k_{rev}} * t + b_{k_{rev}} \\
\text{Modélisation log-linéaire : } & k_{rev}(t) = a_{k_{rev}} log(t) + b_{k_{rev}} * t + c_{k_{rev}} \\
\text{Modélisation logarithmique : } & k_{rev}(t) = a_{k_{rev}} log(t) + b_{k_{rev}} \\
\text{Modélisation quadratique : } & k_{rev}(t) = a_{k_{rev}} * t^2 + b_{k_{rev}} * t + c_{k_{rev}}
\end{aligned}$$

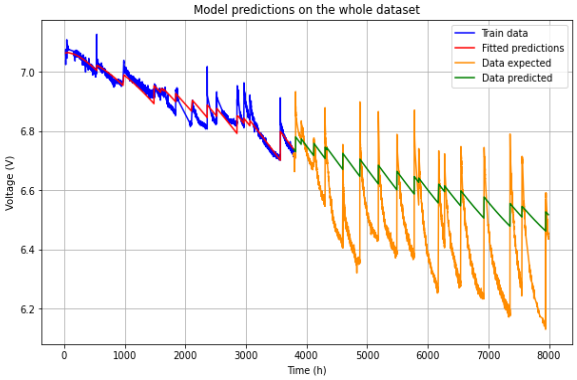
Les figures suivantes présentent respectivement les prédictions de chaque modèle sur les données d'entraînement et de validation 5.1, les erreurs absolues de prédiction sur les données de validation 5.2 ainsi qu'une comparaison des erreurs moyennes absolues de chaque modèle 5.3.



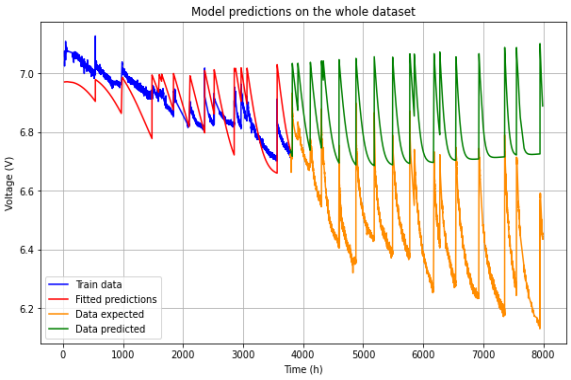
(a) linéaire



(b) log-linéaire

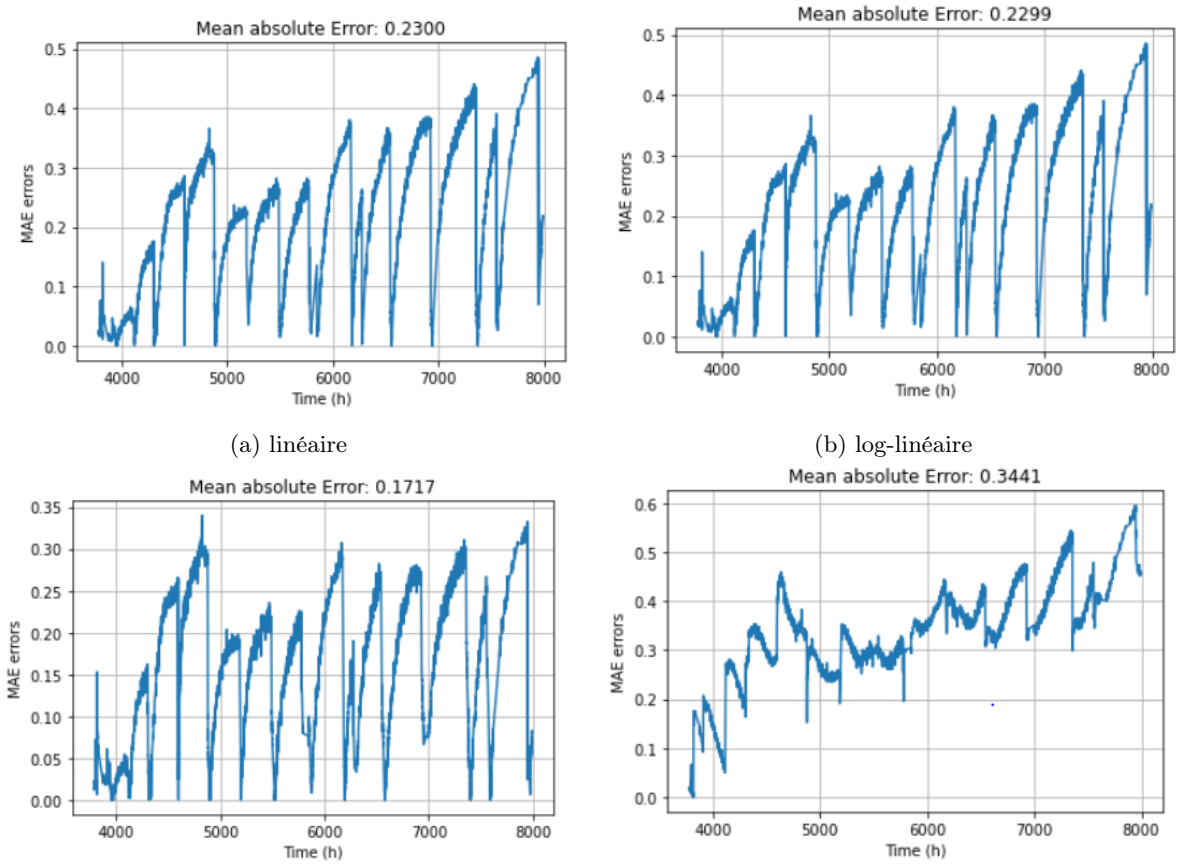


(c) logarithmique



(d) quadratique

FIGURE 5.1 – Résultats du modèle paramétré sur les données d'entraînement et les données de validation



(a) linéaire

(b) log-linéaire

(c) logarithmique

(d) quadratique

FIGURE 5.2 – Évolution des erreurs du modèle sur les données de validation

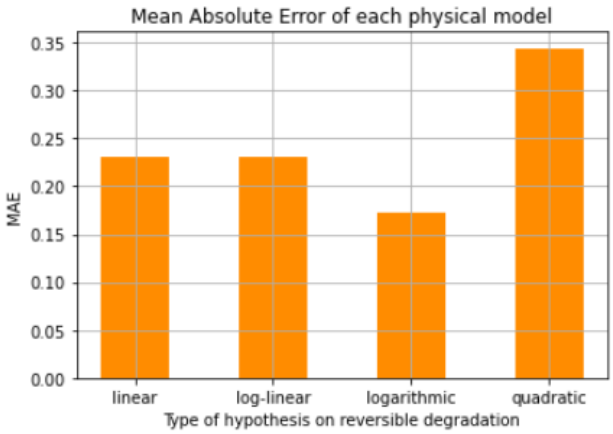


FIGURE 5.3 – Comparaison des erreurs moyennes absolues des modèles

5.1.2 Commentaires et analyses

Les valeurs des paramètres estimés par chacun des modèles sont présentées dans le tableau 5.4.

Les prédictions de chaque modèle sur les données de validation 5.1 montrent que l'hypothèse quadratique pour la dégradation réversible donne une mauvaise estimation de la tension de la pile. Cette modélisation ne permet pas au modèle physique de s'adapter aux données même pendant la phase d'apprentissage. On note que pendant les prédictions sur les données de validation, la dégradation irréversible (traduite par la décroissance de la tension) n'est pas bien estimée tandis que les pics de la dégradation

réversibles sont mieux estimés. Le modèle quadratique a des erreurs moyennes absolues plus importantes (0.3441).

Les hypothèses linéaires, log-linéaires et logarithmiques en revanche donnent de bons résultats sur la phase d'apprentissage. Concernant les prédictions du modèle sur les données de validation, on note que la dégradation irréversible est globalement mieux estimée par ces modèles. Ces modèles font cependant des estimations linéaires sur chacune des plages de fonctionnement. Sur une plage, la dégradation n'est pas correctement estimée. Ces estimations sont adaptées pour les premières plages de fonctionnement, mais s'avèrent inadaptées pour les dernières plages qui décroissent de façon exponentielle. La dégradation réversible, elle est difficilement apprise par ces modèles. Les prédictions montrent des pics de faible amplitude.

Les performances des deux modèles (linéaire et log-linéaire) sur les données de validation (0.2300 et 0.2299 respectivement) montrent que les erreurs du modèle augmentent par plage et est plus importante au niveau des pics induits par les arrêts démarrages 5.3. Les performances entre les deux modèles sont sensiblement égales. Cependant, en observant les valeurs calculées par les critères AIC et BIC 5.3, on remarque que le modèle log-linéaire obtient les valeurs de AIC et BIC les plus grandes. On peut retrouver ces valeurs dans le tableau 5.3. Le principe de parcimonie nous incite à sélectionner alors le modèle linéaire qui, en termes de simplicité de modélisation de la dégradation réversible de la tension (k_{rev}), est aussi le plus simple.

En observant cette fois-ci les performances du modèle logarithmique, on peut remarquer qu'il a de meilleures performances sur les données de validation (0.1717). Ces prédictions sur les dégradations de la pile sont meilleures que celles des modèles précédents mais présente aussi des limites en termes de dégradation réversible et irréversible sur les plages. En observant cependant, les statistiques de paramétrage de ce modèle, on peut remarquer que les valeurs de AIC et BIC de ce modèle sont moins bonnes que celles des autres. En termes de complexité, ce modèle est moins bon que le modèle linéaire, avec lequel il a le même nombre de paramètres. Il y a donc un compromis à faire entre simplicité et efficacité, pour la sélection du meilleur modèle.

Nous nous sommes aussi intéressés aux temps d'entraînement des modélisations physiques. Comme le présente le tableau 5.1, on remarque que le modèle à l'hypothèse linéaire est celui avec le temps d'entraînement le plus court et celui à l'hypothèse quadratique le plus long. Le temps d'entraînement de ces modèles est relativement court.

Modèles	Linéaire	Quadratique	Log-linéaire	Logarithmique
Durée d'entraînement (s)	28.30	43.15	33.83	41.60

TABLE 5.1 – Durée d'entraînement des modèles physiques

Ces calculs ont été effectués avec une taille des données d'entraînement égale à 50% des données totale. On s'est aussi alors demandé si la taille des données avait une influence sur le temps d'entraînement. Contrairement à ce que l'on pouvait penser, le temps d'entraînement ne dépend pas de la taille des données. A titre illustratif nous présentons les différents temps d'entraînement en utilisant le modèle avec hypothèse logarithmique dans le tableau 5.2.

Taille des données	10%	20%	30%	40%	50%	60%	70%
Durée d'entraînement (s)	55.85	10.99	76.38	21.82	40.24	87.21	84.94

TABLE 5.2 – Évolution du temps d'entraînement du modèle physique avec la taille des données

Afin d'avoir une meilleure idée des performances des modèles, nous avons étudié aussi si les performances des modèles dépendaient de la taille des données utilisées pour paramétrer le modèle. Les modèles ainsi entraînés ont été validés sur les données restantes. Comme le montre la figure 5.4, les performances des modèles varient en fonction de la proportion des données utilisées pour l'entraînement. Ainsi, on peut voir que les modèles ont de moins bonne performance avec 20% des données totales que 10% ou de moins bonnes performances avec 50% qu'avec 30%. Il est intéressant de remarquer que le modèle logarithmique est le plus souvent, meilleur au modèle linéaire en termes de performance.

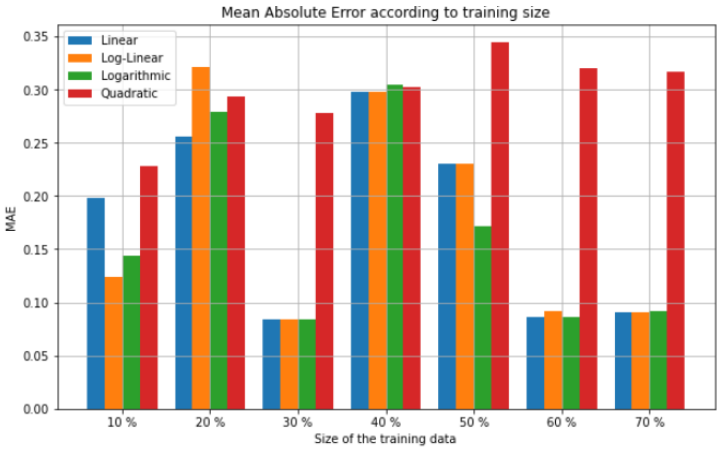
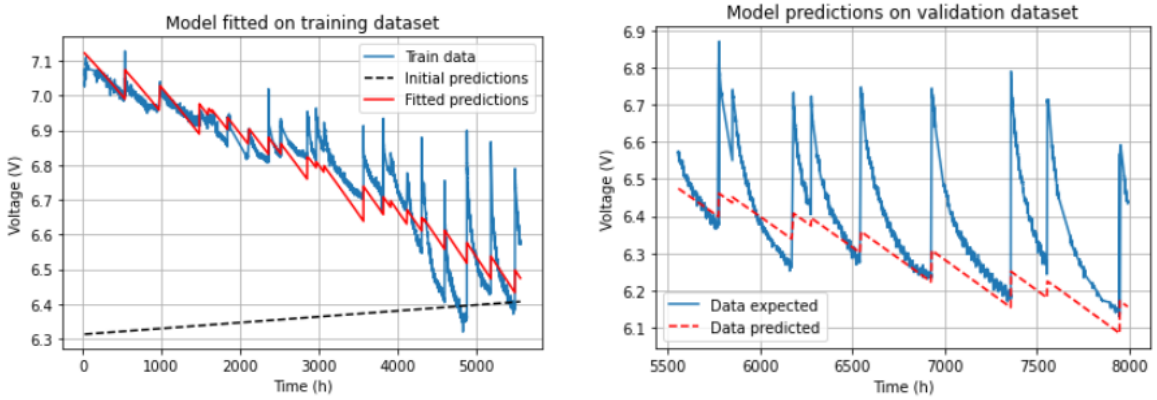


FIGURE 5.4 – Comparaison des erreurs moyennes absolues en fonction de la taille des données d'entraînement

On pourrait penser aussi qu'en augmentant les tailles de données d'entraînement, le modèle aurait ainsi l'opportunité de mieux apprendre la dégradation réversible (qui est plus marquée sur la fin de la campagne). Cependant, les résultats en augmentant par exemple la taille des données à 70%, sont moins bons que ceux obtenus avec 50% pour l'estimation de la dégradation. Le modèle n'arrive pas à bien apprendre la dégradation réversible (même pendant la phase d'entraînement). La figure suivante montre les performances du modèle logarithmique avec 70% des données utilisées pour l'entraînement.



(a) Apprentissage (b) Validation

FIGURE 5.5 – Prédiction du modèle logarithmique avec 70% des données pour l'entraînement

Statistiques	Modèle linéaire	Modèle quadratique	Modèle log-linéaire	Modèle logarithmique
Nombres d'itérations	25	35	23	28
χ-square	372.90	4860.76	372.90	389.38
Reduced χ-square	$4.09.10^{-4}$	$5.34.10^{-3}$	$4.09.10^{-4}$	$4.28.10^{-4}$
AIC	-7093899.73	-4758515.89	-7093897.73	-7054579.57
BIC	-7093829.41	-4758433.84	-7093815.69	-7054509.25

TABLE 5.3 – Statistiques de paramétrage des modèles

Paramètres		Modèle linéaire	Modèle quadratique	Modèle log-linéaire	Modèle logarithmique
$J_o(t_0)$		$3.17.10^{-7}$	$1.63.10^{-7}$	$3.17.10^{-7}$	$4.43.10^{-7}$
$R^A_{diff}(t_0)$		$5.77.10^{-3}$	10^{-10}	$5.77.10^{-3}$	$1.44.10^{-2}$
$R^B_{diff}(t_0)$		$2.59.10^{-5}$	10^{-10}	$5.17.10^{-6}$	$3.14.10^{-8}$
k_{irrev}		$4.16.10^{-4}$	$1.63.10^{-7}$	$4.16.10^{-4}$	$2.88.10^{-4}$
k_{rev}	$a_{k_{rev}}$	$3.88.10^{-7}$	$3.12.10^{-10}$	$5.68.10^{-15}$	$2.76.10^{-4}$
	$b_{k_{rev}}$	$-6.92.10^{-5}$	$1.52.10^{-6}$	$3.88.10^{-7}$	$-1.63.10^{-3}$
	$c_{k_{rev}}$		$-1.39.10^{-4}$	$-6.92.10^{-5}$	

TABLE 5.4 – Paramètres estimés des modèles

5.2 Réseaux de neurones

5.2.1 Prétraitement et mise en forme des données

Pour réaliser un entraînement efficace des réseaux de neurones, il est important de normaliser les données avant de les passer aux réseaux. Cette étape de normalisation permet à l'architecture de réseaux de neurones, de ne pas être affecté par les échelles des données d'entrées. Elle permet aussi d'éviter que les sorties du réseau ne soient saturantes et que cela entraîne le problème des *vanishing gradients*. Ainsi, chaque architecture que nous implémenterons sera précédée d'une couche de prétraitement de données qui normalise les données. Cette couche réalise l'opération suivante pour chaque entrée x du réseau :

$$x' = \frac{x - \mu}{\sigma} \tag{5.1}$$

où $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ et $\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}$ désignent respectivement la moyenne et la variance empirique des données d'entraînement.

Pour réaliser l'entraînement des réseaux de neurones, nous avons utilisé la méthode des fenêtres glissantes (*Sliding Moving Method*). Cette méthode consiste à décomposer les données d'entraînements en segments de données (ou fenêtres) d'une taille déterminée $n < N$, où N désigne la taille des données. Le fonctionnement de cette méthode est décrit dans la figure 5.6. On accumule les n données temporelles de la fenêtre 1 pour effectuer la prédiction des n données suivantes. On décale ensuite la fenêtre de la taille de l'offset et on répète le processus jusqu'à ce que toutes les données d'entraînement aient été parcourues.

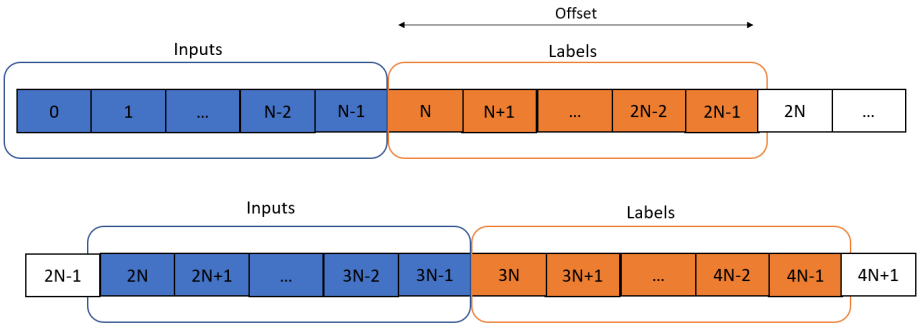


FIGURE 5.6 – Méthode des fenêtres glissantes

5.2.2 Environnement technique

L'entraînement des réseaux de neurones a été effectué avec Python 3.6.1 sur l'outil Anaconda. Les architectures de réseaux de neurones ont été développées avec la librairie TensorFlow 2.5.0 [37] développée par Google.

L'optimiseur utilisé pour l'entraînement est l'algorithme Adam, qui est une méthode de descente de gradient stochastique basée sur l'estimation adaptative des moments du premier et du second ordre.

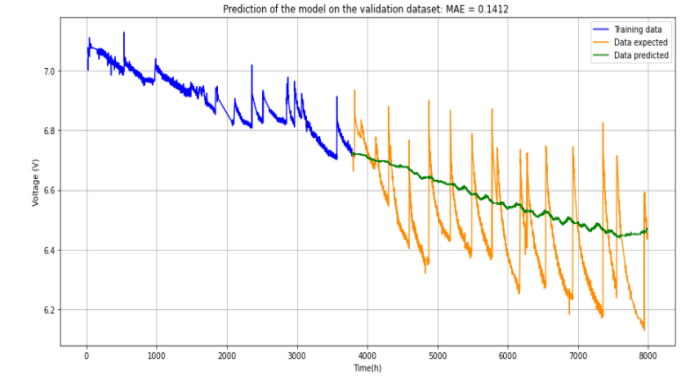
Le nombre maximum d'époques utilisé est de 20, mais en vue d'améliorer l'entraînement et éviter le sur-apprentissage, nous avons décidé d'appliquer la méthode de l'arrêt anticipé (*Early Stopping*) si l'erreur sur les données de validation ne diminue pas sur 2 époques consécutives. Le ratio données d'entraînement - données de validation est de 50-50. La taille de la fenêtre d'apprentissage est N=100000.

L'entraînement des architectures LSTM nécessitent d'importantes ressources de calcul (notamment du GPU). L'utilisation du GPU (Graphics Processing Units) permet d'accélérer l'entraînement de ces architectures et aussi de réduire leur temps d'inférence. De ce fait, l'environnement AWS (Amazon Web Services) a été utilisé.

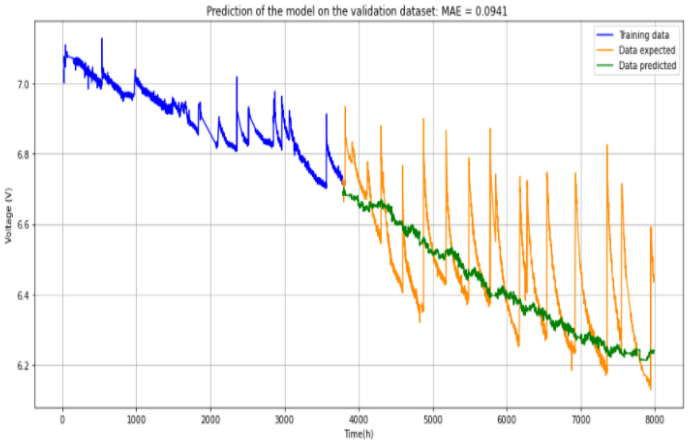
AWS est une plateforme du groupe Amazon spécialisée dans les services de cloud computing à la demande. Elle permet d'avoir accès à des ressources de calcul virtuellement illimitées pour effectuer des calculs très gourmands en ressources, à l'occurrence des entraînements de modèles d'apprentissage automatique. Nous avons ainsi utilisé une instance EC2 (Elastic Compute Cloud) *p2.xlarge* qui possède une RAM de 61 Go, de 4 CPU (Central Processing Units) et un GPU de 4 Go.

5.2.3 Expérimentations et résultats

La figure 5.7 illustre les performances des modèles linéaires et convolutifs implémentés sur les données de validation.



(a) linéaire

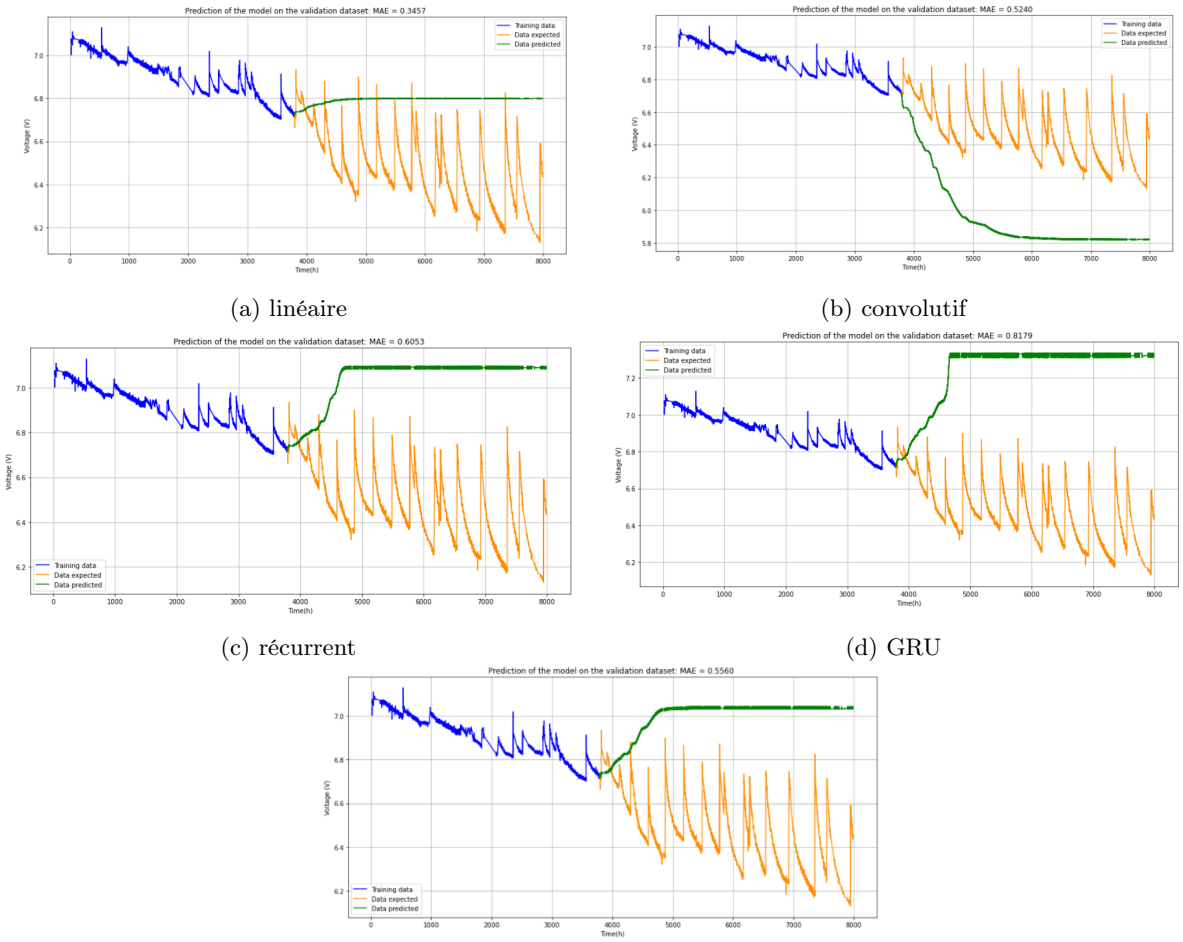


(b) convolutif

FIGURE 5.7 – Prédictions des modèles sur les données de validation

La taille de la fenêtre d'apprentissage étant très élevée, il nous était difficile d'entraîner le modèle LSTM dans des temps raisonnables ($> 3-5h$ pour une époque). D'autres approches plus simplifiées des réseaux de neurones récurrents, à savoir des RNNs simples ou des GRU ont aussi été implémentés. Mais le problème des temps d'entraînement se posaient aussi. Nous avons donc décidé d'implémenter ces modèles avec une taille de fenêtre d'apprentissage réduite $N = 1000$. Des modèles convolutifs et linéaires ont été

aussi entraînés avec cette même taille de fenêtre afin de mieux comparer les modèles. Ces résultats sont illustrés avec la figure 5.8.



(a) linéaire

(b) convolutif

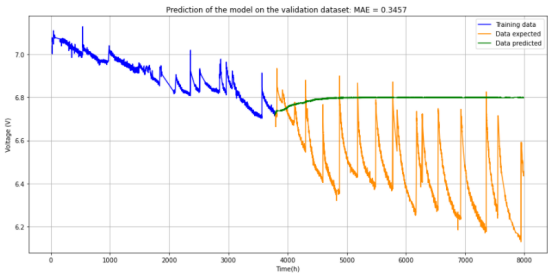
(c) récurrent

(d) GRU

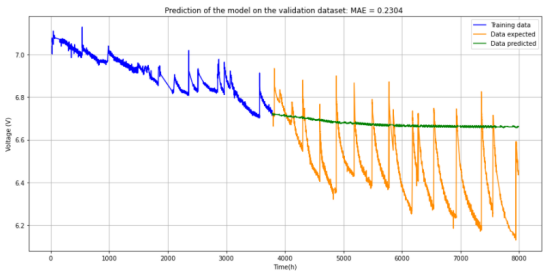
(e) LSTM

FIGURE 5.8 – Prédictions des modèles avec une fenêtre de taille $N = 1000$

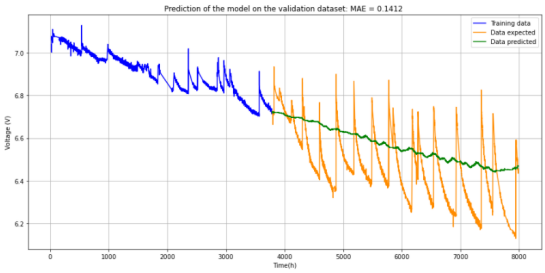
Nous nous sommes aussi intéressés aux performances du modèle en fonction de la taille de la fenêtre d'apprentissage. Pour illustrer l'influence de ce paramètre, seul le modèle linéaire a été utilisé en raison du temps d'entraînement de ce modèle qui est moins important que celui des autres modèles. La figure 5.9 illustre ces résultats.



(a) $N = 1000$



(b) $N = 10000$



(c) $N = 100000$

FIGURE 5.9 – Influence de la taille de la fenêtre d'apprentissage sur les performances du modèle linéaire

5.2.4 Commentaires et analyses

Les résultats de la figure 5.7 montrent que le modèle linéaire a de moins bonnes performances sur les données de validation que le modèle convolutif. La dégradation irréversible est difficilement prédite avec ce modèle. Le modèle convolutif parvient à mieux prédire la dégradation irréversible du modèle et a de meilleures performances. Les dégradations réversibles ne sont pas du tout estimées avec le modèle linéaire et le modèle convolutif. Ce qui constitue un désavantage par rapport au modèle physique. Les différents pics induits par les phases d'arrêts démarrages ne sont pas appris par les réseaux de neurones. Cela s'explique par le fait que les données soient normalisées avant l'entraînement du réseau de neurones et de ce fait, les pics et les bruits sont ignorés.

L'analyse de la figure 5.8 illustrant les performances de différents modèles pour une taille de fenêtre $N = 1000$ montrent que les modèles n'arrivent pas à se généraliser sur les données de validation. La taille de la fenêtre ne permet pas aux modèles de bien apprendre la tendance de dégradation de la tension, même pour des modèles efficaces sur les longues séquences temporelles comme les LSTM ou les GRU. Les difficultés d'entraînement de ces architectures ne nous permettent pas de vérifier l'efficacité de cette approche sur notre problème.

Nous présentons aussi les temps d'entraînement de ces différents modèles. Le tableau montre que la durée d'entraînement est relativement importante comparativement aux modèles physiques. Notons le temps d'entraînement des modèles RNNs, GRU et LSTM malgré la taille de la fenêtre ($N = 1000$). Ces temps d'entraînement très élevé ont rendu difficiles les expérimentations sur les modèles récurrents qui devront être menées dans d'autres travaux.

Modèle	Linéaire	Convolutif	RNN	GRU	LSTM
Taille des fenêtres	$N = 100000$		$N = 1000$		
Durée d'entraînement	48mn	2h20mn	5h	6h20mn	4h15mn

TABLE 5.5 – Temps d'entraînement des modèles d'apprentissage

Il est à noter que contrairement à ce que l'on pouvait attendre à l'issue de l'état de l'art, le modèle physique est celui avec les temps d'entraînement les plus faibles comparativement aux temps d'entraînement des modèles d'apprentissage statistiques qui sont beaucoup plus importants.

Pour ce qui est de l'influence de la taille des fenêtres sur les performances, la figure montre que le modèle est plus robuste si la taille des fenêtres est plus importante ($N = 100000$). Le modèle arrive, avec des fenêtres plus importantes, à mieux apprendre l'évolution de la dégradation, contrairement à lorsque la taille est plus petite ($N = 1000$). Le modèle est victime de sur-apprentissage lorsque la taille de la fenêtre est petite, ce qui l'empêche de bien se généraliser aux données inconnues. Cependant il est à noter que plus la taille de la fenêtre est importante, plus important sera le temps de calcul. Ce temps de calcul sera d'autant plus grand que le modèle sera plus difficile à entraîner, comme c'est le cas avec les LSTM.

5.3 Modèle hybride

5.3.1 Expérimentations et résultats

Les prédictions sur le modèle hybride ont été réalisées en utilisant le modèle physique sous hypothèse logarithmique et le modèle d'apprentissage convolutif. Ces deux modèles ont été retenus en raison de leur meilleure performance sur les données de validation.

La figure suivante présente tout d'abord les performances du modèle hybride en utilisant la méthode d'ensemble du vote à majorité 5.10.

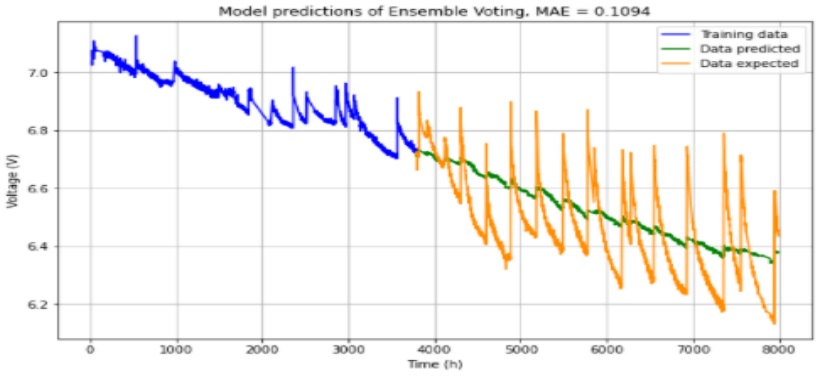


FIGURE 5.10 – Prédictions de la méthode Ensemble Voting

Les performances de la méthode Weigted Average Model sont illustrées sur la figure 5.11.

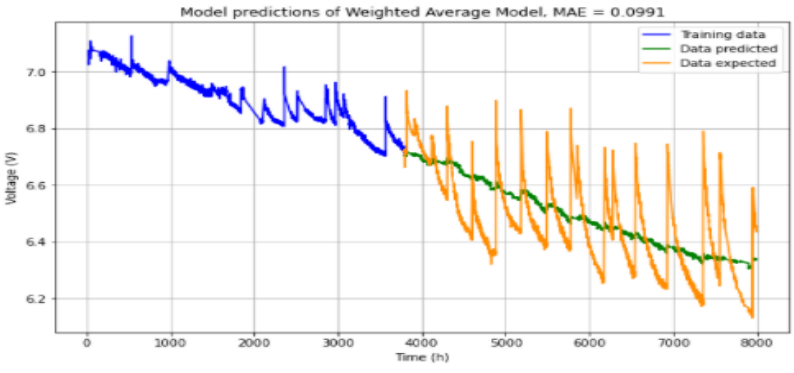


FIGURE 5.11 – Prédictions de la méthode Weighted Average Model

Les poids qui ont été attribués au modèle sont illustré par le tableau suivant :

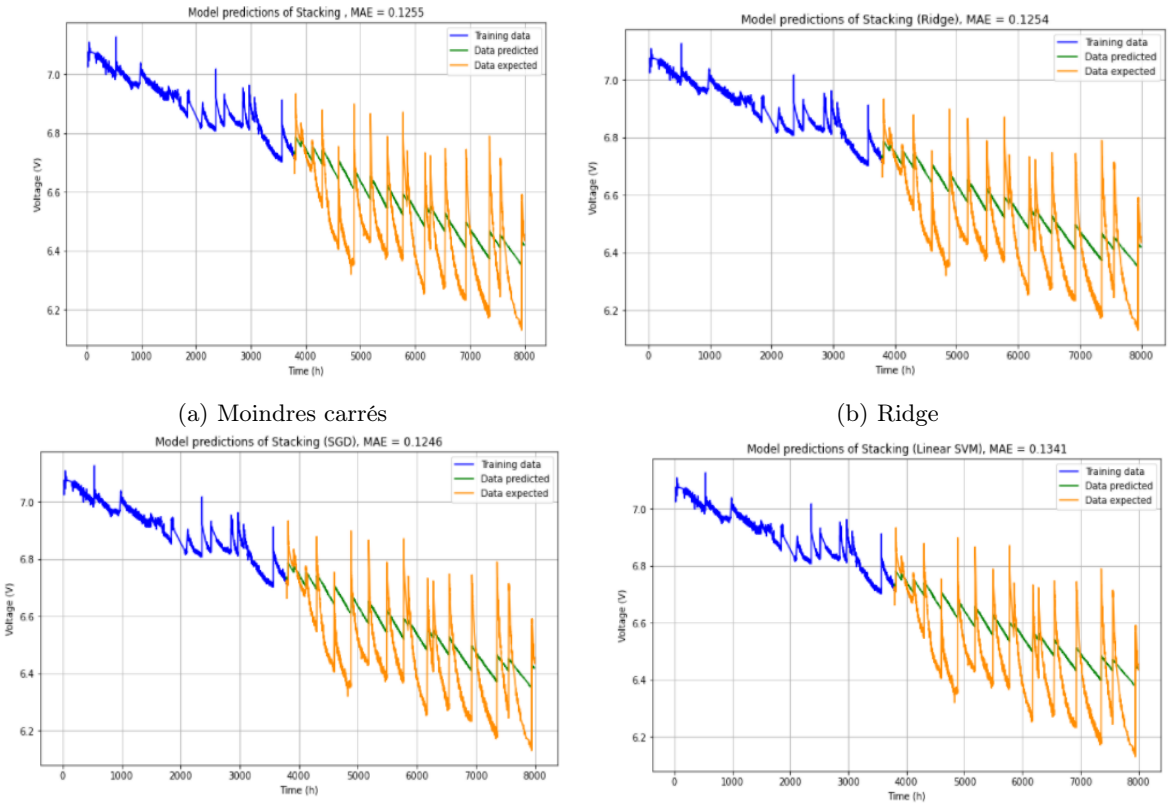
	Modèle physique	Modèle LSTM
Poids	0.35	0.65

TABLE 5.6 – Poids pour la méthode Weighted Average

La méthode Stacking fut implémentée en utilisant 4 modèles différents comme meta-learner :

- Régression linéaire (approche aux moindres carrés)
- Régression Ridge (régularisée)
- Régression avec descente de gradient stochastique
- SVM (avec hypothèse linéaire)

Les prédictions sur les données de validation sont illustrées sur la figure 5.12.



(a) Moindres carrés

(b) Ridge

(c) Descente de gradient stochastique

(d) SVM

FIGURE 5.12 – Prédictions de la méthode Stacking

5.3.2 Commentaires et analyses

L'analyse des performances des méthodes hybrides implémentées montrent que les performances ne diffèrent pas de celles obtenues avec les modèles d'apprentissage ou physiques. Les modèles ont des estimations moyennes de la dégradation réversible et de la dégradation irréversible par plages de fonctionnement. En termes d'erreurs moyennes absolues, les meilleures performances sont atteintes avec la méthode Weighted Average (0.0991). Cette méthode qui accorde beaucoup de poids au modèle d'apprentissage automatique par rapport au modèle physique (65% contre 35%) permet d'avoir des performances meilleures à celles obtenues avec le modèle physique mais légèrement en dessous de celles obtenues avec le modèle convolutif. On obtient ainsi un modèle avec des prédictions guidées par la modélisation physique. Nous nous sommes aussi intéressés à l'influence des poids sur les performances de la prédiction de la méthode Weighted Average. Nous avons donc fait varier les poids des modèles physiques et d'apprentissage. Les résultats sont illustrés dans la figure ci-dessous.

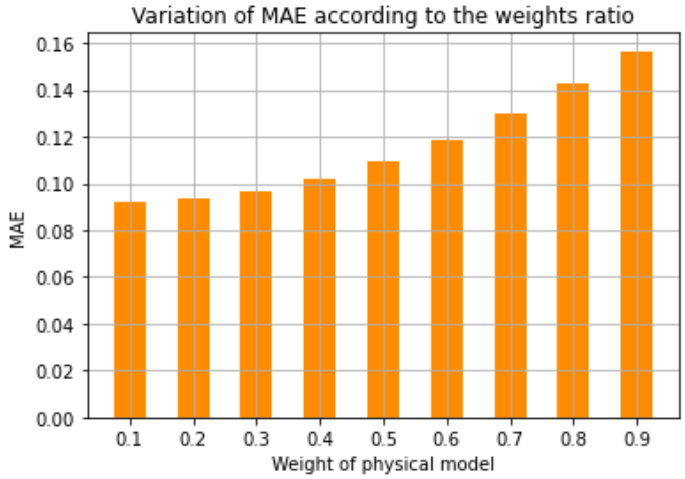


FIGURE 5.13 – Influence des poids sur les performances de la méthode Weighted Average

On peut remarquer dans la figure 5.13 que le modèle est moins performant si le poids associé au modèle physique augmente. Ce résultat n'a rien d'étonnant étant donné que le modèle physique a de moins bonnes performances que le modèle d'apprentissage automatique. Les résultats que nous avons obtenus avec la méthode Weighted Average ne sont pas optimaux en termes de performance permet d'avoir un bon compromis entre modélisation physique et apprentissage automatique.

La méthode Ensemble Voting a des performances légèrement en dessous de la méthode Weighted Average (0.1094) mais a de meilleures performances que le modèle physique. Les performances sont en dessous de celle du modèle d'apprentissage, ce qui est étonnant car la méthode Ensemble Voting permet d'avoir des performances meilleures à chaque meta-learner pris individuellement.

La méthode Stacking a des performances assez variables en fonction de l'algorithme utilisé comme meta-learner. On obtient des performances autour de 0.12. Les algorithmes linéaires permettent d'avoir de bonnes performances mais les estimations sont linéaires sur les différentes plages de fonctionnement. Plus le meta-learner devient complexe, plus les prédictions se détériorent comme c'est le cas avec l'algorithme SVM utilisé comme meta-learner. Utiliser un modèle plus complexe comme des forêts aléatoires ou du Gradient Boosting, revient aussi à rajouter une nouvelle boite noire au modèle et donc perdre en explicabilité. L'objectif de cette méthode hybride était de pouvoir tirer parti de la méthode physique notamment pour l'estimation des dégradations réversibles (pics). On peut noter que cet objectif est difficilement atteint, étant donné que le modèle physique n'estime pas correctement les pics.

Concernant les temps d'entraînement de la méthode hybride, le temps est borné par le temps qu'il faudra pour entraîner le modèle "le plus lent", en l'occurrence dans notre étude, le modèle d'apprentissage automatique. L'hybridation ne permet pas d'obtenir un compromis sur les durées d'entraînement.

La figure ci-après récapitule les performances de chacun des modèles implémentés.

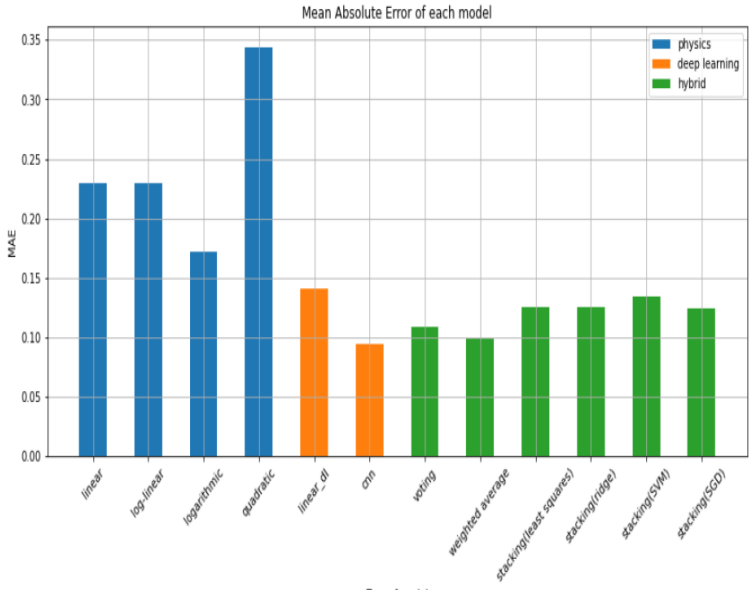


FIGURE 5.14 – Comparaison des performances de tous les modèles

Chapitre 6

Conclusion

6.1 Bilan

La problématique qui a guidé les travaux de ce stage, était l'étude du vieillissement des piles à combustible à partir de la littérature et d'une base de données expérimentales. La revue bibliographique a montré la complexité à modéliser les phénomènes physiques qui influent sur le vieillissement des piles. Les méthodes d'apprentissage et même hybrides ne traitent pas des principaux aspects du vieillissement ou se basent sur la tension de la pile qui ne traduit pas complètement les dégradations au sein de la pile. Ces méthodes sont en plus difficiles à expliquer en raison de leur aspect "boîte noire" ou de l'absence de modélisation physique consistante.

L'analyse de la base de données expérimentales a montré que la majeure partie des grandeurs présentes n'expliquent pas la dégradation des piles malgré des corrélations importantes. Les grandeurs les plus caractéristiques du fonctionnement, à savoir les voltamétries, courbes de polarisation, restent difficiles à obtenir quoique nécessaires pour une meilleure analyse.

Les résultats obtenus dans les travaux de modélisations du vieillissement des piles à combustible sont encourageants sur les possibilités envisageables. Le modèle physique arrive, malgré les nombreuses hypothèses et simplifications, à fournir une bonne estimation de l'évolution de la tension de la pile, notamment des pics induits par les phénomènes réversibles de la dégradation.

Les techniques d'apprentissage automatique qui ont été implémentées dans ce stage, quoique basiques, ont fourni de bons résultats sur l'estimation de la dégradation irréversible de la pile. Ces méthodes ne sont toutefois pas adaptées pour estimer les pics, qui sont des informations importantes dans l'étude du vieillissement des piles. Les méthodes d'apprentissage automatique, plus adaptées pour les séries temporelles longues, n'ont pas pu être testées, en raison de la difficulté d'effectuer un entraînement efficace sur les unités de calcul.

L'approche hybride implantée nous a permis d'avoir des performances meilleures à celles que l'on pouvait obtenir avec le modèle physique. Cette hybridation de modèles est passée par les méthodes d'ensemble connues pour leurs efficacité dans le domaine de l'apprentissage automatique. L'utilisation des méthodes d'ensemble nous permet d'avoir un modèle qui réalise un compromis entre performance et explicabilité. L'utilisation de la physique avec des approches d'apprentissage automatique, nous offre ainsi un modèle dit "gray-box", dont les prédictions sont en partie explicables grâce au modèle physique.

6.2 Perspectives

Les perspectives à l'issue de ces travaux sont nombreuses.

Tout d'abord, l'un des principaux freins rencontrés lors de cette étude était la base de données. Malgré une base de données très importante, une bonne partie de ces données n'étaient pas exploitables pour nos recherches. Les corrélations de Chatterjee trouvées entre les grandeurs de la base de données ne traduisaient pas forcément de la dégradation de la pile. Par contre, les travaux de modélisation de Robin et Kneer [25, 27], montrent une grande importance de la perte de surface active dans la dégradation de la pile. Une autre donnée importante qui manquait à cette base de données est la mesure de la résistance ohmique de la pile qui intervient dans la dégradation (voir C.1). Il serait alors intéressant d'envisager, lors des prochaines campagnes expérimentales, des mesures sur la perte de surface active de la pile et les résistances à l'aide de voltamétries cycliques et de spectroscopies d'impédance. Les courbes de polarisation

sont aussi des informations qu'il serait important d'avoir afin de pouvoir avoir une meilleure modélisation des phénomènes physico-chimiques de dégradation dans la pile.

Ensuite, les modèles de modélisation data-driven qui ont été utilisés au cours de ces travaux étaient assez basiques. La revue bibliographique à montrer qu'il existe de nombreuses approches qu'il serait intéressant d'explorer. Nous n'avons pas réussi à tester les performances des réseaux de neurones récurrents (LSTM, RNN etc) sur la modélisation du vieillissement des piles. Il existe aussi de nombreuses améliorations de ces architectures LSTM de base qui permettent un meilleur traitement des séries temporelles. De futurs travaux sont envisageables pour améliorer les performances des approches d'apprentissage automatique.

Les méthodes de décomposition temporelle et fréquentielle à l'aide des ondelettes ou des transformées de Fourier sont autant d'astuces qui couplées aux réseaux de neurones permettent de les enrichir d'informations sur l'évolution temporelle des séries. Les méthodes bayésiennes ainsi que les processus gaussiens sont aussi des approches ayant de bonnes performances sur les séries temporelles, qu'il serait intéressant de tester, notamment grâce à l'avantage que présentent ces méthodes de réussir à fournir une quantification de l'incertitude sur la prédiction.

L'hybridation par des méthodes d'ensemble, en vue d'atteindre un modèle explicable reste une idée intéressante mais pas la seule méthode pour pouvoir obtenir des modèles explicables. L'une des options que nous n'avons pas pu explorer pendant ce stage c'est l'utilisation de fonction objectif pénalisée afin de respecter les contraintes physiques. Il existe aussi des approches que l'on peut envisager qui consiste à se baser sur l'utilisation des caractéristiques issues de modélisation physique pour entraîner les modèles d'apprentissage statistique. Il est aussi possible d'utiliser l'apprentissage automatique pour paramétrer les modèles physiques. Cette approche pourrait être envisager dans nos travaux pour la modélisation de la dégradation réversible qui n'est pas parfaitement modélisée dans notre modèle physique.

L'analyse de sensibilité [38] (Sensitivity Analysis) qui consiste à analyser l'influence des entrées sur les sorties des modèles, ou les méthodes de projection d'entropie [39] sont aussi des approches pouvant permettre d'obtenir des modèles explicables mais aussi généralisables sur des données différentes. Ces approches permettraient de connaître quels sont les paramètres les plus importants dans la prédiction de la dégradation de la pile.

Nous n'avons pas eu l'occasion de tester nos modèles sur les piles des autres campagnes. Ces campagnes qui sont assez variables de par leur durée, le type de piles utilisé (Campagne 3) ou le nombre de cellules dans la pile (Campagne 4), nous permettraient de vérifier la généralisabilité de nos méthodes.

Un autre point à explorer serait le choix d'une métrique mieux adaptée au vieillissement des piles à combustible. Dans notre étude, nous avons fait le choix de l'erreur moyenne absolue qui est plus robuste que l'erreur moyenne au carrée. Cependant, avec cette métrique il nous arrive de voir des modèles avec de bonnes performances moyennes, malgré le fait que les pics dus aux arrêts démarrages soient mal ou pas du tout estimés. L'estimation de ces pics, qui sont des points d'intérêts afin de savoir quel sera le regain en tension de la pile après une pause ou un arrêt de fonctionnement, est très importante. Des travaux sont à envisager afin de fournir une métrique mieux adaptée, permettant par exemple de mieux caractériser les performances d'un modèle sur l'estimation de la dégradation (pics) et/ou une fonction objectif pour guider l'apprentissage afin de pénaliser les modèles qui n'arrivent pas à bien estimer les pics.

6.3 Contributions personnelles du stage

Nous résumons dans cette section les principales contributions de ce stage. Nous avons tout d'abord recherché à partir de l'état de l'art quelles sont les grandeurs qui influent sur le vieillissement de la pile et quelles sont les procédures expérimentales à mener sur de prochaines campagnes afin de les obtenir.

Nous avons aussi proposé une méthode d'hybridation des approches physiques et d'apprentissage automatique par les méthodes d'ensemble en vue d'obtenir un compromis entre les deux approches. Les codes d'entraînement des différentes architectures pour l'entraînement sur le cloud ou un environnement local ont été fournis et un extrait est fourni en annexes D.

6.4 Retour d'expérience

J'ai pris beaucoup de plaisir à passer les 5 derniers mois de stage, sur sujet avec Vitesco Technologies et les différents collaborateurs académiques. Ce stage est pour moi le point d'entrée qu'il me fallait pour le monde de la recherche et surtout pour me lancer dans une thèse. J'ai beaucoup appris au cours de ces

derniers mois, notamment à travailler de façon autonome, à lire et faire la synthèse de nombreux articles scientifiques sur un domaine où je n'avais aucune connaissance a priori.

Ce stage m'a permis de voir comment fonctionne l'entité R&D d'une grande entreprise et les différentes collaborations entre entités d'une même entreprise mais aussi avec des chercheurs et industriels extérieurs.

Ce fut pour moi une expérience assez nouvelle de devoir faire beaucoup d'auto-formation sur le thème des technologies à base d'hydrogène et surtout d'être confronté aux défis que présentent les techniques d'apprentissage dans le monde de l'industrie. J'ai été amené à échanger avec de nombreux ingénieurs et chercheurs sur la technologie "pile à combustible", les statistiques, l'apprentissage automatique, la maintenance prédictive ou sur l'utilisation de la physique dans le machine learning. Ce sont des discussions qui furent très enrichissantes non seulement pour le stage, mais aussi pour ma culture personnelle.

J'ai enrichi ma culture et ma connaissance sur l'Intelligence artificielle hybride et interprétable, notamment sur les méthodes alliant connaissance physique et intelligence artificielle pour les diagnostics et les pronostics. Je suis d'autant plus motivé à pousser encore plus loin mes connaissances dans ce domaine, que je trouve fortement lié à l'industrie 4.0 et aux différents défis auxquels l'industrie est aujourd'hui confrontée.

Bibliographie

[1] Malik Tognan. *Etude de dégradations des performances de Piles à Combustible PEM BT alimentées en H2/O2 lors de campagnes d'endurance : du suivi de l'état de santé en opération à la modélisation du vieillissement.* PhD thesis, 2018. Thèse de doctorat dirigée par Turpin, Christophe Génie Électrique Toulouse, INPT 2018.

[2] Sébastien Wasterlain. *Approches expérimentales et analyse probabiliste pour le diagnostic de piles à combustible de type PEM.* Theses, Université de Franche-Comté, February 2010.

[3] Malik Tognan, Christophe Turpin, Olivier Rallières, Olivier Verdu, Karine Lombard, and André Rakotondrainibe. Analyse et modélisation du vieillissement d'un stack de Pile à Combustible PEM H2/O2. In *Symposium de Genie Electrique*, Grenoble, France, June 2016.

[4] Marine Jouin, Rafael Gouriveau, Daniel Hissel, Marie Cécile Péra, and Noureddine Zerhouni. Pemfc aging modeling for prognostics and health assessment. *IFAC-PapersOnLine*, 48(21) :790–795, 2015. 9th IFAC Symposium on Fault Detection, Supervision andSafety for Technical Processes SAFEPROCESS 2015.

[5] Marine Jouin, Rafael Gouriveau, Daniel Hissel, Marie-Cécile Péra, and Noureddine Zerhouni. Prognostics of pem fuel cell in a particle filtering framework. *International Journal of Hydrogen Energy*, 39(1) :481–494, 2014.

[6] D. Zhang, P. Baraldi, C. Cadet, N. Yousfi-Steiner, C. Bérenguer, and E. Zio. An ensemble of models for integrating dependent sources of information for the prognosis of the remaining useful life of proton exchange membrane fuel cells. *Mechanical Systems and Signal Processing*, 124 :479–501, 2019.

[7] L. Vichard, F. Harel, A. Ravey, P. Venet, and D. Hissel. Degradation prediction of pem fuel cell based on artificial intelligence. *International Journal of Hydrogen Energy*, 45(29) :14953–14963, 2020.

[8] Jiawei Liu, Qi Li, Weirong Chen, Yu Yan, Yibin Qiu, and Taiqiang Cao. Remaining useful life prediction of pemfc based on long short-term memory recurrent neural networks. *International Journal of Hydrogen Energy*, 44, 10 2018.

[9] Jian Zuo, Hong Lv, Daming Zhou, Qiong Xue, Liming Jin, Wei Zhou, Daijun Yang, and Cunman Zhang. Deep learning based prognostic framework towards proton exchange membrane fuel cell for automotive application. *Applied Energy*, 281 :115937, 2021.

[10] Jiawei Liu, Qi Li, Ying Han, Guorui Zhang, Xiang Meng, Jiaxi Yu, and Weirong Chen. Pemfc residual life prediction using sparse autoencoder-based deep neural network. *IEEE Transactions on Transportation Electrification*, 5(4) :1279–1293, 2019.

[11] Kui Chen, Salah Laghrouche, and Abdesslem Djerdir. Aging prognosis model of proton exchange membrane fuel cell in different operating conditions. *International Journal of Hydrogen Energy*, 45(20) :11761–11772, 2020.

[12] Yucen Xie, Jianxiao Zou, Chao Peng, Yun Zhu, and Fei Gao. A novel pem fuel cell remaining useful life prediction method based on singular spectrum analysis and deep gaussian processes. *International Journal of Hydrogen Energy*, 45(55) :30942–30956, 2020.

[13] Yujie Cheng, Noureddine Zerhouni, and Chen Lu. A hybrid remaining useful life prognostic method for proton exchange membrane fuel cell. *International Journal of Hydrogen Energy*, 43(27) :12314–12327, 2018.

[14] Jiayu Chen, Dong Zhou, Chuan Lyu, and Chen Lu. A novel health indicator for pemfc state of health estimation and remaining useful life prediction. *International Journal of Hydrogen Energy*, 42(31) :20230–20238, 2017.

[15] Daming Zhou, Fei Gao, Elena Breaz, Alexandre Ravey, and Abdellatif Miraoui. Degradation prediction of pem fuel cell using a moving window based hybrid prognostic approach. *Energy*, 138 :1175–1186, 2017.

[16] J.-S.R. Jang. Anfis : adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3) :665–685, 1993.

[17] Renyou Xie, Rui Ma, Sicheng Pu, Liangcai Xu, Dongdong Zhao, and Yigeng Huangfu. Prognostic for fuel cell based on particle filter and recurrent neural network fusion structure. *Energy and AI*, 2 :100017, 07 2020.

[18] Fu-Kwun Wang, Chang-Yi Huang, Tadele Mamo, and Xiao-Bin Cheng. Ensemble model for the degradation prediction of proton exchange membrane fuel cell stacks. *Quality and Reliability Engineering International*, 37(1) :34–46, 2021.

[19] Anuj Karpatne, Gowtham Atluri, James H. Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science : A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10) :2318–2331, Oct 2017.

[20] Timur Bikmukhametov and Johannes Jäschke. Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models. *Computers & Chemical Engineering*, 138 :106834, 2020.

[21] Sourav Chatterjee. A new coefficient of correlation, 2020.

[22] Sourav Chatterjee and Susan Holmes. Xicor : Association measurement through cross rank increments. `https://cran.r-project.org/web/packages/XICOR/index.html`, 2020.

[23] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. Pywavelets : A python package for wavelet analysis. *Journal of Open Source Software*, 4(36) :1237, 2019.

[24] Shinji Jomori, Nobuaki Nonoyama, and Toshihiko Yoshida. Analysis and modeling of pemfc degradation : Effect on oxygen transport. *Journal of Power Sources*, 215 :18–27, 2012.

[25] Christophe Robin. *Développement d'un modèle prédictif de durée de vie d'une pile PEMFC pour une application aéronautique : étude des interactions entre le cœur de pile et les conditions d'opération du système*. Theses, Université Grenoble Alpes, November 2015.

[26] Marine Jouin. *Contribution au pronostic d'une pile à combustible de type PEMFC : approche par filtrage particulaire*. PhD thesis, 2015. Thèse de doctorat dirigée par Zerhouni, NoureddineHissel, DanielPéra, Marie-Cécile et Gouriveau, Rafael Automatique Besançon 2015.

[27] Alexander Kneer and Nadja Wagner. A semi-empirical catalyst degradation model based on voltage cycling under automotive operating conditions in PEM fuel cells. *Journal of The Electrochemical Society*, 166(2) :F120–F127, 2019.

[28] Jorge J. Moré. The levenberg-marquardt algorithm : Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[29] Wikipedia contributors. Levenberg–marquardt algorithm — Wikipedia, the free encyclopedia, 2021. [Online ; accessed 9-August-2021].

[30] Wikipedia contributors. Trust region — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Trust_region&oldid=994102428`, 2020. [Online ; accessed 9-August-2021].

[31] Matthew Newville, Till Stensitzki, Daniel B. Allen, and Antonino Ingargiola. LMFIT : Non-Linear Least-Square Minimization and Curve-Fitting for Python, September 2014.

[32] Wikipédia. Critère d'information d'akaike — wikipédia, l'encyclopédie libre, 2021. [En ligne ; Page disponible le 20-juillet-2021].

[33] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 :1735–80, 12 1997.

[35] Xueheng Qiu. Ensemble time series forecasting with applications in power systems and financial markets, 2018.

[36] M. A. Ganaie, Minghui Hu, M. Tanveer*, and P. N. Suganthan*. Ensemble deep learning : A review, 2021.

[37] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow : Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[38] Fabrice Gamboa, Thierry Klein, Agnès Lagnoux, and Leonardo Moreno. Sensitivity analysis in general metric spaces. *Reliability Engineering & System Safety*, 212 :107611, 03 2021.

[39] François Bachoc, Fabrice Gamboa, Max Halford, Jean-Michel Loubes, and Laurent Risser. Explaining machine learning models using entropic variable projection, 2020.

Annexe A

Tension des campagnes

A.1 Description détaillée des bases de données

Le tableau ci-dessous présente les différentes conditions nominales de fonctionnement des différentes campagnes de vieillissement.

Référence campagne		C1	C2	C3	C4
Température	T°(C)	50	70	70	70
Pression	P_{H_2}/P_{O_2} (bara)	1,2	2	1,2	2
Coefficients stœchiométriques	$\lambda_{H_2}/\lambda_{O_2}$	1,2/1,4	1,2/1,4	1,5/1,5	1,28/1,86
Densité de courant	J_{nom} (A/cm^2)	1	1,5	1	1
Puissance nominale	P_{nom} (kW)	~ 1	$\sim 1,4$	$\sim 2,7$	~ 10

Table A.1 – Conditions nominales de fonctionnement

Les différentes piles des campagnes sont constituées d'AMEs (Assemblage Membrane Electrodes) de différents types (Type 1 à 4). Il est à noter que la campagne 3 est assez particulière : elle est composée de trois piles (S1, S2, S3) électriquement en série et fluidiquement parallèle. Le nombre de cellules, ainsi que les surfaces de chacun des stades est présenté dans le tableau ci-dessous.

Référence campagne	C1	C2	C3 (S1/S2/S3)	C4
N_{cell} (-)	10	10	10	1000
Surface (cm^2)	130	130	400	130
Composition des AMEs	Type 1	Type 1	Type 2/Type 3/Type 4	Type 1

Table A.2 – Paramètres des différentes piles de la campagne

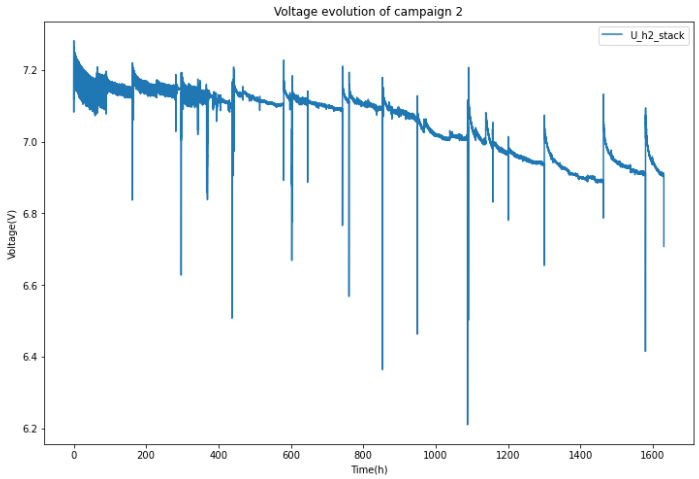
Comme mentionné précédemment, les durées des campagnes étaient conditionnées par le dépassement d'un certain seuil de fuite interne. La durée des plages par contre ainsi que la durée des temps d'arrêt ne suivait aucun plan d'expériences prédéfini. Le tableau ci-dessous présente les différentes caractéristiques des plages de fonctionnement par campagne.

Référence campagne	C1	C2	C3 (S1)	C3 (S2)	C3 (S3)	C4
Nombre plages (-)	37	19	38	19	25	21
Temps moyen (h)	218	94	113	108	119	88
Temps min (h)	3.8	0.1	1.5	1.5	1.5	0.2
Temps max (h)	554	169	261	186	186	184
Durée campagne (h)	8072	1795	4306	2045	2966	1839

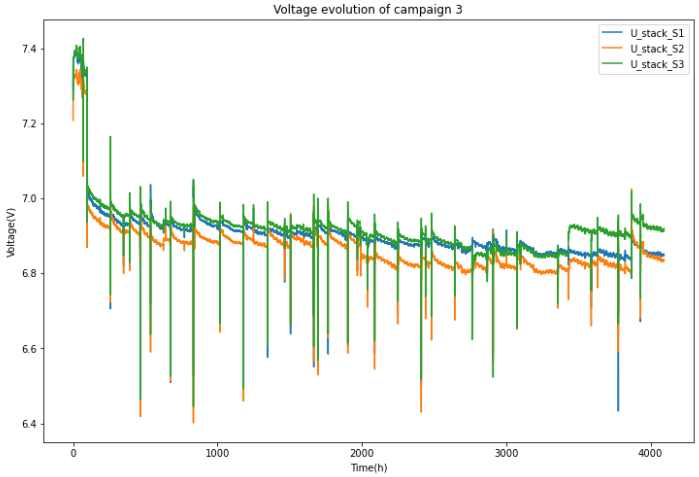
Table A.3 – Temps des plages de fonctionnement par campagne et par pile

A.2 Tensions des autres campagnes

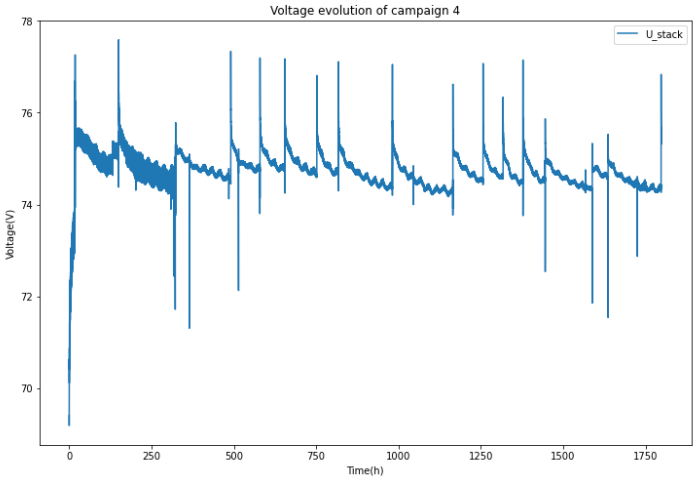
Les figures suivantes présentent les tensions des campagnes 2, 3 et 4.



(a) Campagne 2



(b) Campagne 3



(c) Campagne 4

FIGURE A.1 – Tension des piles des autres campagnes

Annexe B

Études des corrélations

B.1 Mesure de corrélation de Chatterjee

Considérons un couple de variables aléatoires (X, Y) où Y est non constant. Soient $(X_1, Y_1), ...(X_n, Y_n)$ des couples de variables indépendantes et identiquement distribuées, ayant la même loi que (X, Y) pour $n \geq 2$. On suppose qu'il n'existe pas de liens entre X_i et Y_i $\forall i$. Réordonnons les couples en $(X_{(1)}, Y_{(1)}), ...(X_{(n)}, Y_{(n)})$, tel que $X_{(1)} \leq ... \leq X_{(n)}$.
L'absence de liens entre les X_i $\forall i$ rend cet ordre unique. Soit alors r_i le rang de $Y_{(i)}$ qui est le nombre de j tels que $Y_{(j)} \leq Y_{(i)}$. La mesure de Chatterjee s'écrit alors :

$$\xi_n(X, Y) = 1 - \frac{3 \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1} \tag{B.1}$$

En présence de liens, on définit en plus l_i comme le nombre de j tels que $Y_{(j)} \geq Y_{(i)}$. La formule devient alors :

$$\xi_n(X, Y) = 1 - \frac{n \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{2 \sum_{i=1}^{n} l_i(n - l_i)} \tag{B.2}$$

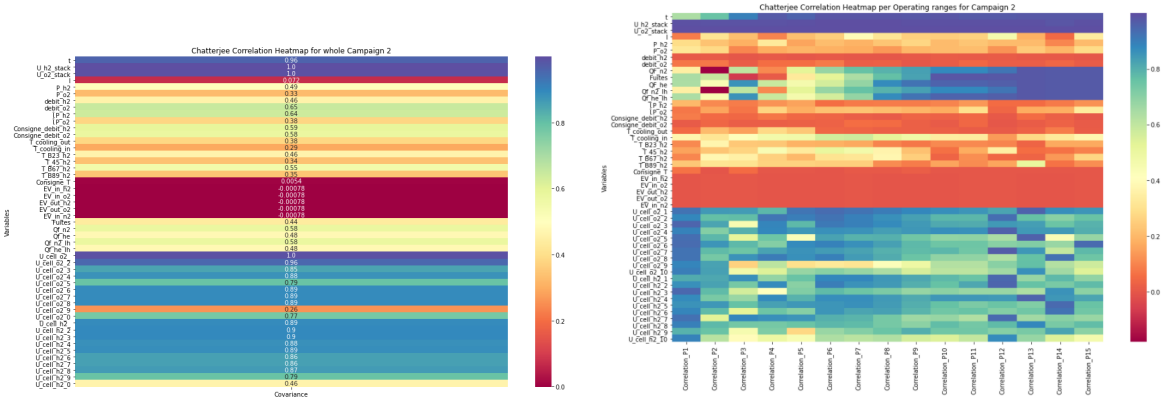
ξ_n est un estimateur consistant d'une mesure de dépendance entre les variables X et Y comme le théorème de l'article de Chatterjee [21] :

Théorème 1.1 : Si Y n'est quasiment pas constant, alors $\xi_n(X, Y)$ converge assurément vers la limite $\xi(X, Y)$ lorsque $n \longrightarrow +\infty$

$$\xi(X, Y) = \frac{\int Var(\mathbb{E}(1_{\{Y \geq t\}}|X))d\mu(t)}{\int Var(\mathbb{E}(1_{\{Y \geq t\}}))d\mu(t)} \tag{B.3}$$

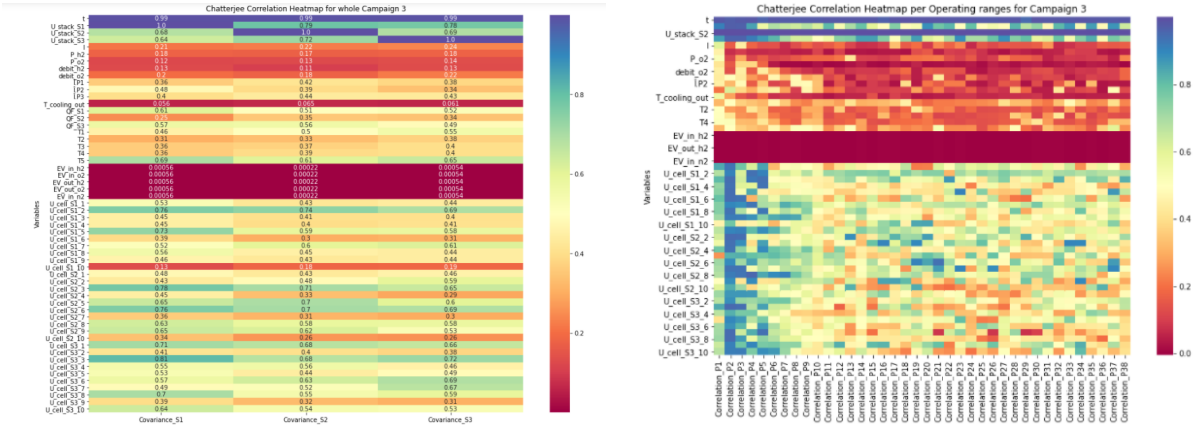
où μ désigne la loi de Y.
Cette limite appartient à l'intervalle $[0; 1]$ et est égale à 0 si et seulement si X et Y sont indépendants et à 1 si et seulement s'il existe une fonction mesurable $f : \mathbb{R} \longrightarrow \mathbb{R}$ telle que $f(X) = Y$.
Les preuves sur la démonstration du théorème sont disponibles dans l'article [21].

B.2 Calcul des corrélations sur les autres campagnes



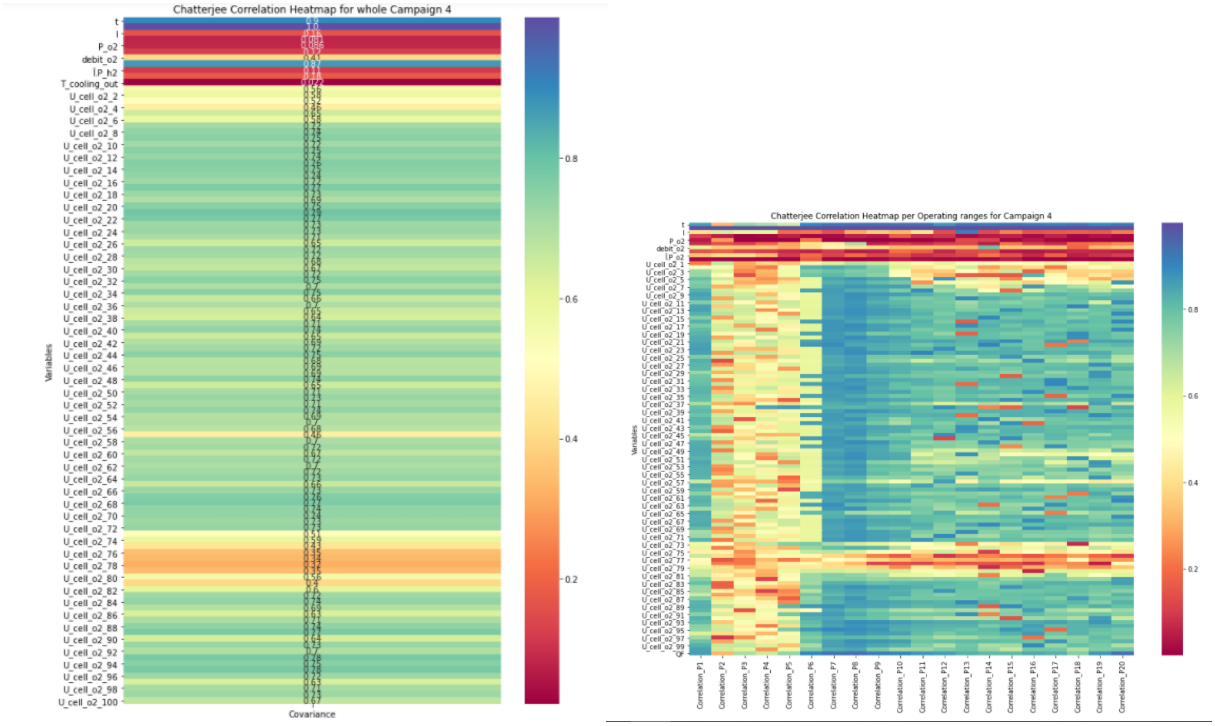
(a) Corrélations globales (b) Corrélation par plages

FIGURE B.1 – Heatmap des corrélations de la campagne 2



(a) Corrélations globales (b) Corrélation par plages

FIGURE B.2 – Heatmap des corrélations de la campagne 3



(a) Corrélations globales (b) Corrélation par plages

FIGURE B.3 – Heatmap des corrélations de la campagne 4

Annexe C

Modèle physique

C.1 Élaboration du modèle

Le modèle de dégradation de base utilisé est le suivant :

$$V_{cell}(J) = E_{rev} - \eta_{act} - \eta_{diff} - \eta_{ohm} \tag{C.1}$$

Il existe différentes expressions des pertes de la pile dans la littérature et aucune modélisation ne fait consensus. Nous avons décidé de nous baser sur certaines modélisations que nous avons jugé intéressantes. Tout d'abord Robin et al. dans ses travaux [25] a introduit un facteur de dégradation pour modéliser la perte de surface active. La dissolution du platine ainsi que les rayons des particules de platine sont les hypothèses retenues par Robin dans sa modélisation. Ce taux de dégradation τ_{degr} intervient dans les pertes d'activation ainsi que les pertes de diffusion. Concernant la modélisation de la perte de surface active, plutôt que de se baser sur les travaux de Robin, nous avons utilisé les travaux de Kneer et al. [27]. Ces travaux stipulent que la perte de surface active suit une loi de décroissance exponentielle en fonction d'un coefficient de proportionnalité k et des cycles de tension N et s_{min} qui désigne la surface active restante en fin de vie de la pile (valeur trouvée expérimentalement à 0.2 dans les expérimentations de Kneer et al.).

$$s(N) = s_{min} + (1 - s_{min}) * e^{-kN} \tag{C.2}$$

Notre apport aux modélisations de Kneer et Robin a été de remplacer le coefficient de proportionnalité par des coefficients empiriques modélisant les dégradations irréversibles et réversibles qui affectent la pile k_{irrev}, k_{rev}. Notre taux de dégradation devient alors :

$$\tau_{degr}(t) = s_{min} + (1 - s_{min}) * e^{-(k_{irrev} * t + k_{rev} * \Delta t)} \tag{C.3}$$

avec $\Delta t = t - t_0^i$ où t_0^i désigne le temps d'arrêt de la plage de fonctionnement i.

Comme observé dans la section d'identification des facteurs d'identification du vieillissement, les pics observés sur la tension de la pile, induits par les phénomènes réversibles varient au cours du temps. Elles sont moins marquées en début de campagne et deviennent plus importantes vers la fin de la campagne. Dans un souci de généralité, on notera alors que le coefficient empirique k_{rev} n'est pas constant tout au long de la campagne, mais varie en fonction du temps. Notre taux de dégradation devient donc :

$$\tau_{degr}(t) = s_{min} + (1 - s_{min}) * e^{-(k_{irrev} * t + k_{rev}(t) * \Delta t)} \tag{C.4}$$

On modélise donc les pertes d'activation η_{act} par la formule suivante :

$$\eta_{act} = \frac{RT}{\alpha n F} ln(\frac{J + J_n}{J_0(t_0) + \tau_{degr}(t)}) \tag{C.5}$$

La modélisation des pertes de diffusion passe par les hypothèses faites par Jomori et al. [24] :

$$\eta_{diff} = (\frac{R^A_{diff}(t_0)}{\tau_{degr}(t)} + R^B_{diff}(t_0)) * (J + J_n) \tag{C.6}$$

Les pertes ohmiques s'écrivent dans la littérature [3] comme un produit entre la résistance ohmique R_{ohm} et la densité de courant J et la densité de courant de crossover J_n.

$$\eta_{ohm} = R_{ohm}(t) * (J + J_n) \tag{C.7}$$

Des spectroscopies d'impédance ont permis d'obtenir les valeurs des résistances ohmiques en début (BOL) et en fin de vie (EOL). Tognan et al. [3] dans ses travaux modélisent la résistance ohmique comme évoluant de façon linéaire entre ses deux valeurs. On a alors :

$$R_{ohm}(t) = R_{ohm}^{BOL} + \frac{R_{ohm}^{EOL} - R_{ohm}^{BOL}}{duree_{campagne}} * t \tag{C.8}$$

On obtient ainsi l'équation finale de notre modèle physique qui est la suivante :

$$V_{cell}(J,t) = E_{rev}(P,T) - \frac{RT}{\alpha nF} ln(\frac{J+J_n}{J_0(t_0)+\tau_{degr}(t)}) - (\frac{R_{diff}^{A}(t_0)}{\tau_{degr}(t)} + R_{diff}^{B}(t_0)) * (J+J_n) - R_{ohm}(t) * (J+J_n) \tag{C.9}$$

Le tableau suivant récapitule les différents paramètres connus de ce modèle physique.

Symbole	Paramètre	Valeur	Unité
α	Coefficient de transfert global de charge	0.5	(-)
n	Nombre d'électrons dans la réaction	2	(-)
F	Constante de Faraday	96485	C/mol
R	Constante des gaz parfaits	8.314	J/K/mol
T	Température	273.15	K
E_{rev}	Tension réversible de la cellule	1.12	V
s_{min}	Surface active minimale	0.2	(-)
J_0	Densité de courant d'échange	$\sim 10^{-6}$	A/cm^2
J_n	Densité de courant de crossover	2	mA/cm^2
R_{ohm}^{BOL}	Résistance ohmique en début de vie	0.0814	Ωcm^2
R_{ohm}^{EOL}	Résistance ohmique en fin de vie	0.0949	Ωcm^2

TABLE C.1 – Paramètres du modèle physique

Annexe D

Codes des modèles

D.1 Modèle physique

Le code utilisé pour définir le modèle physique à partir de l'équation est donné ci-après. Il a été implanté avec le langage Python.

Listing D.1 – Code du modèle physique

```python
def get_voltage(t_values,Jo,k_irrev,k_rev_a,k_rev_b,R_diff_A,R_diff_B):
    '''
    Compute the voltage according to polarization equation.
    Linear hypothesis for reversible degradations.

    Args:
        t_values :
        Jo       :
        k_irrev  :
        k_rev_a  :
        k_rev_b  :
        R_diff_A :
        R_diff_B :
    '''

    # Fixed parameters
    alpha = 0.5
    n = 2
    F = 96485
    R = 8.314
    T = 273.15450
    E_rev = 1.16
    Jn = 0.002
    s_min = 0.2
    n_cells = 10

    def get_R_ohm(t,t_total=df['t'].iloc[-1]):
        '''
        Compute R ohmic
        '''
        R_ohm_BOL = 0.0949
        R_ohm_EOL = 0.0814

        return R_ohm_BOL +(R_ohm_EOL-R_ohm_BOL)*t/t_total

    def get_tau_degr(t,t_arret,k_irrev,k_rev_a,k_rev_b):
        '''
        Compute the degradation rate.
        '''
        delt = t - t_arret
        k_rev = k_rev_a*t + k_rev_b # Linear hypothesis
        #k_rev = k_rev_a*(np.log(t)) + k_rev_b  # Logarithmic hypothesis

        return s_min + (1-s_min)*(np.exp(-(k_irrev*t + k_rev*delt)))

    t = t_values[:,0]
```

```
47      t_arret = t_values[:,1]
48      J = 1
49      tau_degr = get_tau_degr(t,t_arret,k_irrev,k_rev_a,k_rev_b)
50      r_ohm = get_R_ohm(t)
51      loss_act = (R*T)/(alpha*n*F)*(np.log((J+Jn)/(tau_degr*Jo)))
52      loss_diff = ((R_diff_A/tau_degr)+ R_diff_B)*(J+Jn)
53      loss_ohm = r_ohm * (J+Jn)
54      V = (E_rev - loss_act - loss_diff - loss_ohm)*n_cells
55
56      return V
```

Le code de paramétrage du modèle physique avec la librairie LMFIT est donné dans le listing ci-dessous. Le code de prétraitement des données ainsi que celui d'initialisation du modèle est aussi fourni.

Listing D.2 – Paramétrage du modèle

```
1  import matplotlib.pyplot as plt
2  from sklearn.metrics import mean_absolute_error
3  from lmfit import Model,Parameters
4  from lmfit.model import save_modelresult, load_modelresult, load_model, save_model
5
6
7  def preprocess_data(data):
8      '''
9      Preprocess the data for the physical model.
10     Put time values and stopping time in one array.
11
12     Args:
13         data : Dataframe of the dataset with the 't', 't_arret' and 'U_h2_stack' values.
14
15     '''
16     t_data = data['t'].values
17     t_arret = data['t_arret'].values
18     v_data = data['U_h2_stack'].values
19
20     t_values = np.empty((len(v_data), 2))
21     t_values[:,0] = t_data
22     t_values[:,1] = t_arret
23     return t_values,v_data
24
25
26 def init_model(get_voltage, verbose=True,
27                k_rev_c=False):
28     '''
29     Create the Lmfit Model and initialize the parameters.
30
31     Args :
32         get_voltage : The function to wrap in the Lmfit model.
33         verbose     : if the information must be printed or not.
34         k_rev_c     : if true, a 'k_rev_c' parameter will be initialize.
35     '''
36     model_phy = Model(get_voltage)#, independent_vars=['t_values'],
37     params = Parameters()
38
39     params.add('Jo', value=1e-8, min=0, max=1e-3)
40     params.add('k_irrev',value=0 , min=0, max=0.1)
41     params.add('k_rev_a', value=0, min=0)
42     params.add('k_rev_b', value=0)
43     if (k_rev_c):
44         params.add('k_rev_c', value=0)
45     params.add('R_diff_A',value=0 , min=0, max=0.5)
46     params.add('R_diff_B',value=0, min=0, max=0.5)
47
48     model_phy.make_params()
49
50     if (verbose):
51         print('parameter names: {}'.format(model_phy.param_names))
52         print('independent variables: {}'.format(model_phy.independent_vars))
53
54     return model_phy, params
55
56
```

```python
def train_physical_model(model_phy, params, t_values_train,v_data_train,
                         method='least_squares',verbose=True):
    '''
    Fit the physical model to the train data.

    Args:
        model_phy      : The physical model to fit.
        params         : The parameters of the physical model.
        t_values_train : The time values ('t' and (t_arret)) for training.
        v_data_train   : The voltage values for training.
        method         : The optimization method to use for fitting.
        verbose        : If the information must be printed or not.
    '''

    result = model_phy.fit(v_data_train,params, t_values=t_values_train,method=method)
    if (verbose):
        print(result.fit_report())
    return result

def get_model_predictions(model_phy, result, t_values_val):
    '''
    Get the predictions of the physical model on a dataset.

    Args:
        model_phy    : The physical model fitted.
        result       : The model result which contains the fit results.
        t_values_val : The time values ('t' and (t_arret)) for predictions.
    '''
    v_predicted = model_phy.eval(result.params, t_values=t_values_val)
    return v_predicted
```

D.2 Modèle d'apprentissage automatique

Le code d'entraînement des modèles d'apprentissage est donné dans le listing ci-après.

Listing D.3 – Code d'entraînement réseaux de neurones

```python
def compile_and_fit(model, window, patience=4, max_epochs=10,save_path=''):
    '''
    Compile the model and train on the dataset.

    Args:
        model      : The model to train.
        window     : The window dataset with training and validation data.
        patience   : The number of epochs to wait for early stopping.
        max_epochs : The maximum number of epochs
        save_path  : The path to the folder to save the model and the logs.
    '''

    # Create the Tensorboard callback
    log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    log_dir = os.path.join(save_path,log_dir)
    tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                      patience=patience,
                                                      mode='min')
    model.compile(loss=tf.losses.MeanSquaredError(),
                  optimizer=tf.optimizers.Adam(),
                  metrics=[tf.metrics.MeanAbsoluteError()])
    history = model.fit(window.train, epochs=max_epochs,
                        validation_data=window.val,
                        callbacks=[early_stopping,tensorboard_callback])
    return history
```

Les codes des différents réseaux de neurones sont donnés dans le listing ci-après.

Listing D.4 – Code des réseaux de neurones

```python
def create_linear_model(num_features, normalizer, out_steps):
    '''
    Create the linear model.

    Args:
        num_features : The features to pass to the model.
        normalizer   : The Keras preprocessing layer for Normalization.
        out_steps    : The size of the window moving method
    '''
    multi_linear_model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
    normalizer,
    # Take the last time-step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
    # Shape => [batch, 1, out_steps*features]
    tf.keras.layers.Dense(out_steps*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([out_steps, num_features])
    ])
    return multi_linear_model

def create_dense_model(num_features, normalizer, out_steps):
    '''
    Create the linear model.

    Args:
        num_features : The features to pass to the model.
        normalizer   : The Keras preprocessing layer for Normalization.
        out_steps    : The size of the window moving method
    '''

    multi_dense_model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
    normalizer,
    # Take the last time step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
    # Shape => [batch, 1, dense_units]
    tf.keras.layers.Dense(512, activation='relu'),
    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(out_steps*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([out_steps, num_features])
    ])
    return multi_dense_model

def create_conv_model(num_features, normalizer, out_steps,conv_width=3):
    '''
    Create the CNN model.

    Args:
        num_features : The features to pass to the model.
        normalizer   : The Keras preprocessing layer for Normalization.
        out_steps    : The size of the window moving method
    '''

    multi_conv_model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
    normalizer,
    # Shape [batch, time, features] => [batch, conv_width, features]
    tf.keras.layers.Lambda(lambda x: x[:, -conv_width:, :]),
    # Shape => [batch, 1, conv_units]
    tf.keras.layers.Conv1D(256, activation='relu', kernel_size=(conv_width)),
    # Shape => [batch, 1,  out_steps*features]
    tf.keras.layers.Dense(out_steps*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([out_steps, num_features])
    ])
    return multi_conv_model
```

```
74
75  def create_rnn_model(num_features, normalizer, out_steps):
76      '''
77      Create the RNN model.
78
79      Args:
80          num_features : The features to pass to the model.
81          normalizer   : The Keras preprocessing layer for Normalization.
82          out_steps    : The size of the window moving method
83      '''
84
85      multi_rnn_model = tf.keras.Sequential([
86      tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
87      normalizer,
88      # Adding more `rnn_units` just overfits more quickly.
89      tf.keras.layers.SimpleRNN(32, return_sequences=False),
90      # Shape => [batch, out_steps*features]
91      tf.keras.layers.Dense(out_steps*num_features,
92                            kernel_initializer=tf.initializers.zeros()),
93      # Shape => [batch, out_steps, features]
94      tf.keras.layers.Reshape([out_steps, num_features])
95      ])
96      return multi_rnn_model
97
98
99  def create_gru_model(num_features, normalizer, out_steps):
100     '''
101     Create the GRU model.
102
103     Args:
104         num_features : The features to pass to the model.
105         normalizer   : The Keras preprocessing layer for Normalization.
106         out_steps    : The size of the window moving method
107     '''
108
109     multi_gru_model = tf.keras.Sequential([
110     tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
111     normalizer,
112     # Adding more `rnn_units` just overfits more quickly.
113     tf.keras.layers.GRU(32, return_sequences=False),
114     # Shape => [batch, out_steps*features]
115     tf.keras.layers.Dense(out_steps*num_features,
116                           kernel_initializer=tf.initializers.zeros()),
117     # Shape => [batch, out_steps, features]
118     tf.keras.layers.Reshape([out_steps, num_features])
119     ])
120     return multi_gru_model
121
122 def create_lstm_model(num_features, normalizer, out_steps):
123     '''
124     Create the LSTM model.
125
126     Args:
127         num_features : The features to pass to the model.
128         normalizer   : The Keras preprocessing layer for Normalization.
129         out_steps    : The size of the window moving method
130     '''
131
132     multi_lstm_model = tf.keras.Sequential([
133     tf.keras.layers.InputLayer(input_shape=[out_steps,num_features]),
134     normalizer,
135     # Adding more `lstm_units` just overfits more quickly.
136     tf.keras.layers.LSTM(32, return_sequences=False),
137     # Shape => [batch, out_steps*features]
138     tf.keras.layers.Dense(out_steps*num_features,
139                           kernel_initializer=tf.initializers.zeros()),
140     # Shape => [batch, out_steps, features]
141     tf.keras.layers.Reshape([out_steps, num_features])
142     ])
143     return multi_lstm_model
```

D.3 Modèle hybride

Le code pour la méthode hybride est présenté dans le listing ci-après.

Listing D.5 – Code des méthodes d'ensemble

```
    def ensemble_voting_predictions(self, weighted_average=False, weights=None):
        '''
        Compute the predictions with the ensemble voting and weighted average method.

        Args:
            weighted_average : if true compute the weighted average method.
                               Else voting ensemble method.
            weights          : the weights for the weighted average model.
                               If None, it will be computed based on
                               the validation dataset. Can be list ([w_physical, w_dl])
                               or dictionaries.

        '''
        if weighted_average:
            if (weights is None):
                self.weights = self.compute_weights()
            else:
                if(isinstance(weights,list)):
                    self.weights =  {name:w for name,w in zip(self.types,weights)}
                elif (isinstance(weights,dict)):
                    self.weights = weights
                else:
                    raise Exception('TypeError : Must be list or dict type !')
            pred_array = np.zeros(len(self.val))
            for model_type in self.types:
                pred_array += self.weights[model_type]*self.get_predictions(model_type)
            self.score = self.compute_score(self.val['U_h2_stack'].values, pred_array,self.metric)
            return pred_array

        else:
            # Get and stack the predictions of all the models
            pred_array = None
            for model_type in self.types:
                if isinstance(pred_array, np.ndarray):
                    pred_array = np.stack((pred_array,self.get_predictions(model_type)), axis=0)
                else:
                    pred_array = self.get_predictions(model_type)
            pred_array = np.mean(pred_array, axis=0)
            self.score = self.compute_score(self.val['U_h2_stack'].values, pred_array,self.metric)

            return pred_array

    def stacking_training(self,meta_type='linear',train_ratio=0.7,**kwargs):
        '''
        Train the meta model for the stacking method.
        Can be (non negative) Linear Regression, Ridge Regression, SGD Regression, Linear SVM.

        Args:
            meta_type   : the type for the meta model. Can be linear, ridge, svm, sgd.
            train_ratio : the fraction p to get the (1-p) fraction of the
                          training dataset used for training
                          the meta model.
            kwargs      : optional arguments for the meta model.

        '''

        # Get the subset (30%) from training data
        n = len(self.train)
        train = self.train[int(n*train_ratio):]

        # Make predictions from the models
        pred_array = None
        for model_type in self.types:
            if isinstance(pred_array, np.ndarray):
                pred_array = np.stack((pred_array,
                                       self.get_predictions(model_type,data=train,
                                                            stacking=True,
```

```
                                                        train_ratio=train_ratio)),
                                        axis=-1)
                #print("train",pred_array.shape)
            else:
                pred_array = self.get_predictions(model_type,
                                                  data=train,
                                                  stacking=True,
                                                  train_ratio=train_ratio)

        # Create the meta-model
        if meta_type=='linear':
            model = LinearRegression(positive=True,**kwargs)
        elif meta_type=='ridge':
            model = Ridge(alpha=0.5,solver='sag',**kwargs)
        elif meta_type=='svm':
            model = svm.LinearSVR(**kwargs)
        elif meta_type=='sgd':
            model = SGDRegressor(loss='huber', **kwargs)
        elif meta_type=='gaussian':
            model = GaussianProcessRegressor(**kwargs)
        elif meta_type=='knn':
            model = KNeighborsRegressor(**kwargs)
        elif meta_type=='boosting':
            model = GradientBoostingRegressor(**kwargs)
        else:
            raise Exception('The type of the meta-model is unknown')

        self.meta_model = Pipeline([('scaler', StandardScaler()),
                                    ('model',model)
                                    ])
        # Train the meta learner
        self.meta_model.fit(pred_array, train['U_h2_stack'].values)

    def stacking_predictions(self, meta_type='linear', train_ratio=0.7, **kwargs):
        '''
        Make the predictions for the stacking method.
        Can be (non negative) Linear Regression, Ridge Regression, SGD Regression, Linear SVM.

        Args:
            meta_type   : the type for the meta model. Can be linear, ridge, svm, sgd.
            train_ratio : the fraction p to get the (1-p) fraction of
                          the training dataset used for training
                          the meta model.
            kwargs      : optional arguments for the meta model.
        '''

        if (self.meta_model == None):
            self.stacking_training(meta_type=meta_type, train_ratio=train_ratio,
                                   **kwargs)
        # Get and stack the predictions of all the models
        inputs = None
        for model_type in self.types:
            if isinstance(inputs, np.ndarray):
                inputs = np.stack((inputs,self.get_predictions(model_type)), axis=-1)
                #print("pred", inputs.shape)
            else:
                inputs = self.get_predictions(model_type)

        # Make predictions with the meta learner
        if (meta_type=='gaussian'):
            predicts, std = self.meta_model.predict(inputs, return_std=True)
            self.score = self.compute_score(self.val['U_h2_stack'].values,
                                            predicts,self.metric)
            return predicts,std
        else:
            predicts = self.meta_model.predict(inputs)
            self.score = self.compute_score(self.val['U_h2_stack'].values,
                                            predicts,self.metric)
            return predicts
```

Résumé

La durée de vie des piles à combustible constitue un frein à l'exploitation de la technologie hydrogène dans l'industrie automobile. Ce rapport présente l'étude d'une base de données expérimentales de vieillissement des piles à combustible afin d'identifier les facteurs principaux du vieillissement. Nous proposons aussi un modèle prédictif de l'évolution de la tension de la pile. Ce modèle hybride qui est basé à la fois sur de l'apprentissage automatique et sur une modélisation des phénomènes physiques permet d'obtenir un modèle partiellement explicable tout en ayant une erreur moyenne absolue de 0.0991.

Mots-clés : Pile à combustible, Vieillissement, Modèle physique, Machine learning, Explicabilité de modèles.

Abstract

The lifespan of fuel cells is a barrier to the use of hydrogen technology in the automotive industry. This report presents the study of an experimental fuel cell aging database in order to identify the main factors of aging. We also propose a predictive model of the evolution of the battery voltage. This hybrid model, based both on machine learning and on a modeling of physical phenomena, makes it possible to obtain a partially explainable model while having a mean absolute error of 0.0991.

Keywords : Fuel cell, Aging, Physical model, Machine learning, Model explainability.