



HAL
open science

Contribution à la synthèse de contrôleurs neuronaux robustes par imitation

Jérémy Pinguet

► **To cite this version:**

Jérémy Pinguet. Contribution à la synthèse de contrôleurs neuronaux robustes par imitation. Automatique. Université Paris-Saclay, 2023. Français. NNT : 2023UPASG002 . tel-04475364

HAL Id: tel-04475364

<https://theses.hal.science/tel-04475364>

Submitted on 23 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Contribution à la synthèse de contrôleurs neuronaux robustes par imitation

*Contribution to the synthesis of robust neural
controllers by imitation*

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies
de l'Information et de la Communication (STIC)
Spécialité de doctorat : Automatique
Graduate School : Informatique et Sciences du Numérique
Référent : CentraleSupélec

Thèse préparée dans l'unité de recherche **Laboratoire des Signaux et Systèmes (Université Paris-Saclay, CNRS, CentraleSupélec)** et **Safran Electronics & Defense**, sous la direction de **Guillaume SANDOU**, professeur et le co-encadrement de **Philippe FEYEL**, docteur-ingénieur

Thèse soutenue à Paris-Saclay, le 06 janvier 2023, par

Jérémy PINGUET

Composition du jury

Membres du jury avec voix délibérative

Cristina STOICA
Professeure, Université Paris-Saclay, CentraleSupélec
Jean-Marc BIANNIC
Professeur, ONERA et ISAE-Supaero
Edouard LAROCHE
Professeur, Université de Strasbourg
Madiha NADRI WOLF
Maîtresse de Conférences, Université Claude Bernard
Lyon 1

Présidente
Rapporteur & Examineur
Rapporteur & Examineur
Examinatrice

Remerciements

Je tiens à exprimer ma profonde gratitude envers les personnes qui m'ont aidé tout au long de ma recherche et de la rédaction de cette thèse.

Tout d'abord, je souhaite remercier les membres du jury de thèse pour leur temps et leurs efforts à évaluer mon travail. J'adresse toute ma reconnaissance aux deux rapporteurs, Jean-Marc Biannic, professeur à l'ONERA et l'ISAE-Supaero, ainsi qu'Édouard Laroche, professeur à l'université de Strasbourg, pour leurs lectures minutieuses des travaux et leurs retours qui ont permis d'améliorer la qualité de ce manuscrit. Je remercie Mahiha Nadri Wolf, maîtresse de conférence à l'université Claude Bernard Lyon 1, pour sa participation au jury de thèse en tant qu'examinatrice, et je remercie également Cristina Stoica, professeure à CentraleSupélec, d'avoir accepté de présider le jury.

Je tiens à exprimer mes plus sincères remerciements à mon encadrant industriel Philippe Feyel, docteur-ingénieur chez Safran, sans lequel cette recherche doctorale n'aurait jamais existé. Je lui suis reconnaissant pour avoir placé sa confiance en moi dès notre rencontre et tout au long de ces années. Je le remercie particulièrement pour son expertise innovante, sa passion communicative et contagieuse, ainsi que sa grande disponibilité que 5503 kilomètres n'ont pu atténuer. Sa bienveillance et sa capacité à m'orienter m'ont aidé à faire face aux difficultés rencontrées lors de cette aventure scientifique et humaine. Je désire également adresser ma sincère gratitude à mon directeur de thèse Guillaume Sandou, professeur à CentraleSupélec, pour son soutien, sa patience et la confiance qu'il m'a accordé pendant ces années. Ses précieux conseils, ses commentaires constructifs en plus de sa sympathie permanente ont été indispensables pour mener à bien ce travail.

Je tiens également à remercier les personnes du laboratoire qui m'ont entouré pendant ces années. Je remercie l'ensemble des membres du département automatique de CentraleSupélec, notamment les enseignants et les chercheurs que j'ai pu rencontrer et avec qui j'ai pu échanger sur des sujets passionnants. Mes pensées se tournent naturellement vers mes compagnons doctorants avec qui j'ai partagé cette expérience, je remercie Vincent, Thomas, Dario, Matthieu, Maxime, Benjamin, Martin, Joy, Nihed, Baptiste, Miguel pour tous ces excellents moments passés ensemble autour d'un café, d'un tableau blanc, d'un sujet de TP, d'un code buggé, d'un repas plus ou moins goûteux, et tous ceux autour d'une bière, d'un jeu, d'un mölkkky, d'un filet de beach-volley ou d'une pièce de théâtre.

Je souhaite exprimer ma reconnaissance à l'ensemble de mes collègues de Safran Electronics & Defense et les nombreuses personnalités enrichissantes que j'ai pu rencontrer. J'ai une pensée

particulière pour Maxime, Nicolas, Benjamin, Estelle, Philippe, William qui par leur bonne humeur et leur enthousiasme ont permis d'entretenir une ambiance conviviale harmonieusement répartie entre rires et discussions théoriques. Je remercie également les membres du pôle asservissement pour leur intérêt et leur sympathie, ainsi que l'équipe du PAD pour m'avoir chaleureusement accueillie durant cette période particulière.

Je ne saurais également oublier de remercier ma famille et mes amis pour leur soutien et leur encouragement tout au long de cette aventure. Je remercie mes amis manceaux, dont l'amitié n'a pas d'égal et perdure depuis les bancs du lycée où ce long chemin académique a débuté à leur côté. Je remercie également mes amis de l'ENSEM et particulièrement mes colocs pour tous ces moments exceptionnels passés ensemble et leur soutien véritable.

Mon éternelle reconnaissance va à mes parents et mon frère pour l'amour inconditionnel qu'ils me portent. Mes pensées les plus chères vont à Lucile, qui a eu la lourde tâche d'accompagner mes occupations comme mes préoccupations, et dont la compréhension et le sincère soutien ont permis de rendre l'accomplissement de ce doctorat possible.

Table des matières

Remerciements	iii
Table des matières	vii
Table des figures	xi
Liste des tableaux	xv
Introduction générale	1
I Association des outils neuronaux et de la commande robuste	9
1 Identification de systèmes dynamiques par réseaux de neurones	11
1.1 Introduction	12
1.2 Les réseaux de neurones comme modèle de systèmes dynamiques	12
1.2.1 Fonctionnement du perceptron multicouche (MLP)	12
1.2.2 Introduction à l'identification	14
1.2.3 Identification neuronale	16
1.2.4 Architectures générales des réseaux de neurones	19
1.3 Méthodes d'entraînement des réseaux de neurones	20
1.3.1 Indice de performance	20
1.3.2 Rétropropagation de l'erreur	22
1.3.3 Généralisation du réseau	23
1.4 Topologies des réseaux de neurones pour l'identification	24
1.4.1 Fonctions d'activations	24
1.4.2 Topologie des NNARX	24
1.4.3 Topologie des NNOE	26
1.4.4 Topologie générale des SSNN	27
1.4.5 Topologie des SSNNARX et SSNNOE	31
1.5 Procédure d'apprentissage	33
1.5.1 Expérimentation et acquisition des données	33
1.5.2 Pré-traitement des données	35
1.5.3 Sélection de la méthodologie d'apprentissage	37
1.5.4 Analyse des performances et post-traitement du réseau	39

1.6	Conclusions	41
2	Outils d'analyse des systèmes LPV	43
2.1	Introduction	44
2.2	Notions de stabilité et outils dans le cadre linéaire invariant	44
2.2.1	Réponse fréquentielle	45
2.2.2	Norme \mathcal{H}_∞	45
2.2.3	Estimation de la norme \mathcal{H}_∞ par le lemme borné réel	46
2.2.4	Transformation en w	47
2.3	Introduction aux systèmes LPV	49
2.3.1	Définition des systèmes LPV	49
2.3.2	Transformation linéaire fractionnaire (LFT)	50
2.4	Analyse de stabilité et de performance des systèmes LPV-LFT	53
2.4.1	Notions de stabilité des systèmes LPV	53
2.4.2	Théorème du petit gain	54
2.4.3	Théorème du petit gain avec multiplicateurs	55
2.4.4	Analyse de performance des systèmes LPV-LFT	57
2.5	Conclusions	60
3	Modélisation des SSNN pour la commande robuste	61
3.1	Introduction	62
3.2	Représentation des SSNN par des LPV-LFT	62
3.2.1	Formulation quasi-LPV	62
3.2.2	Première formulation LPV-LFT	65
3.2.3	Formulation quasi-LPV non-biaisée	67
3.2.4	Formulation LPV-LFT normalisée	69
3.3	Modélisation des associations de SSNN	73
3.3.1	La mise en série de SSNN est un SSNN	73
3.3.2	Prise en compte des retards dans la forme SSNN finale	74
3.3.3	L'interconnexion de SSNN est un SSNN	75
3.4	Analyse des SSNN via la formulation LPV-LFT	76
3.4.1	Analyse de stabilité et de performance des SSNN	77
3.4.2	Analyse des SSNN avec filtrage des paramètres	78
3.4.3	Analyse des SSNN par μ -analyse	79
3.5	Conclusions	81
II	Méthodes de synthèse de contrôleurs neuronaux robustes	83
4	Identification neuronale de contrôleurs robustes	85
4.1	Introduction	86
4.2	Apprentissage de contrôleurs et enjeu de robustesse	87
4.2.1	Méthodologie d'apprentissage de contrôleur	87
4.2.2	Problème de la stabilité	88
4.2.3	Application au chariot mobile avec pendule	89
4.2.3.1	Présentation du système et du contrôleur considéré	89
4.2.3.2	Identification neuronale du contrôleur	90
4.2.3.3	Simulation de la boucle d'asservissement	92

4.3	Méthodes d'évaluation des marges de stabilité	94
4.3.1	Marges de stabilité dans le cadre linéaire invariant	94
4.3.2	Marges de stabilité de SSNN	101
4.3.3	Méthodologie d'apprentissage et d'évaluation des marges	101
4.3.3.1	Identification neuronale du contrôleur et du système	101
4.3.3.2	Évaluation des marges de stabilité via le modèle LPV-LFT associé	103
4.3.4	Application	104
4.4	Méthodes d'apprentissage avec prise en compte de stabilité	105
4.4.1	Notions de problème d'optimisation multi-objectifs	106
4.4.2	Formulation multi-objectifs de l'apprentissage de contrôleurs robustes	107
4.4.3	Optimisation multi-objectifs sans gradient	108
4.4.3.1	Apprentissage par algorithmes génétiques	109
4.4.3.2	Formulation multi-objectifs	109
4.4.3.3	Méthodologie d'implémentation	110
4.4.3.4	Techniques d'amélioration de la convergence pour l'apprentissage neuronal	111
4.4.4	Application	112
4.5	Conclusions et perspectives	117
4.5.1	Conclusions	117
4.5.2	Perspectives	118
5	Approche multi-modèle pour l'implémentation du contrôleur	119
5.1	Introduction	120
5.2	Introduction au MMAC	120
5.2.1	Principe	120
5.2.2	Architecture du MMAC	121
5.3	Extension neuronale du MMAC	124
5.3.1	Intérêts de l'approche	124
5.3.1.1	Consolidation de l'apprentissage	124
5.3.1.2	Considération des situations difficiles	124
5.3.2	Architecture du MMAC neuronal (NMMAC)	125
5.3.2.1	Banque de contrôleurs	125
5.3.2.2	Banques modèles-estimateurs et logique de sélection	126
5.4	Étude de la stabilité et réglage dans le cas linéaire	127
5.4.1	État de l'art	127
5.4.2	Analyse du processus de décision incertain	128
5.4.2.1	Structure d'analyse du MMAC	129
5.4.2.2	Domaine de stabilité du MMAC fondé sur la μ -analyse	131
5.4.2.3	Application	132
5.4.3	Réglage de la logique de décision dans le cas linéaire	135
5.4.3.1	Approche <i>Estimator-based MMAC</i>	135
5.4.3.2	Critère caractéristique du réglage du MMAC	136
5.4.3.3	Optimisation du réglage du MMAC	137
5.4.3.4	Application	137
5.4.4	Limitations de l'étude	140
5.5	Conclusions et perspectives	142
5.5.1	Conclusions	142

5.5.2	Perspectives	142
III	Applications à un pilote automatique d'avion	143
6	Plateforme d'apprentissage d'autopilotes neuronaux	145
6.1	Introduction	146
6.2	Module de co-simulation	147
6.2.1	Le simulateur de vol : X-Plane	147
6.2.1.1	Présentation	147
6.2.1.2	Interface de données	148
6.2.2	Mise en pratique de la co-simulation	149
6.2.2.1	Configuration de X-Plane	149
6.2.2.2	Configuration de Matlab et Simulink	150
6.3	Module d'apprentissage de contrôleurs robustes	151
6.3.1	Synthèse d'un autopilote de guidage en vol libre	151
6.3.1.1	Description du problème	151
6.3.1.2	Génération des données	152
6.3.1.3	Imitation de l'autopilote de X-Plane	154
6.3.1.4	Imitation d'un pilote	156
6.3.2	Synthèse d'un autopilote de guidage pour le suivi du chemin de vol	164
6.3.2.1	Description du problème	164
6.3.2.2	Génération des données	167
6.3.2.3	Apprentissage de l'autopilote neuronal	167
6.3.2.4	Expérimentations	170
6.4	Module de déploiement de l'autopilote par multi-modèle	176
6.4.1	Déploiement d'un NMMAC de guidage en vol libre	176
6.4.1.1	Description et intérêt du problème	176
6.4.1.2	Apprentissage multi-modèle de l'autopilote de X-Plane	176
6.4.1.3	Implémentation du NMMAC pour le contrôle autonome	178
6.4.2	Déploiement d'un NMMAC de guidage pour le suivi de chemin de vol	179
6.4.2.1	Description du problème	179
6.4.2.2	Apprentissage multi-modèle d'un pilote	182
6.4.2.3	Implémentation du NMMAC pour le suivi de chemin de vol	184
6.5	Conclusions et perspectives	186
6.5.1	Conclusions	186
6.5.2	Perspectives	188
	Conclusion générale	190
	Bibliographie	195

Table des figures

1	Schéma décrivant les étapes de conception d'un autopilote par imitation : collecte des données, apprentissage hors ligne, puis contrôle autonome ou assistance . . .	5
1.1	Représentation d'un neurone : un neurone artificiel calcule la sortie d'une fonction souvent non-linéaire d'une somme pondérée de ses entrées	13
1.2	Représentation d'un MLP	14
1.3	Représentation d'un MLP entièrement récurrent	14
1.4	Identification neuronale par NNARX	18
1.5	Identification neuronale par NNOE	18
1.6	Topologie du NNARX	26
1.7	Topologie du NNOE	27
1.8	Topologie du SSNN	29
1.9	Topologie du SSNN sous forme matricielle	30
1.10	Topologie du SSNNOE	31
1.11	Topologie du SSNNARX	33
1.12	Exemple de signal d'excitation	34
2.1	Forme standard LFT supérieure et inférieure	51
2.2	Représentation LPV-LFT	52
2.3	Système d'interconnexion pour le théorème du petit gain	54
2.4	Système d'interconnexion pour le théorème du petit gain avec multiplicateurs . .	56
2.5	Analyse de performance des systèmes LPV-LFT via une analyse de stabilité . . .	58
2.6	Analyse de performance des systèmes LPV-LFT avec multiplicateurs	58
3.1	Représentation LPV-LFT d'un SSNN isolant les non-linéarités	66
3.2	Représentation LPV-LFT d'un SSNN non-biaisé isolant les non-linéarités	69
3.3	Intervalle de la sortie d'un neurone non-linéaire tangente hyperbolique en fonction de son entrée	71
3.4	Représentation LPV-LFT normalisée d'un SSNN non-biaisé	71
3.5	Connexion en série de deux SSNN	73
3.6	Système en boucle fermée composé de deux SSNN	76
3.7	Représentation LPV-LFT d'un SSNN avec filtrage des paramètres	79
4.1	Méthodologie de développement de contrôleurs neuronaux robustes à partir de données	86
4.2	Schéma d'apprentissage de contrôleur	87
4.3	Schéma d'apprentissage de contrôleur avec signal perturbateur d'excitation . . .	88

4.4	Chariot mobile avec pendule	89
4.5	Contrôleur à trois degrés de liberté	90
4.6	Résultats d'apprentissage des contrôleurs neuronaux sur les données de l'ensemble de test	92
4.7	Simulation des contrôleurs d'origine et neuronaux dans le cas nominal et avec rejet d'une perturbation d'entrée de $-5V$ à partir de 8s	93
4.8	Schéma de simulation de la boucle d'asservissement	94
4.9	Simulation des contrôleurs d'origine et neuronaux dans des cas de variations de gain	95
4.10	Marges de stabilité généralisées	96
4.11	Régions d'exclusion dans le plan de Nyquist obtenues pour les différentes marges de stabilité	99
4.12	Marges de gains et de phases généralisées déduites de la marge de disque	100
4.13	Schéma d'apprentissage de contrôleur et du système à asservir	103
4.14	Front de Pareto	107
4.15	Fronts de Pareto des ensembles de contrôleurs neuronaux optimaux	114
4.16	Simulation d'un sous-ensemble des contrôleurs optimaux de Pareto (K_1^*)	115
4.17	Simulation d'un sous-ensemble des contrôleurs optimaux de Pareto (K_2^*)	116
5.1	Architecture du MMAC	121
5.2	Forme LFT pour l'analyse du MMAC	130
5.3	Chariot mobile avec pendule inversé	132
5.4	Analyse de la stabilité pour les 3 configurations du système	134
5.5	Évolution temporelle et trajectoires des validités	139
5.6	Évaluation de la fonction de coût	140
5.7	Réponses du MMAC pour le réglage optimal retenu lors de variations paramétriques	141
6.1	Les trois modules de la plateforme d'apprentissage d'autopilotes neuronaux	146
6.2	Co-simulation entre X-Plane et MATLAB	147
6.3	Schéma de simulation de la boucle d'asservissement de guidage	152
6.4	Cessna 172SP	152
6.5	Instrument de vol G1000 de X-Plane 11	153
6.6	Résultat d'apprentissage du contrôleur pour l'imitation de l'autopilote de X-Plane sur les données de validation et de test	155
6.7	Résultat d'apprentissage du système pour l'imitation de l'autopilote de X-Plane sur les données de validation et de test	156
6.8	Comparaison des simulations de l'autopilote de X-Plane et de celui appris pour un scénario test	157
6.9	Résultat d'apprentissage du contrôleur pour l'imitation du pilote sur les données de validation et de test	158
6.10	Résultat d'apprentissage du contrôleur pour l'imitation du pilote sur les données de validation et de test	159
6.11	Comparaison des simulations du pilote et du contrôleur appris pour un scénario test	160
6.12	Front de Pareto des contrôleurs neuronaux optimaux pour l'imitation du pilote	162
6.13	Comparaison des simulations du pilote et d'un sous-ensemble des contrôleurs optimaux de Pareto pour un scénario test	163
6.14	Cirrus Vision SF50	164

6.15	Suivi de chemin de vol	166
6.16	Suivi du chemin de vol d'apprentissage obtenu avec le pilote	168
6.17	Fronts de Pareto des autopilotes neuronaux optimaux pour le suivi de chemin de vol	171
6.18	Schéma de simulation de la boucle d'asservissement de guidage pour le suivi de chemin de vol	171
6.19	Simulations d'un sous-ensemble des autopilotes optimaux pour le suivi du chemin appris	173
6.20	Simulations d'un sous-ensemble des autopilotes optimaux pour le suivi d'un chemin de test	174
6.21	Simulations d'un sous-ensemble des autopilotes optimaux pour une vitesse plus élevée	175
6.22	Autopilote NMMAC de guidage	177
6.23	Simulation de l'autopilote NMMAC de guidage pour un scénario test comportant des variations de vitesse	180
6.24	Comparaison des simulations de l'autopilote de X-Plane et du NMMAC pour une scénario test comportant des variations de vitesse	181
6.25	Fronts de Pareto des autopilotes du NMMAC pour le suivi de chemin de vol . . .	183
6.26	Suivis du chemin de vol obtenus pour une vitesse de 220kt	184
6.27	Suivis du chemin de vol obtenus pour une vitesse de 160kt et lorsque le volet gauche est arraché	185
6.28	Simulations de l'autopilote NMMAC de suivi de chemin de vol pour des variations du système	187

Liste des tableaux

1.1	Exemples de fonctions d'activation	25
1.2	Coefficient de normalisation pour les différentes normalisations affines	37
3.1	Fonctions d'activation et leur gain	64
4.1	Valeurs nominales des paramètres	90
4.2	Paramètres utilisés pour l'apprentissage dans le cas d'étude du chariot mobile avec pendule	91
4.3	Résultats d'apprentissage du contrôleur sur les bases de données obtenues par l'excitation de la consigne seule (K_1^{NN}) et avec la perturbation d'excitation supplémentaire (K_2^{NN})	91
4.4	Différents types d'incertitudes pour l'analyse de la stabilité	97
4.5	Marges de stabilité et leurs marges généralisées	100
4.6	Schémas d'interconnexion associés aux marges de stabilité de SSNN	102
4.7	Marges de stabilité du contrôleur à identifier évaluées pour le système linéarisé	104
4.8	Résultats d'apprentissage du système non-linéaire du chariot mobile avec pendule	105
4.9	Marges de stabilité des contrôleurs évaluées pour le système neuronal	105
4.10	Résultats de l'apprentissage multi-objectifs des contrôleurs neuronaux pour les deux bases de données	113
4.11	Marges de stabilité des sous-ensembles de contrôleur optimaux de Pareto	114
6.1	Format des datagrammes de X-Plane	149
6.2	Exemple de format d'un groupe de données	149
6.3	Résultats d'apprentissage pour l'imitation de l'autopilote de X-Plane	155
6.4	Marges de stabilité de l'asservissement obtenu par l'imitation de l'autopilote de X-Plane	156
6.5	Résultats d'apprentissage pour l'imitation du pilote	158
6.6	Marges de stabilité de l'asservissement obtenu par l'imitation du pilote	160
6.7	Résultats de l'apprentissage multi-objectifs obtenu par l'imitation du pilote	161
6.8	Résultats d'apprentissage du système pour le suivi de chemin de vol	170
6.9	Résultats de l'apprentissage multi-objectifs des autopilotes de suivi de chemin de vol obtenus par l'imitation du pilote	170
6.10	Résultats d'apprentissage des trois paires modèles-contrôleurs pour l'imitation de l'autopilote de X-Plane	177
6.11	Marges de stabilité des trois paires modèles-contrôleurs pour l'imitation de l'autopilote de X-Plane	178

6.12	Résultats d'apprentissage des systèmes du NMMAC pour le suivi de chemin de vol	182
6.13	Résultats de l'apprentissage multi-objectifs des autopilotes de suivi de chemin de vol obtenus pour les deux modèles additionnels	183

Introduction générale

Contexte de l'étude

Les travaux présentés dans ce rapport ont été réalisés au cours d'une thèse de doctorat menée dans le cadre d'une convention CIFRE (Convention Industrielle de Formation par la REcherche) entre Safran Electronics & Defense (anciennement Sagem) et CentraleSupélec. La thèse s'inscrit dans la continuité d'une thématique globale « d'optimisation de la commande robuste par les techniques d'optimisation moderne » au sein de Safran Electronics & Defense. Elle fait suite à des travaux déjà réalisés dans cette thématique :

- **[Hirwa, 2013] Méthode de commandes avancées appliquées aux viseurs.**
Cette thèse propose une méthode de synthèse de contrôleurs robustes d'ordre réduit voire de structure fixée utile pour l'implémentation. Ces contraintes de structure sont rendues possibles par l'emploi de techniques d'optimisation non-lisse adaptées aux problématiques de type H_2/H_∞ .
- **[Feyel, 2015] Optimisation des correcteurs par les méta-heuristiques. Application à la stabilisation inertielle de ligne de visée.**
En plus de la contrainte de structure, cette thèse propose une méthode de synthèse de correcteurs H_∞ prenant en compte les exigences d'un cahier des charges qui peuvent être mathématiquement formalisées grâce à l'emploi de techniques d'optimisation sans gradient.
- **[Frasnedo, 2016] Optimisation des lois de commande d'un imageur sur critère optronique. Application à un imageur à double étage de stabilisation.**
Cette thèse adresse la problématique d'optimisation de fonctions couteuses en temps de calcul relatives à la thèse antérieure en utilisant l'optimisation bayésienne.
- **[Pouilly-Cathelain, 2020] Synthèse de correcteurs s'adaptant à des critères multiples de haut niveau par la commande prédictive et les réseaux de neurones.**
Cette thèse propose une méthode de synthèse de lois de commande prédictive en considérant tout type de contraintes cette fois-ci évaluables en temps réel. Les travaux abordent le cadre plus général des systèmes non-linéaires en utilisant l'apprentissage neuronal.

Objectifs de la thèse

La problématique du contrôle consiste à développer un algorithme capable de manipuler de façon adéquate les entrées d'un système dans le but d'amener ses sorties à une valeur souhaitée. Le contrôleur est alors synthétisé au regard d'un certain degré d'optimalité comme par exemple en termes de rapidité, de performances ou de garanties d'un certain niveau de stabilité. Du point de vue de l'industrie, l'automaticien en charge de la conception d'une loi de commande adopte souvent la stratégie suivante : il débute par la construction d'un modèle mathématique du système dynamique à contrôler, il doit par la suite comprendre le comportement recherché du système et formaliser mathématiquement ces spécifications ou règles à respecter, enfin sur la base de ces deux éléments il élabore le modèle de contrôle capable de calculer les actions à effectuer en fonction des consignes. Néanmoins, au vu de la demande croissante d'applications intelligentes, les règles deviennent de plus en plus complexes et les comportements recherchés de plus en plus exigeants. Le développement de systèmes autonomes requiert alors de la part de l'automaticien un large ensemble de connaissances et compétences spécifiques. Pourtant, un utilisateur dépourvu de ces connaissances saura généralement réaliser les tâches souhaitées même s'il ne sait pas programmer le comportement en question. De ce

constat et des progrès informatiques, l'idée d'apprendre par imitation a alors gagné en intérêt. Ce paradigme consiste à vouloir transmettre les connaissances à un système numérique à partir de résultats expérimentaux pour qu'il puisse extraire les comportements à suivre.

Les travaux proposés dans cette thèse portent donc sur la conception de systèmes de contrôle capables d'apprendre sur des macro-comportements ou macro-décisions régis par des règles. La complexité des règles et des processus à piloter guident vers l'emploi de contrôleurs de type réseaux de neurones pour leur propriété d'approximation universelle. Lorsque ces règles sont directement formulables mathématiquement, alors le travail est une extension neuronale des travaux déjà réalisés par [Feyel, 2015]. Lorsque la formalisation mathématique des règles est trop complexe et variée (comme par exemple un code de la route), une alternative d'apprentissage est de pouvoir disposer, pour un état du système donné et pour certaines conditions extérieures, de l'action réalisée par un opérateur (un pilote, un conducteur, un régulateur, etc.), et dans ce cas, ce sont ces décisions qui traduisent de fait indirectement les règles. On propose alors d'entraîner directement le contrôleur neuronal sur une base de données regroupant ces informations. Afin de garantir la sûreté d'un tel contrôleur, l'apprentissage devra se faire sous contrainte de stabilité. En effet une attention particulière doit être portée sur la propriété stabilisante du contrôleur y compris lorsque celui-ci perfectionne son apprentissage avec de nouvelles bases de données disponibles.

Ainsi, l'objectif de la thèse est d'apprendre par imitation un contrôle de type réseaux de neurones de manière efficace et robuste sur une base de données regroupant des comportements régis par des règles complexes. L'étude passe alors par la mise en œuvre de différentes phases de recherche :

- La collecte des décisions prises par un opérateur au regard du système et de certaines conditions extérieures.
- L'apprentissage d'un contrôleur sur cette base de données traduisant indirectement les règles, de façon le plus efficace et robuste possible et en prenant en compte a priori des contraintes complexes telles que la stabilité de la boucle d'asservissement résultante.
- La consolidation des connaissances du contrôleur lors de la mise à disposition de nouvelles données.

Les méthodes de développement de systèmes de contrôle présentés dans cette thèse se veulent génériques. Cependant, afin de se confronter à des problématiques concrètes, les travaux seront appliqués au développement de pilotes automatiques neuronaux d'aéronefs. L'objectif est alors d'imiter les facultés de pilotage d'un pilote capable de gérer des conditions de vol réelles. Le schéma présenté par la figure 1 synthétise dans ce contexte aéronautique les différentes étapes de développement. L'autopilote sera entraîné sur une base de données regroupant les actions et réactions du pilote vis-à-vis de l'environnement extérieur et de l'état de l'appareil. Une fois l'apprentissage terminé, l'intelligence artificielle pourrait par exemple avoir un rôle d'assistance ou de conseil pour le pilote y compris en cas de situations dégradées telles la manifestation de pannes ou de conditions météorologiques difficiles.

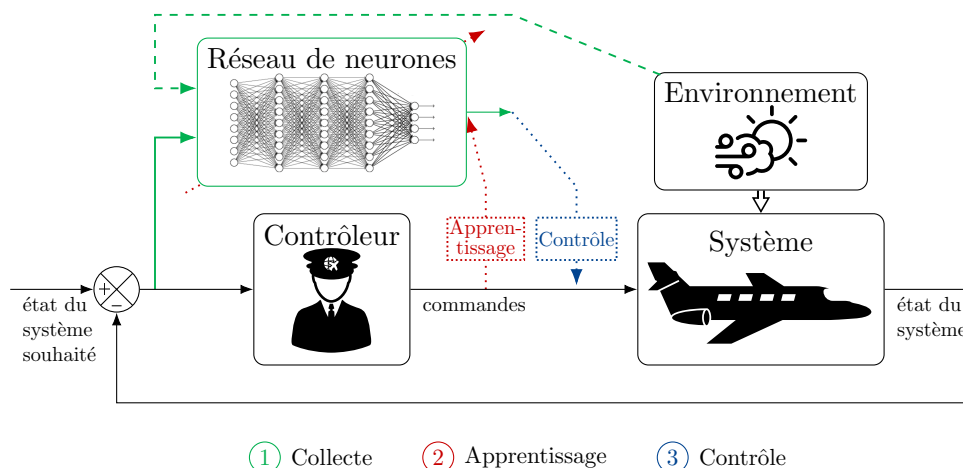


FIGURE 1 – Schéma décrivant les étapes de conception d'un autopilote par imitation : collecte des données, apprentissage hors ligne, puis contrôle autonome ou assistance

Organisation de la thèse

La première partie de ce mémoire est composée de trois chapitres consacrés à la présentation des outils d'apprentissage par réseaux de neurones et de la commande robuste puis de l'unification des deux domaines.

Chapitre 1

Le premier chapitre est consacré à la présentation des réseaux de neurones et leur utilisation pour l'identification hors ligne de systèmes dynamiques. Il introduit dans un premier temps, le perceptron multicouche qui par des liaisons récurrentes permet de modéliser des systèmes dynamiques. Les méthodes d'identification à proprement parler, sont par la suite présentées, elles consistent à optimiser les différents poids du réseau afin d'effectuer la régression de séries temporelles selon un apprentissage supervisé. De plus, ce chapitre dresse un état de l'art des structures neuronales qui sont utilisées dans nos travaux. Ces réseaux opèrent selon un comportement hybride linéaire et non-linéaire qui peut être exprimé suivant une représentation d'état pour être aisément incorporé dans les méthodes de contrôle. Finalement, ce chapitre établit les outils d'apprentissage neuronal qui sont utilisés dans l'ensemble des travaux.

Chapitre 2

Le deuxième chapitre est dédié à l'introduction des outils de la commande robuste pour l'analyse de systèmes linéaires à paramètres variants. Les concepts propres à l'automatique et indispensables à notre étude y sont présentés. Ce chapitre fournit les principaux résultats théoriques d'analyse de robustesse de systèmes représentés par transformation linéaire fractionnaire.

Chapitre 3

Ce chapitre conclut la première partie en fédérant les outils développés au cours des deux chapitres le précédant. Il décrit une méthodologie permettant de modéliser les modèles neuro-naux selon une formulation linéaire à paramètres variants. Cette représentation donne alors la possibilité d'employer les résultats théoriques pour établir des méthodes d'analyse de stabilité

et de performance de ces structures neuronales.

La deuxième partie du mémoire est composée de deux chapitres qui présentent de façon complémentaire les méthodes d'apprentissage de contrôleurs neuronaux robustes puis l'approche multi-modèle afin de consolider et d'adapter la loi de commande à différentes situations.

Chapitre 4

Ce chapitre est consacré à l'apprentissage de contrôleurs neuronaux en tenant compte de critères de robustesse et de stabilité. Il expose en premier lieu le contexte d'apprentissage de lois de commande et les enjeux de stabilité intrinsèques au problème. Par la suite, deux contributions majeures de cette thèse sont présentées : Tout d'abord, une méthode d'évaluation des marges de stabilité d'une boucle d'asservissement neuronale est présentée. Enfin, en s'appuyant sur cette base, une méthodologie d'apprentissage de contrôleurs est décrite afin de directement considérer les critères de robustesse en complément de ceux de performance d'imitation selon une formulation multi-objectifs.

Chapitre 5

Ce chapitre est dédié au développement d'une loi de commande adaptative par une approche multi-modèle. Lorsque le processus à asservir peut appartenir à une large classe de système, les travaux présentés dans ce chapitre consistent à décomposer l'espace de fonctionnement du système et de sa stratégie de contrôle selon plusieurs sous-espaces pour lesquelles des paires modèle-contrôleur sont mises à l'œuvre localement. Les concepts multi-modèle sont étendus à l'utilisation de modèles et de contrôleurs neuronaux. Enfin, ce chapitre établit dans le cadre linéaire une méthode d'analyse de stabilité du contrôle adaptatif multi-modèle ainsi qu'une stratégie de réglage.

La troisième, et dernière partie, du mémoire est composée d'un unique chapitre dédié à l'application des méthodes définies dans la thèse au développement d'un pilote automatique d'avion.

Chapitre 6

Ce dernier chapitre est consacré à la description d'une méthodologie complète d'apprentissage d'autopilotes neuronaux robustes. Cette dernière prend la forme d'une plateforme articulée autour de trois axes de recherche complémentaire. Un premier module expose en détail les éléments de développement d'un environnement de co-simulation entre MATLAB et le simulateur de vol X-Plane. Le deuxième module met en pratique l'ensemble du processus d'apprentissage par imitation de contrôleurs robustes développé au chapitre 4 afin de développer dans un premier temps, un autopilote de suivi de consignes de cap et d'altitude, puis dans un second temps, un autopilote de suivi de chemin de vol défini par une succession de points de passage. Le troisième module exploite les perspectives de l'approche multi-modèle et il constitue donc une application du chapitre 5 pour déployer les différentes facultés de l'autopilote neuronal dans un contexte de contrôle tolérant aux pannes. Ce chapitre est consacré au développement de la sur-couche de guidage de l'aéronef, mais les travaux développés dans cette thèse ne sont pas restreints à ce cas d'étude. Ils constituent une bonne base de travail pour le lecteur souhaitant approfondir l'étude notamment au pilotage plus bas niveau ou étendre le cas d'application à d'autres domaines.

Productions scientifiques

Conférences internationales avec actes

[Pinguet *et al.*, 2020] : Pinguet, J., Feyel, P. et Sandou, G. (2020). A design and analysis method for estimator-based multiple model adaptive control. *In 2020 8th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 74-81. IEEE.

[Pinguet *et al.*, 2021] : Pinguet, J., Feyel, P. et Sandou, G. (2021). A neural autopilot training platform based on a matlab and x-plane co-simulation. *In 2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1200-1209. IEEE.

Pinguet, J., Feyel, P. et Sandou, G. (2023). A data-based neural controller training method with tunable stability margin using multi-objective optimization. *IFAC Proceedings Volumes*.

Communications dans un groupe de travail

Pinguet, J., Feyel, P. et Sandou, G. (2019). Analyse de stabilité du contrôle adaptatif multi-modèle. *Journée des doctorants de Safran Electronics & Defense*, Eragny, France.

Pinguet, J., Feyel, P. et Sandou, G. (2020). Élaboration d'un système de contrôle par autoapprentissage basé sur les techniques neuronales. *Journée des doctorants de Safran Electronics & Defense*, Massy, France.

Pinguet, J., Feyel, P. et Sandou, G. (2021). A neural autopilot training platform based on a Matlab and X-Plane co-simulation. *Journée SYCOMORE du Laboratoire Signaux et Systèmes*, Gif-sur-Yvette, France.

Pinguet, J., Feyel, P. et Sandou, G. (2021). Développement d'un autopilote neuronal à l'aide d'une co-simulation entre Matlab et X-Plane. *Journée des doctorants de Safran Electronics & Defense*, Massy, France.

Première partie

**Association des outils neuronaux et de la
commande robuste**

Identification de systèmes dynamiques par réseaux de neurones

Sommaire

1.1	Introduction	12
1.2	Les réseaux de neurones comme modèle de systèmes dynamiques	12
1.2.1	Fonctionnement du perceptron multicouche (MLP)	12
1.2.2	Introduction à l'identification	14
1.2.3	Identification neuronale	16
1.2.4	Architectures générales des réseaux de neurones	19
1.3	Méthodes d'entraînement des réseaux de neurones	20
1.3.1	Indice de performance	20
1.3.2	Rétropropagation de l'erreur	22
1.3.3	Généralisation du réseau	23
1.4	Topologies des réseaux de neurones pour l'identification	24
1.4.1	Fonctions d'activations	24
1.4.2	Topologie des NNARX	24
1.4.3	Topologie des NNOE	26
1.4.4	Topologie générale des SSNN	27
1.4.5	Topologie des SSNNARX et SSNNOE	31
1.5	Procédure d'apprentissage	33
1.5.1	Expérimentation et acquisition des données	33
1.5.2	Pré-traitement des données	35
1.5.3	Sélection de la méthodologie d'apprentissage	37
1.5.4	Analyse des performances et post-traitement du réseau	39
1.6	Conclusions	41

SECTION 1.1

Introduction

Le principal objectif de ce chapitre est de décrire les approches d'identification fondées sur les réseaux de neurones qui s'avèrent être concrètement applicables à une large classe de systèmes dynamiques non-linéaires. Dans la section 1.2, nous introduisons les principales notions liées aux réseaux de neurones artificiels et leur utilisation pour l'identification. Nous présentons les architectures des perceptrons multicouches (MLP) dont les propriétés temporelles conférées par les liens récurrents et les séquences temporelles d'entrées-sorties permettent de modéliser les systèmes dynamiques. Nous abordons dans la section 1.3, les méthodes d'apprentissage qui sont utilisées dans les travaux afin d'optimiser les poids des modèles neuronaux selon des méthodes supervisées de régression de séries temporelles. Dans la section 1.4, nous développons les topologies de réseaux de neurones pour l'identification notamment celles des SSNN qui confère une liaison entre les techniques de *machine learning* et le domaine de l'automatique. Finalement, ce chapitre fournit une méthodologie complète pour réaliser l'identification neuronale hors ligne de systèmes dynamiques dont nous résumons les principales étapes dans la section 1.5.

SECTION 1.2

Les réseaux de neurones comme modèle de systèmes dynamiques**1.2.1 Fonctionnement du perceptron multicouche (MLP)**

Un neurone artificiel ou perceptron est une entité élémentaire qui, pour un nombre d'entrées donnée, pondère les signaux, les additionne, puis applique une fonction dite d'activation. La figure 1.1 représente un neurone simple à n_u entrées dont le comportement est donné par :

$$y = \sigma \left(\sum_{j=1}^{n_u} w_j u_j + b \right) = \sigma (w^T u + b) \quad (1.1)$$

où $y \in \mathbb{R}$ est la sortie du neurone, $u \in \mathbb{R}^{n_u}$ est le vecteur d'entrée, $w \in \mathbb{R}^{n_u}$ est le vecteur des poids, $b \in \mathbb{R}$ est le biais et $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ est la fonction d'activation. Notons que le biais fait également partie des poids qui définissent le neurone, il peut être interprété comme la pondération appliquée à une entrée fictive constante et unitaire.

Les neurones qui ont des entrées identiques sont regroupés sous forme de couches de neurones. Une couche de n_ℓ neurones génère une sortie $y_\ell \in \mathbb{R}^{n_\ell}$ qui est mathématiquement décrite par :

$$y_\ell = \sigma_\ell (W_\ell u + b_\ell) = \sigma_\ell \left(\begin{bmatrix} w_1^T \\ \vdots \\ w_{n_\ell}^T \end{bmatrix} u + \begin{bmatrix} b_1 \\ \vdots \\ b_{n_\ell} \end{bmatrix} \right) \quad (1.2)$$

Chaque couche est caractérisée par deux éléments. La première caractéristique est les poids de la couche qui correspondent à une matrice multiplicative $W \in \mathbb{R}^{n_\ell \times n_u}$ ainsi qu'un vecteur additif $b \in \mathbb{R}^{n_\ell}$. Les lignes de la matrice et du vecteur représentent les pondérations de chaque neurone et son biais. Le second élément est sa fonction d'activation $\sigma_\ell : \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_\ell}$. En effet il est fréquemment considéré qu'une même fonction est appliquée terme à terme à l'ensemble des neurones de la couche.

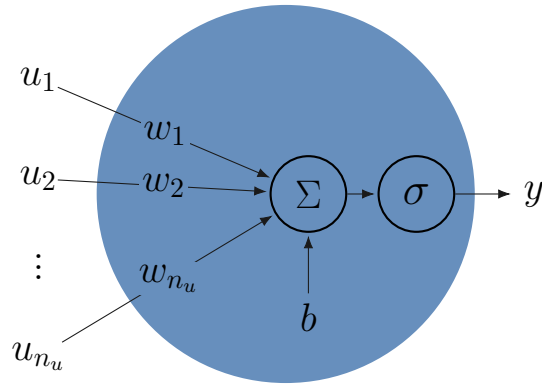


FIGURE 1.1 – Représentation d’un neurone : un neurone artificiel calcule la sortie d’une fonction souvent non-linéaire d’une somme pondérée de ses entrées

Les entrées d’un neurone peuvent être des entrées extérieures ou bien les sorties d’autres neurones. Des réseaux de neurones peuvent ainsi être construits à partir d’une succession de couches. La profondeur du réseau correspond au nombre de couches tandis que sa largeur est reliée au nombre de neurones de chaque couche. Communément, la dernière couche d’un réseau est appelée couche de sortie tandis que les autres sont appelées couches cachées. Il est aussi commun de désigner le vecteur d’entrée comme couche d’entrée. La largeur des couches cachées ne possède aucune limitation théorique à l’inverse de la couche de sortie dont le nombre de neurones est identique au nombre de sorties.

Les neurones et les couches peuvent être combinés de bien des façons, néanmoins une architecture des plus utilisées est le perceptron multicouche ou MLP pour l’anglais *multilayer perceptron*. La structure basique du MLP est construite en disposant les couches de neurones les unes après les autres, une couche prend uniquement pour entrée les sorties de la couche précédente ou une éventuelle entrée extérieure. On parle également de couche entièrement connectée (ou communément *fully connected*) lorsque les neurones de la couche sont connectés à tous ceux de la couche suivante. Un exemple de structure à deux couches cachées, n entrées et m sorties est illustré par la figure 1.2. De manière générale, la forme mathématique décrivant le comportement des réseaux comportant L couches successives entièrement connectées est :

$$y = \sigma_L(W_L(\dots \sigma_1(W_1 u + b_1) \dots) + b_L) \quad (1.3)$$

ou encore :

$$\begin{cases} h_0 = u \\ h_\ell = \sigma_\ell(W_\ell h_{\ell-1} + b_\ell) \quad \text{pour } \ell = 1, \dots, L \\ y = h_L \end{cases} \quad (1.4)$$

De par la structure du MLP présenté, l’information se propage uniquement du vecteur d’entrée à la sortie, c’est pourquoi ces types de réseaux sont référencés comme réseaux de neurones à propagation directe ou FNN pour *Feedforward Neural Network*. Il n’y a pas de lien de rétroaction pour lequel la sortie d’au moins un neurone alimente sa propre entrée après une ou plusieurs action de mémorisation. Lorsque les réseaux sont étendus pour inclure des boucles de rétroaction, ils sont alors appelés réseaux de neurones récurrents ou RNN pour *Recurrent Neural Networks*.

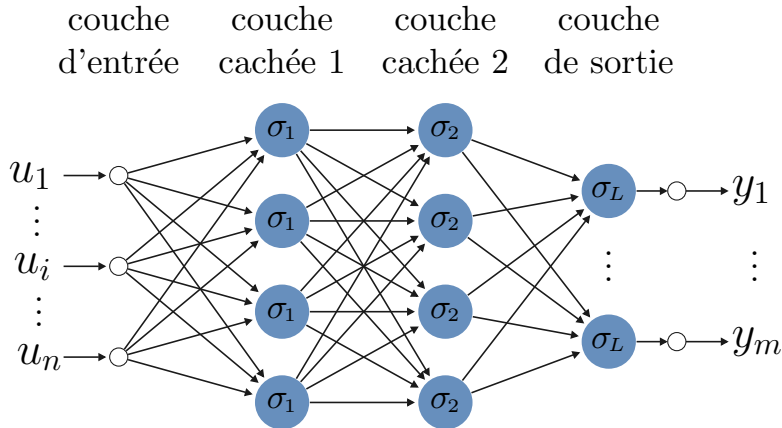


FIGURE 1.2 – Représentation d'un MLP

Théoriquement, la topologie de réseaux la plus générale est celle entièrement récurrente où les sorties de tous les neurones sont connectées aux entrées de tous les neurones (voir l'exemple de la figure 1.3 pour un réseau avec une seule couche cachée). Une infinité de structure peut alors être imaginée et représentée par ce formalisme en fixant certains poids de connexion à zéro pour exprimer l'absence de connexion. Néanmoins malgré la puissance de modélisation irréfutable d'une grande structure complexe, un compromis doit être trouvé au regard des difficultés d'apprentissage et d'utilisation que rencontrera un tel réseau.

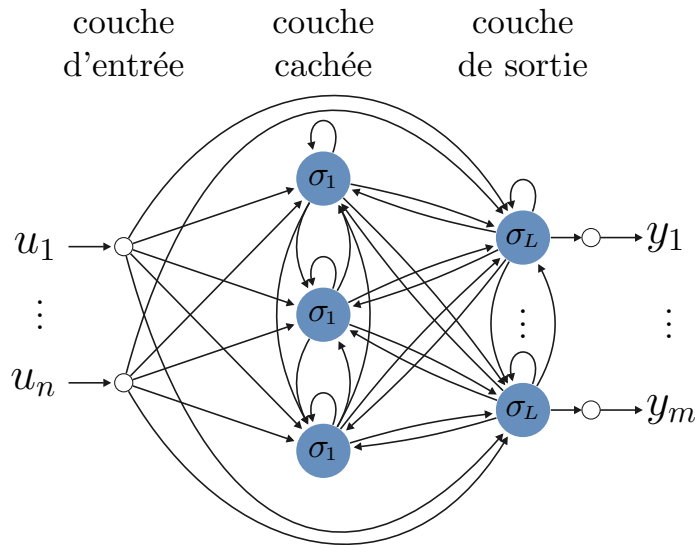


FIGURE 1.3 – Représentation d'un MLP entièrement récurrent

1.2.2 Introduction à l'identification

Du point de vue de l'apprentissage, la modélisation de systèmes dynamiques se rapproche de la prédiction de séries temporelles. En effet, les deux problématiques consistent à élaborer des modèles fondés sur des données antérieures afin de les utiliser pour des situations futures. Caractériser un système signifie identifier tous les différents facteurs physiques qui influencent le

comportement du processus. Cela se traduit en particulier par la détermination des paramètres statiques et dynamiques qui régissent le système. Dans le contexte de la modélisation, le terme dynamique indique le caractère dépendant du temps du processus. Les processus dynamiques peuvent être classés en systèmes à temps continu et en systèmes à temps discret. Puisque les réseaux de neurones sont entraînés sur des données échantillonnées à des intervalles de temps discrets, ce sont principalement les systèmes à temps discret qui sont considérés dans cette étude. De façon générale, un signal x à l'instant d'échantillonnage k est dénoté $x(k)$.

En fonction du niveau de connaissance disponible *a priori* concernant le système, l'identification peut être abordée de différentes manières. Nous nous focaliserons sur les cas où le comportement physique du processus n'est que peu ou pas connu, ainsi l'identification se concentrera exclusivement sur les données expérimentales selon une approche de modélisation en *boite noire*. La philosophie de ce genre de méthode est en opposition aux modélisations en *boite blanche* où un modèle purement physique est utilisé, ou encore de la modélisation en *boite grise* pour laquelle la compréhension partielle du fonctionnement interne du système est utilisée dans le but d'améliorer la modélisation empirique.

L'identification de système utilise des méthodes statistiques de régression temporelle pour construire un modèle mathématique à partir de différentes mesures. Une technique d'approche se fondant sur des influences externes et des mesures du comportement du système, consiste à déterminer la relation qui existe entre elles sans entrer dans le détail de ce qui se passe réellement à l'intérieur du système. Par ce type d'approche en *boite noire*, l'identification repose donc exclusivement sur les entrées de stimuli et les réactions de sortie. Ainsi, le système inconnu est caractérisé par une base de données finie constituée de paires de vecteurs d'entrée-sortie :

$$Z^{dt} = \left\{ u(k), y(k) \mid k = 1, \dots, N_{dt} \right\} \quad (1.5)$$

où $u(k) \in \mathbb{R}^{n_u}$ est un vecteur d'entrée du réseau de neurones tandis que $y(k) \in \mathbb{R}^{n_y}$ est le vecteur de sortie mesurée correspondant pour un même instant donné. L'apprentissage considéré dans ce travail ne suppose donc aucune connaissance de la structure interne du processus à identifier.

L'objectif de l'identification est d'élaborer un modèle qui, lorsqu'il sera soumis aux mêmes entrées u que le système réel, produira des sorties y_{est} similaires à celles mesurées y . Dans de nombreux cas pratiques, le comportement du système peut être caractérisé en considérant une fenêtre d'entrées passées \bar{u} combinée à une fenêtre de sorties passées \bar{y} du système. Plus formellement, les vecteurs d'entrée-sortie augmentés exploités pour l'identification sont définis pour $n_{in} \in \mathbb{N}$ et $n_{out} \in \mathbb{N}$ tels que :

$$\bar{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k-n_{in}) \end{bmatrix} \quad \text{et} \quad \bar{y}(k) = \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-n_{out}) \end{bmatrix} \quad (1.6)$$

avec $\bar{u}(k) \in \mathbb{R}^{m_u}$ de dimension $m_u = n_u(n_{in} + 1)$ et $\bar{y}(k) \in \mathbb{R}^{m_y}$ de dimension $m_y = n_y n_{out}$. Dans le cas linéaire et invariant, la relation entre les entrées et les sorties est l'équation de récurrence [Söderström et Stoica, 1989] :

$$y(k) + a_1 y(k-1) + \dots + a_{n_{out}} y(k-n_{out}) = b_0 u(k) + b_1 u(k-1) + \dots + b_{n_{in}} u(k-n_{in}) \quad (1.7)$$

Le problème d'identification peut également être vue comme un problème de prédiction où il s'agit de déterminer la sortie estimée $y_{est}(k) \in \mathbb{R}^{n_y}$ future selon les observations précédentes :

$$y_{est}(k) = -a_1 y(k-1) - \dots - a_{n_{out}} y(k-n_{out}) + b_0 u(k) + b_1 u(k-1) + \dots + b_{n_{in}} u(k-n_{in}) \quad (1.8)$$

En se basant sur ce raisonnement linéaire, il est possible de l'étendre dans le cadre de la modélisation de systèmes non-linéaires. L'idée demeure que la connaissance d'une fenêtre d'entrées passées du système combinée à une fenêtre de sorties passées fournit suffisamment d'informations pour caractériser l'état du système. Parmi d'autres modèles de régression, ce concept est incarné dans le domaine de l'automatique par les structures de modèles NARX pour *Non-linear AutoRegressive with eXternal input* et NOE pour *Nonlinear Output Error* [Jung, 1998]. Il est donc respectivement supposé pour les deux modèles, que la sortie estimée du système dynamique peut être décrite à chaque temps d'échantillonnage comme une fonction des entrées et des sorties mesurées passées :

$$y_{est}(k) = f_{\text{NARX}}(u(k), \dots, u(k-n_{in}), y(k-1), \dots, y(k-n_{out})) \quad (1.9)$$

ou selon une fonction des entrées et des sorties estimées passées :

$$y_{est}(k) = f_{\text{NOE}}(u(k), \dots, u(k-n_{in}), y_{est}(k-1), \dots, y_{est}(k-n_{out})) \quad (1.10)$$

1.2.3 Identification neuronale

Pour l'identification de système il est alors possible d'utiliser les techniques d'intelligence artificielle telles que le *machine learning*. Les algorithmes d'apprentissage sont employés selon un mode supervisé sur le jeu de données pour lequel il s'agit d'obtenir une régression des séries temporelles. L'emploi des réseaux de neurones afin d'identifier le processus est alors assez direct puisqu'ils constituent une classe de modèles dynamiques discrets pour lesquelles il existe une relation non-linéaire entre les entrées et les sorties. Dans ce cadre de régression de séries temporelles, la couche de sortie des réseaux est généralement composée de fonctions d'activations linéaires.

La puissance de modélisation des réseaux de neurones permet d'aboutir à des modèles pouvant représenter des non-linéarités complexes comme des comportements de saturation, d'hystérésis, chaotiques ou une combinaison de différents phénomènes. De plus, les réseaux de neurones artificiels ont été largement proposés (voir par exemple [Sjöberg *et al.*, 1995]) et utilisés en partie en raison de leurs capacités d'approximation inhérente. En effet, une propriété bien connue est que les réseaux de neurones à deux couches, avec une fonction d'activation sigmoïde pour la couche cachée et une fonction linéaire pour la couche de sortie sont des approximateurs universels [Hornik *et al.*, 1989]. C'est-à-dire qu'un réseau de ce type contenant suffisamment de neurones, peut approcher avec un quelconque niveau de précision, n'importe quelle fonction non-linéaire continue [Cybenko, 1989, Funahashi, 1989].

Les réseaux de neurones non récurrents (FNN) sont des structures intrinsèquement statiques, leur utilisation pour la modélisation doit cependant tenir compte d'un aspect fondamental qui est la dynamique des systèmes considérés. Afin de doter un réseau neuronal statique de propriétés dynamiques, le modèle doit comporter un terme de mémoire qui peut être incorporé principalement selon deux méthodes [Nelles, 2020, Patan, 2008] :

- L'insertion d'un effet de mémoire extérieur au réseau
- La considération implicite de la dynamique par l'utilisation de connexions récurrentes

L'usage de la première stratégie d'externalisation de la dynamique permet alors à des structures statiques telles que les FNN de réaliser l'approximation de séquences spatio-temporelles de données. Cette capacité leur est artificiellement procurée en fournissant aux réseaux une séquence d'entrées et de sorties passées comme évoquée précédemment pour le NARX (1.9). La méthode d'externalisation de la dynamique n'est pas destinée exclusivement aux FNN et peut être utilisée en complément de la seconde méthode qui considère des réseaux récurrents (RNN). La concaténation des signaux retardés qui forme les séquences temporelles est appelée TDL pour l'anglais *Tapped Delay Line*.

Définition 1.1 : *Tapped Delay Line* (TDL)

Soit deux entiers positifs $n_1 \leq n_2$, la $TDL(n_1 : n_2)$ représente le système de concaténation de retards qui a un signal $s(k) \in \mathbb{R}^{n_s}$ associe un vecteur de signaux retardés $\bar{s}(k) = [s^T(k-n_1) \cdots s^T(k-n_2)]^T \in \mathbb{R}^{m_s}$ de dimension $m_s = n_s(n_2 - n_1 + 1)$.

La modélisation de systèmes dynamiques par réseaux de neurones distingue généralement deux approches associées aux méthodes évoquées. Ces approches abordent l'identification par des réseaux FNN et RNN d'un point de vue général, même si les démarches sont illustrées à l'aide des modèles NNARX et NNOE qui seront présentés plus en détails dans la section 1.4. Les terminologies de ces structures particulières sont des extensions neuronales des modèles NARX et NOE présentées par [Ljung, 1998] et reprises par [Nørgård et al., 2000]. Notons qu'il est également possible d'utiliser les termes *modèle série-parallèle* pour la structure FNN et *modèle parallèle* pour la structure RNN issus de [Narendra et Parthasarathy, 1990].

Approche FNN (dynamique extérieur au réseau). Dans ce type d'approche, les sorties précédentes du système réel sont souvent fournies au réseau en plus des entrées afin qu'il puisse générer les sorties estimées. Ainsi la $TDL(0 : n_{in})$ est appliqué à l'entrée $u(k)$ et la $TDL(1 : n_{out})$ est utilisée pour la sortie mesurée $y(k)$. Un exemple immédiat de cette catégorie est une extension neuronale du NARX (1.9) dénommé *Neural Network AutoRegressive with eXternal input* (NNARX) :

$$y_{nn}(k) = f_{\text{NNARX}}\left(u(k), \dots, u(k-n_{in}), y(k-1), \dots, y(k-n_{out})\right) \quad (1.11)$$

Comme le montre la figure 1.4, ce type de réseaux ne présente pas de récurrence c'est pourquoi il correspond à une structure FNN. La principale caractéristique de ces réseaux neuronaux est qu'ils ne possèdent aucune dynamique intrinsèque au sein de leur structure. Une fonction non-linéaire statique relie l'espace des entrées à l'espace des sorties de façon déterministe.

L'avantage de l'apprentissage des NNARX et des FNN de manière plus générale est qu'il est relativement rapide et efficace. L'un des inconvénients bien connus de cette approche est qu'elle ne permet de modéliser que des systèmes pour lesquels la sortie ne dépend que d'un nombre fini de valeurs des entrées et sorties passées. De plus, cette représentation souffre d'une grande sensibilité aux dimensions des fenêtres de retards et d'une vulnérabilité inévitable aux entrées et sorties bruitées. En effet, la stabilité de l'apprentissage des FNN n'est pas garantie, et la prédiction peut changer de manière significative lorsque les entrées sont légèrement modifiées. Lorsque les FNN sont utilisés comme systèmes dynamiques ou prédicteurs à long terme, une

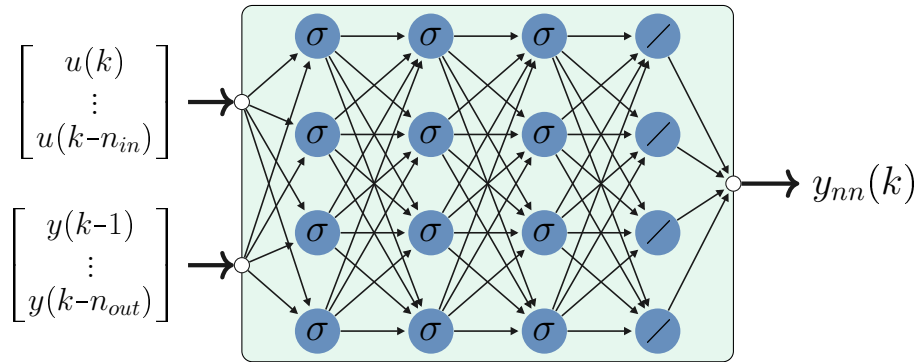


FIGURE 1.4 – Identification neuronale par NNARX

boucle de rétroaction de sortie est créée artificiellement pour fournir les sorties passées au réseau (ceci coïncide alors avec le schéma de la figure 1.5). Cependant, cette connexion de récurrence, qui n'a pas été prise en compte lors de l'entraînement, provoque des erreurs qui peuvent s'accumuler, ce qui ne garantit pas l'utilisation des FNN comme modèle dynamique.

Approche RNN (dynamique interne au réseau). Une autre façon de représenter les systèmes dynamiques consiste à incorporer des connexions de rétroaction sous la forme de réseaux récurrents qui possèdent une dynamique propre. Un exemple de réseau de cette catégorie est le *Neural Network Output Error* (NNOE) qui, comme le montre la figure 1.5, utilise les sorties estimées précédentes comme entrée du modèle afin de prédire les futures sorties :

$$y_{nn}(k) = f_{\text{NNOE}}\left(u(k), \dots, u(k-n_{in}), y_{nn}(k-1), \dots, y_{nn}(k-n_{out})\right) \quad (1.12)$$

Cette structure récurrente englobe une classe plus large de systèmes non-linéaires puisque l'historique du système entier peut être intrinsèquement intégré dans la dynamique du modèle. Les RNN fournissent donc des modèles d'identification universels dans le sens où ils peuvent approximer uniformément tout système dynamique non-linéaire multi-entrées multi-sorties (MIMO) sur un intervalle de temps fini et pour tout signal d'entrée continu et borné [Garzon et Botelho, 1999, Jin *et al.*, 1999].

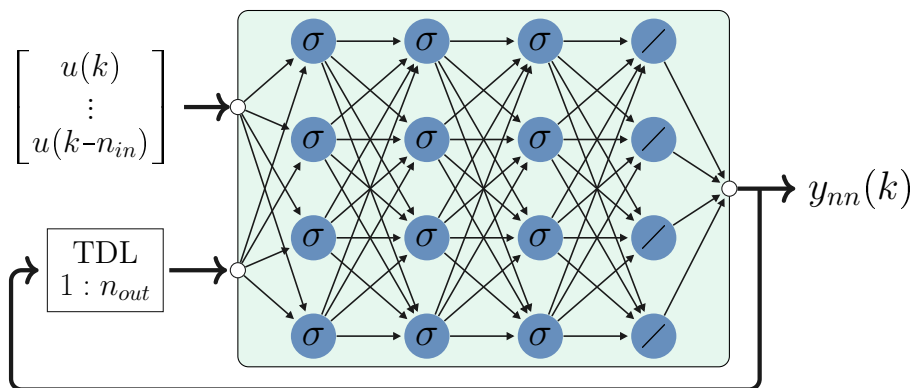


FIGURE 1.5 – Identification neuronale par NNOE

De plus, la structure NNOE est moins vulnérable aux données bruitées par le fait que

son vecteur d'entrée comporte ses propres sorties précédentes, l'accumulation des erreurs d'estimations sera donc minimisée pendant les phases d'apprentissage. Bien qu'ils soient plus stables du point de vue de l'apprentissage, les algorithmes d'apprentissages des NNOE et des structures RNN, de façon générale, présentent des difficultés à converger pour des données provenant de systèmes très complexes.

1.2.4 Architectures générales des réseaux de neurones

L'architecture du réseau est la description de ses couches telle que leur nombre, leurs types, leurs tailles, leurs propriétés, leurs entrées et leurs sorties. De nombreuses architectures ont été développées et popularisées dans la littérature au cours des décennies. Chaque structure possède ses avantages et ses inconvénients ce qui rend son utilisation plus ou moins adaptée aux types de problématique adressés. Déterminer l'architecture la plus favorable pour une application donnée est un exercice complexe dont les conclusions sont souvent nuancées. Le lecteur intéressé peut se référer aux nombreux ouvrages sur le sujet tels que [Hagan *et al.*, 2014] ou [Goodfellow *et al.*, 2016].

Dans notre étude, la problématique adressée est l'apprentissage de réseaux de neurones selon une méthode supervisée afin de parvenir à une régression temporelle de systèmes dynamiques. Ainsi de façon non exhaustive, nous pouvons citer les principales architectures qui peuvent être impliquées dans ce cadre d'étude et leurs domaines d'application privilégiés :

- ***Multilayer Perceptron (MLP)*** : Les perceptron multicouches ont été présentés dans la section 1.2.1 précédente, cette architecture peut être organisée selon des MLP récurrents même si le terme MLP seul désigne souvent des réseaux ne comportant pas de connexion de récurrence, c'est-à-dire des FNN. Dans ce type de structure les neurones sont donc arrangés séquentiellement en couches et le réseau peut comporter une à plusieurs couches cachées. Les MLP sont reconnus pour leur large spectre d'application dans des problématiques faisant intervenir des domaines de connaissance bien différents. Cette architecture est donc l'une des plus polyvalentes en terme d'applicabilité pour de la régression ou de la classification. Son champ d'application couvre des domaines tels que la reconnaissance vocale ou d'image, l'identification ou le contrôle de processus, la prédiction de séries temporelles ou l'optimisation de système.
- ***Convolutional Neural Network (CNN)*** : Les réseaux convolutifs [LeCun *et al.*, 1989] sont des réseaux de neurones de type FNN dont au moins une couche emploie l'opération de convolution à la place de la multiplication matricielle générale. Ces types de réseau sont spécialisés pour traiter les données dont l'information est agencée sous formes de grilles de différentes dimensions. Un exemple éloquent dans le cas des images est qu'une partie de l'image est interprétée au moyen d'une convolution sur une grille 2-D de pixels ce qui permet d'identifier ses caractéristiques [Krizhevsky *et al.*, 2012]. Dans le cadre des séries temporelles, une grille 1-D analyse des échantillons de données à intervalles de temps réguliers. Par l'opération de convolution, les paramètres de traitement sont réduits ce qui présente un intérêt certain lors du traitement de données de grandes dimensions. Les CNN sont largement utilisés pour la détection d'objets dans les images et les vidéos, la classification d'images, l'analyse sémantique, le traitement du langage naturel ou d'image, etc.
- ***Radial Basis Function network (RBF)*** : Les réseaux à base radiale [Broomhead *et*

[Lowe, 1988] sont des réseaux souvent de types FNN, typiquement composés d'une seule couche cachée dont les fonctions d'activation sont des fonctions de base radiale telles que des gaussiennes et d'une couche de sortie linéaire [Haykin, 1999]. Les RBF se distinguent des autres réseaux de neurones par rapport à leur stratégie d'apprentissage qui de façon distincte ajuste les paramètres de la couche cachée à fonctions radiales (centres et largeurs) puis les poids de la couche de sortie linéaire. Les RBF peuvent être employés dans presque tous les types de problèmes résolus par les MLP, c'est-à-dire ceux de régressions telles que la prédiction ou ceux de classifications comme la reconnaissance vocale ou de formes.

- **Long Short-Term Memory (LSTM)** : Les réseaux LSTM [Hochreiter et Schmidhuber, 1997] font référence à la notion selon laquelle les RNN possède à la fois une mémoire à long terme et une mémoire à court terme. Les LSTM sont donc des RNN dont l'objectif est de pouvoir d'apprendre cette dépendance à long terme à l'aide de cellules qui sont communément composées d'une porte d'entrée, une porte de sortie et une porte d'oubli. Ces réseaux sont particulièrement adaptés au traitement de séries temporelles pour lesquelles ils sont devenus un élément essentiel des méthodes de *Deep Learning*.

Bien que toutes ces architectures puissent être exploitées pour identifier des systèmes dynamiques, nous nous intéresserons uniquement dans nos travaux aux MLP et MLP récurrents. En effet, comme nous avons pu le constater lors de notre étude et comme il a également été mis en avant par différentes études comparatives [Ogunmolu *et al.*, 2016, Richard *et al.*, 2019], les MLP représentent un bon compromis entre une architecture simple qui sera utile pour les méthodes d'analyses, et un modèle dynamique proposant de bonnes performances d'identification.

SECTION 1.3

Méthodes d'entraînement des réseaux de neurones

L'entraînement d'un réseau de neurones consiste à déterminer les poids du modèle afin que son comportement soit le plus fidèle à celui recherché, pour notre cas d'usage ceci consiste à identifier le système non-linéaire qu'est le réseau. Cette tâche s'apparente à un processus d'optimisation comportant deux étapes. La première étape correspond à définir mathématiquement les performances comportementales d'un réseau à travers l'indice de performance. La seconde étape consiste à rechercher les paramètres c'est-à-dire les poids afin de minimiser l'indice de performance. En d'autres termes, l'indice de performance constitue le critère d'optimisation tandis que le problème d'optimisation réside dans l'identification du réseau de neurones.

1.3.1 Indice de performance

L'objectif de l'apprentissage est de trouver les paramètres du réseau afin de minimiser l'écart entre le comportement du réseau de neurones et celui désiré. La différence comportementale est mesurée numériquement par l'indice de performance qui n'est autre qu'un nombre réel évalué pour un réseau donné et sur un jeu de données fourni :

$$J(\mathcal{W}, Z^{dt}) \tag{1.13}$$

Le vecteur \mathcal{W} regroupe l'ensemble des poids du réseau (matrices de poids et biais) et Z^{dt} est la base de données (1.5). Dans le cadre de l'apprentissage supervisé, l'indice est directement issu de la comparaison entre les sorties de la base de données et celles estimées par le réseau. Le critère de performance constitue l'élément clé de la fonction de coût qu'il convient de minimiser lors de

la phase d'apprentissage. Dans le cadre de régressions temporelles, deux types de fonctions sont principalement utilisés :

- **Erreur absolue moyenne (MAE)** : Cet indice est une métrique de régression qui mesure l'amplitude moyenne des erreurs dans un ensemble de données prédites, sans tenir compte de leurs directions. En d'autres termes, il s'agit d'une moyenne des différences absolues entre les prédictions et les résultats attendus :

$$J_{mae}(\mathcal{W}, Z^{dt}) = \frac{1}{N_{dt}n_y} \sum_{k=1}^{N_{dt}} \sum_{j=1}^{n_y} |y^j(k) - y_{nn}^j(k)| \quad (1.14)$$

où $y^j(k)$ et $y_{nn}^j(k)$ représentent le $j^{\text{ème}}$ élément du vecteur associé.

- **Erreur quadratique moyenne (MSE)** : C'est une des métriques de régression les plus couramment utilisées qui correspond à la différence quadratique moyenne entre les prédictions et les résultats désirés. Les écarts individuels sont cette fois-ci élevés au carré avant d'être additionnés :

$$J_{mse}(\mathcal{W}, Z^{dt}) = \frac{1}{2N_{dt}n_y} \sum_{k=1}^{N_{dt}} [y(k) - y_{nn}(k)]^T [y(k) - y_{nn}(k)] \quad (1.15)$$

***Remarque 1.1.** Deux autres critères peuvent être définis, le SAE (Sum of Absolute Errors) et SSE (Sum of Squared Errors) qui correspondent respectivement aux deux indices présentés ci-dessus sans appliquer les coefficients aux sommes. Dans la pratique, se sont souvent ces critères qui sont utilisés par les algorithmes d'optimisation. En effet les coefficients ont un rôle de normalisation qui est utile pour les comparaisons de performances mais ne changent pas le problème d'optimisation.*

Dans la suite des travaux, nous considérerons uniquement l'erreur quadratique moyenne (MSE). En effet les caractéristiques les plus attrayantes de cet indice sont ses bonnes propriétés mathématiques qui rendent le calcul de sa dérivée plus simple en comparaison avec le MAE. Ainsi lors de l'optimisation, la pertinence de l'indice MSE se démontre par la facilité avec laquelle une règle de mise à jour des poids peut être obtenue, lorsque celle-ci est fondée sur un algorithme de descente de gradient et plus particulièrement lorsqu'elle nécessite le calcul de la matrice jacobienne.

Un autre aspect intéressant du critère MSE porte sur le bruit de mesure que comporte les signaux utilisés pour l'apprentissage. Ce bruit est souvent supposé gaussien. Néanmoins l'impact du non-respect de cette hypothèse se retrouve limité par le MSE et un modèle souvent suffisamment précis peut être obtenu. De plus, dans les cas de plusieurs sorties, si la connaissance du bruit permet d'affirmer que sa variance est suffisamment différente entre les sorties et que la corrélation croisée est significative alors l'indice MSE peut être modifié en fonction [Nørgård et al., 2000] :

$$J_{mse,\Gamma}(\mathcal{W}, Z^{dt}) = \frac{1}{2N_{dt}n_y} \sum_{k=1}^N [y(k) - y_{nn}(k)]^T \Gamma^{-1} [y(k) - y_{nn}(k)] \quad (1.16)$$

où Γ est la matrice de covariance du bruit des sorties du processus. La minimisation de ce critère modifié (1.16) correspond à l'estimateur du maximum de vraisemblance en supposant que le

bruit a une distribution gaussienne et que Γ est connue. Dans un souci d'implémentation, il est souvent plus simple de remanier les données pour l'entraînement lors d'un pré-traitement plutôt que de modifier l'indice de performance des algorithmes. La matrice de covariance étant par construction symétrique et définie positive, elle peut être factorisée selon $\Gamma^{-1} = \underline{\Gamma}^T \underline{\Gamma}$. Ainsi pour un jeu de données comprenant les sorties $y'(k)$, $k = 1, \dots, N_{dt}$, il est souvent plus simple de raisonner sur les sorties transformées $y(k) = \underline{\Gamma}y'(k)$ et le critère MSE sans pondération (1.15).

1.3.2 Rétropropagation de l'erreur

La phase d'apprentissage exploite l'indice de performance dans le but de déterminer les coefficients de poids du réseau qui minimisent ce critère :

$$\hat{\mathcal{W}} = \arg \min_{\mathcal{W}} J(\mathcal{W}, Z^{dt}) \quad (1.17)$$

Un avantage fondamental des réseaux de neurones est l'efficacité de la stratégie de modification des poids qui peut être mise en place en fonction de l'erreur de prédiction du modèle. L'apprentissage est fondé sur l'algorithme essentiel de rétrogradation ou *backpropagation* [Rumelhart et al., 1986a, Rumelhart et al., 1986b] qui est une méthode qui se base sur le calcul du gradient de la fonction de coût par rapport aux paramètres du réseau. L'idée consiste à calculer des dérivées pour permettre à l'information du coût de remonter à travers le réseau et ainsi obtenir une directive de rectification des poids de proche en proche. Cette technique pour propager l'information est très générale et elle peut être utilisée pour calculer d'autres dérivées utiles que le gradient comme le jacobien de la fonction.

En tirant profit du calcul du gradient, l'optimisation des poids du réseau peut être effectuée de façon itérative à l'aide des algorithmes d'optimisation de descente du gradient. Dans la version la plus simple, la modification des poids est effectuée à chaque itération selon la formule suivante :

$$\mathcal{W}(i+1) = \mathcal{W}(i) - \eta \nabla J(\mathcal{W}(i)) \quad (1.18)$$

où $\mathcal{W}(i)$ désigne le vecteur de l'ensemble des poids pour l'itération actuelle, η est le pas d'apprentissage (ou *learning rate*) et $\nabla J(\mathcal{W}(i))$ correspond au gradient de l'indice de performance évalué pour le vecteur de poids $\mathcal{W}(i)$. À partir de cette perspective, de nombreux autres algorithmes d'optimisation du premier ordre ou d'ordre plus élevé ont montré d'excellentes performances pour l'entraînement de réseaux de neurones [Sun et al., 2019]. Parmi les nombreux algorithmes d'apprentissage [Hagan et al., 1996], nous pouvons mentionner ceux envisagés dans ces travaux et disponibles dans la *Deep Learning Toolbox* de MATLAB :

- *Resilient Backpropagation* (RPROP) [Riedmiller et Braun, 1993]
- *Momentum Backpropagation* [Hagan et al., 1996] et *Adaptive Learning Rate Backpropagation* [Vogl et al., 1988] qui peuvent être associés
- *Conjugate Gradient Backpropagation* (CG) et ses dérivés [Hagan et al., 1996] comme le *Scaled Conjugate Gradient Backpropagation* (SCG) [Møller, 1993]
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) [Dennis Jr et Schnabel, 1996]
- Levenberg-Marquardt (LM) [Hagan et Menhaj, 1994]

Ces méthodes d'optimisation emploient le gradient ou le jacobien de la performance du réseau par rapport aux poids. Lorsque les réseaux sont récurrents, le calcul du gradient de l'erreur est plus complexe et il est approximé par des algorithmes tels que *Real-Time-Recurrent-Learning*

(RTRL) [Williams et Zipser, 1989] ou *BackPropagation-Through-Time* (BPTT) [Werbos, 1990].

L'apprentissage des réseaux de neurones récurrents est un problème plus complexe que celui des réseaux qui ne le sont pas. Une difficulté couramment rencontrée est l'évolution exponentielle du gradient à travers les couches [Pascanu *et al.*, 2013]. Les gradients supérieurs à 1 ont tendance à exploser (*exploding gradient*) ce qui engendre des problèmes de stabilité numérique. Tandis que les gradients inférieurs à 1 ont tendance à disparaître à travers les couches (*vanishing gradient*) ce qui rend le processus de mise à jour de certains poids inefficace. Ces phénomènes ont donc pour conséquence de complexifier l'apprentissage des RNN qui comportent des dépendances temporelles de long terme.

Malgré les méthodes employées, l'apprentissage neuronal n'est pas précisément une procédure d'optimisation traditionnelle [Goodfellow *et al.*, 2016]. La principale raison est que l'apprentissage d'un modèle s'effectue la plupart du temps à travers une certaine mesure de performance définie par rapport à l'ensemble des données de test. Néanmoins, la minimisation de ce critère ne constitue pas une fin en soi, mais bien une manière indirecte d'optimiser le comportement général du modèle. Ainsi, l'objectif de l'apprentissage ne peut se résumer à l'obtention d'un indice de performance faible puisqu'il est, en effet, fréquent de pouvoir trouver un modèle qui se comporte de façon inadaptée et qui pourtant engendre une meilleure performance que d'autres modèles aux comportements plus appropriés. La notion de généralisation présentée dans la section suivante aborde cette problématique.

1.3.3 Généralisation du réseau

La finalité de l'entraînement d'un réseau de neurones est qu'il parvienne à extraire l'information pertinente présente dans les données pour converger vers le comportement recherché. Néanmoins le processus se fonde sur une connaissance partielle qui est celle contenue dans la base de données. Le concept de généralisation évoque la capacité du modèle à être fiable dans de nouvelles situations pour lesquelles il n'a pas été entraîné.

La généralisation du modèle est étroitement liée au phénomène bien connu de sur-apprentissage ou l'anglicisme *overfitting*. Cet événement survient lorsque le modèle devient trop spécifique aux données d'apprentissage et surtout aux bruits qu'elles contiennent c'est-à-dire de façon générale aux points qui ne représentent pas les réelles propriétés du système. Un tel réseau présentera alors de pauvres performances en termes d'interpolation ou d'extrapolation.

Dans le but d'estimer l'aptitude de généralisation du réseau et de pouvoir l'améliorer, la méthode considérée implique que deux jeux de données supplémentaires soient consacrés pour remplir ses différentes fonctions. Compte tenu de la quantité limitée de données disponible pour le processus d'apprentissage, la base de données est divisée en trois sous-ensemble : un ensemble d'apprentissage utilisé par l'algorithme d'optimisation des poids, un ensemble de validation et un ensemble de test.

Ensemble de test. Afin d'estimer la capacité de généralisation d'un modèle spécifique, il convient d'évaluer ses performances sur des données qu'il n'a jamais rencontrées auparavant. Ce rôle est assuré par l'ensemble de test qui est indépendant de la procédure d'entraînement. Une fois l'apprentissage effectué, l'indice de performance peut être évalué sur cet ensemble afin

de fournir un indicateur des capacités de généralisation du réseau. Pour que cette estimation soit valide, il est important qu'il soit représentatif de l'ensemble des situations pour lesquelles le réseau sera utilisé, de même l'ensemble de test ne doit jamais prendre part au processus d'optimisation.

Ensemble de validation et arrêt prématuré de l'apprentissage. Pour garantir de bonnes propriétés de généralisation et détecter lorsque celles-ci se détériorent, un ensemble de validation est isolé. Par la méthode d'arrêt prématuré de l'apprentissage (*early stopping*), l'indice de performance du réseau sur ce jeu de données spécifique est surveillé au cours des itérations du processus d'optimisation des poids. L'apprentissage est arrêté lorsque l'erreur de validation n'est plus minimum depuis un certain nombre d'itérations, les poids produisant l'erreur de validation la plus faible sont alors utilisés pour définir le réseau de neurones final.

SECTION 1.4

Topologies des réseaux de neurones pour l'identification

La section suivante décrit les différents types d'architectures de réseaux de neurones et les fonctions d'activation qui sont employées dans ces travaux pour l'identification des systèmes.

1.4.1 Fonctions d'activations

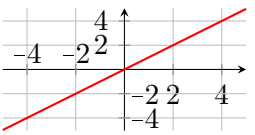
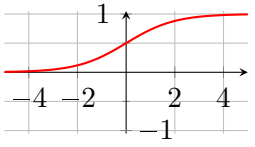
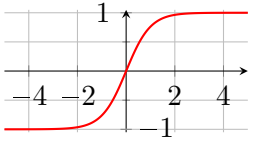
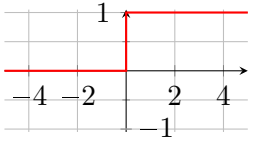
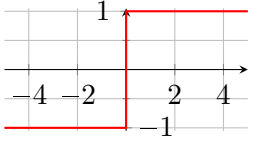
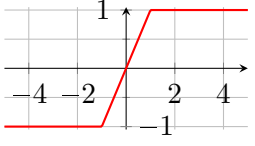
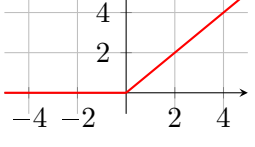
Fondamentalement, les fonctions d'activations d'un réseau de neurones peuvent être de toute forme, néanmoins le plus souvent elles sont dérivables afin d'être compatibles avec les algorithmes d'apprentissage fondés sur le gradient. Le tableau 1.1 illustre des fonctions d'activations communément employées. Historiquement, les fonctions sigmoïde et d'Heaviside ont été utilisées car leur comportement rappelle quelques peu les activations des neurones biologiques. Néanmoins l'expression de la dérivée de la sigmoïde offre une efficacité d'apprentissage importante. À leur image, les activations de la tangente hyperbolique et signe procèdent d'une façon similaire mais produisent une sortie symétrique centrée sur zéro. Les fonctions de saturation symétrique unitaire et celle devenu très populaire RELU (*Rectified Linear Unit*) sont appréciées pour la simplicité de leur dérivée¹ et présentent une non-linéarité parfois suffisante pour que le réseau soit expressif.

1.4.2 Topologie des NNARX

Le modèle NNARX (*Neural Network AutoRegressive with eXternal input*) est une architecture de la catégorie des réseaux FNN, il se compose principalement de deux éléments qui sont des TDL et un réseau MLP. Le MLP approxime la fonction non-linéaire du système et les TDL appliquées aux entrées et aux sorties, introduisent la dynamique dans le modèle NNARX. La topologie d'un NNARX à L couches cachées est présentée par la figure 1.6. En regroupant les poids des neurones d'une même couche selon une notation matricielle par entrée, il est alors possible de décrire le comportement dynamique du réseau.

1. Même si les fonctions ne sont pas dérivables en tout point ce qui peut engendrer des problèmes avec les algorithmes du gradient

TABLE 1.1 – Exemples de fonctions d'activation

Nom	Équation	Graphique
linéaire (<i>purelin</i>)	$f(x) = x$	
sigmoïde (<i>logsig</i>)	$f(x) = \frac{1}{1 + \exp(-x)}$	
tanh (<i>tansig</i>)	$f(x) = \tanh(x)$	
Heaviside (<i>harlim</i>)	$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$	
signe (<i>harlims</i>)	$f(x) = \begin{cases} -1 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$	
saturation (<i>satlins</i>)	$f(x) = \begin{cases} -1 & \text{si } x < -1 \\ x & \text{si } -1 \leq x \leq 1 \\ 1 & \text{si } x > 1 \end{cases}$	
RELU (<i>poslin</i>)	$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$	

Définition 1.2 : NNARX

Un réseau de type NNARX à L couches cachées se définit par la dynamique :

$$\begin{cases} h_0(k) = [\bar{y}^T(k) \bar{u}^T(k)]^T \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 1, \dots, L \\ y_{nn}(k) = W_y h_L(k) + b_y \end{cases} \quad (1.19)$$

où $\bar{y}(k) = [y^T(k-1) \dots y^T(k-n_{out})]^T$ et $\bar{u}(k) = [u^T(k) \dots u^T(k-n_{in})]^T$.

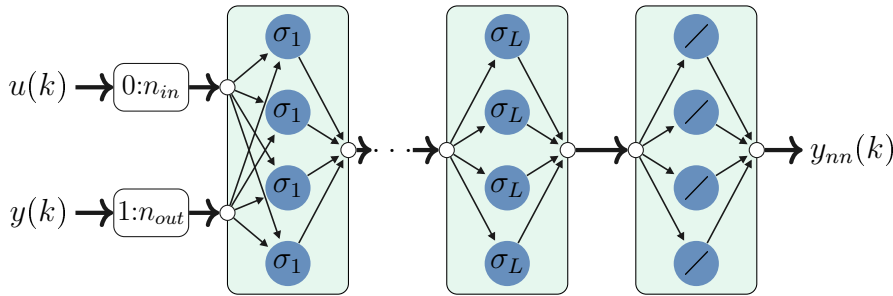


FIGURE 1.6 – Topologie du NNARX

Les réseaux NNARX présentent principalement les caractéristiques suivantes :

- La modélisation est une relation non-linéaire statique entre les entrées et les sorties passées :

$$y_{nn}(k) = f_{\text{NNARX}}(u(k), \dots, u(k-n_{in}), y(k-1), \dots, y(k-n_{out})) \quad (1.20)$$

- L'apprentissage converge rapidement et efficacement, de plus les algorithmes sont facilement parallélisables puisque les signaux sont traités de manière distincte et non comme une continuation des valeurs précédentes de la série temporelle.
- L'emploi de ce modèle pour la prédiction est explicite et le prédicteur obtenu est toujours stable. Néanmoins il peut être sensible aux bruits et autres perturbations.
- Au cours de l'entraînement, seule la prédiction au pas suivant est optimisée plutôt que l'erreur de simulation globale. Aucune performance n'est donc garantie sur plusieurs pas de temps lorsque le réseau est rebouclé sur lui-même et le modèle dynamique ainsi obtenu peut également être instable.

1.4.3 Topologie des NNOE

Le modèle NNOE (*Neural Network Output Error*) correspond à l'architecture associée au NNARX mais dans la catégorie des RNN. Une fonction d'approximation non-linéaire est apportée par un MLP qui traite les entrées et dont la dynamique dépend des sorties précédemment estimées. L'utilisation de TDL n'est pas imposée mais souvent indispensable pour accorder suffisamment de complexité au modèle. La figure 1.7 illustre la topologie d'un NNOE à L couches cachées.

Définition 1.3 : NNOE

Un réseau de type NNOE à L couches cachées se définit par la dynamique :

$$\begin{cases} h_0(k) = [\bar{y}_{nn}^T(k) \bar{u}^T(k)]^T \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 1, \dots, L \\ y_{nn}(k) = W_y h_L(k) + b_y \end{cases} \quad (1.21)$$

où $\bar{y}_{nn}(k) = [y_{nn}^T(k-1) \dots y_{nn}^T(k-n_{out})]^T$ et $\bar{u}(k) = [u^T(k) \dots u^T(k-n_{in})]^T$.

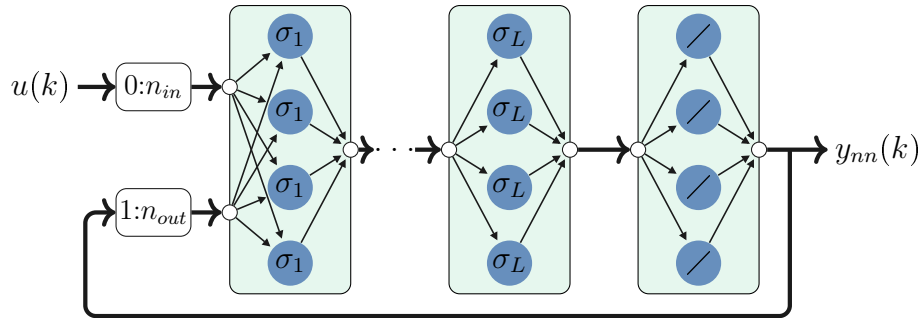


FIGURE 1.7 – Topologie du NNOE

Les réseaux NNOE présentent principalement les caractéristiques suivantes :

- Le modèle permet de représenter tout système dynamique désignant une application entrée-sortie d'ordre fini c'est-à-dire pour lequel il existe une relation :

$$y_{nn}(k) = f_{\text{NNOE}}(u(k), \dots, u(k-n_{in}), y_{nn}(k-1), \dots, y_{nn}(k-n_{out})) \quad (1.22)$$

- L'ordre dynamique du modèle est relié aux choix des fenêtres de retards (par les paramètres n_{in} et n_{out}) tandis que la complexité de la fonction dépend de la structure du réseau.
- Pour l'entraînement, les calculs de gradients de la structure récurrente peuvent devenir fastidieux et demander beaucoup plus de temps. De plus, le problème d'optimisation abordé est plus difficile à résoudre et la convergence de l'algorithme peut être altérée par les phénomènes de *vanishing gradient* ou d'*expoding gradient*.

1.4.4 Topologie générale des SSNN

De façon générale, le terme SSNN (*State-Space Neural Network*) désigne toutes les architectures de réseaux de neurones dont le comportement dynamique peut être exprimé à l'aide d'une représentation d'état de la forme :

$$\begin{cases} x(k+1) = f_{\text{SSNN}}(x(k), u(k)) & \text{équation d'évolution} \\ y_{nn}(k) = g_{\text{SSNN}}(x(k), u(k)) & \text{équation de sortie} \end{cases} \quad (1.23)$$

De telles architectures ont très vite été utilisées dans le domaine de l'automatique par le biais de MLP comme dans [Suykens *et al.*, 1995a, Rivals et Personnaz, 1996, Schenker et Agarwal, 1997, Zamarreño et Vega, 1998] même si l'appellation SSNN n'est pas toujours employée. La

littérature expose de nombreuses formulations de SSNN qui partagent une doctrine commune mais peuvent se différencier par le nombre de couches considéré, l'ajout de termes linéaires, des hypothèses concernant certaines fonctions d'activation ou certains poids, etc. Les architectures les plus communes ont pour origine une structure plus ou moins profonde de MLP non récurrent à laquelle est ajoutée une boucle de récurrence reliant une couche quelconque à la première couche. La sortie de cette couche correspond alors à l'état du système, nous la dénommerons ainsi couche d'état. Lors de la rétropropagation de l'état mais également pour la propagation à la couche suivante, il convient de lui appliquer un retard pur. Les couches précédant l'état décrivent alors l'équation d'évolution de la modélisation tandis que les couches suivantes représentent l'équation de sortie. En suivant ce principe, nous pouvons citer quelques exemples parmi les architectures les plus connues :

- [Suykens *et al.*, 1995a, Suykens *et al.*, 1995b, Suykens *et al.*, 1997] et la théorie des NL_q qui emploie de façon très générique des MLP à q couches pour modéliser le système non-linéaire :

$$\begin{cases} x(k+1) = \sigma_q \left(W_{q-1} \sigma_{q-1} \left(W_{q-2} \dots \sigma_1 \left(W_1^x x(k) + W_1^u u(k) \right) \dots + b_{q-1} \right) + b_q \right) \\ y_{nn}(k) = \underline{\sigma}_q \left(\underline{W}_{q-1} \underline{\sigma}_{q-1} \left(\underline{W}_{q-2} \dots \underline{\sigma}_1 \left(\underline{W}_1^x x(k) + \underline{W}_1^u u(k) \right) \dots + \underline{b}_{q-1} \right) + \underline{b}_q \right) \end{cases} \quad (1.24)$$

La dépendance à l'entrée dans l'équation de sortie est prise en compte en ajoutant une liaison de l'entrée vers la couche suivant l'état qui est sans biais tout comme la première couche.

- [Zamarreño *et Vega*, 1998, Zamarreno *et al.*, 2000, Pulido *et al.*, 2019], les comportements non-linéaires des deux équations décrivant le modèle sont considérés par un MLP à trois couches cachées plus celle de sortie :

$$\begin{cases} x(k+1) = W_2 \sigma_1 \left(W_1^x x(k) + W_1^u u(k) + b_1 \right) \\ y_{nn}(k) = W_4 \sigma_3 \left(W_3^x x(k) + b_3 \right) \end{cases} \quad (1.25)$$

La deuxième couche est rebouclée sur la première afin de constituer l'état du réseau, elle est de plus linéaire et sans biais comme la quatrième couche de sortie.

- [Bendtsen *et Trangbæk*, 2000, Bendtsen *et Trangbæk*, 2002] utilisent les travaux précédents en retirant l'avant-dernière couche, ce qui aboutit à un modèle MLP à deux couches cachées :

$$\begin{cases} x(k+1) = W_2 \sigma_1 \left(W_1^x x(k) + W_1^u u(k) + b_1 \right) \\ y_{nn}(k) = Cx(k) \end{cases} \quad (1.26)$$

La sortie de la seconde couche cachée linéaire correspond à l'état du réseau par son lien de rétroaction. L'équation de sortie est cette fois-ci linéaire puisqu'elle est composée de la couche de sortie qui s'apparente à l'application d'une matrice. Cette architecture est reprise dans [Abbas *et Werner*, 2008a, Abbas *et Werner*, 2008b] où la matrice de l'équation de sortie est imposée afin que l'état du système corresponde à la concaténation des sorties retardées.

- [Haykin, 1999, Patan, 2008, Czajkowski *et al.*, 2014, Luzar *et Czajkowski*, 2016] pour lesquelles la modélisation est réalisée par un MLP ne comportant pas de biais et dont la seule couche cachée est rebouclée sur elle-même afin de constituer l'état du système :

$$\begin{cases} x(k+1) = \sigma_1 \left(W_1^x x(k) + W_1^u u(k) \right) \\ y_{nn}(k) = Cx(k) \end{cases} \quad (1.27)$$

- [Gil *et al.*, 2006, Gil *et al.*, 2013] introduit un comportement hybride au modèle neuronal via une architecture dénommée SSNN affine. L'architecture bénéficie d'une contribution linéaire complétée par la contribution non-linéaire de certaines fonctions d'activation. Par l'extension de cette architecture dans les travaux de [Lachhab *et al.*, 2008], le modèle utilisé est de la forme :

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + A_1\sigma_A(W_Ax(k)) + B_1\sigma_B(W_Bu(k)) \\ y_{nn}(k) = Cx(k) + C_1\sigma_C(W_Cx(k)) \end{cases} \quad (1.28)$$

Le modèle peut être perçu comme un réseau MLP sans biais comportant plusieurs couches cachées. Les premières couches traitent séparément l'état et l'entrée du système, la couche d'état est de plus rebouclée sur elle-même et reliée à l'entrée, enfin les couches suivantes de l'équation de sortie se scindent en un apport linéaire et non-linéaire.

Dans nos travaux, l'architecture SSNN est abordée à l'aide d'une approche générique qui reprend les principaux éléments proposés dans la littérature. La structure des SSNN considérée est illustrée par la figure 1.8 ou sous une forme matricielle équivalente donnée par la figure 1.9. La méthode ne présume pas de la profondeur du réseau, ainsi plusieurs couches cachées peuvent être envisagées. Le réseau comporte différentes liaisons avec l'entrée pour prendre en compte sa dépendance de diverses façons dans le modèle. De plus, la modélisation tient compte d'une probable augmentation de l'entrée u en un vecteur de signaux retardés \bar{u} via une TDL. L'hypothèse majeure de la démarche est que les couches suivant la couche d'état sont linéaires, ce qui impose une relation linéaire pour l'équation de sortie.

Définition 1.4 : SSNN

Un réseau de type SSNN à $L + 1$ couches cachées se définit par la dynamique :

$$\begin{cases} h_0(k) = [x^T(k) \bar{u}^T(k)]^T \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \text{ pour } \ell = 1, \dots, L \\ x(k+1) = Ax(k) + B\bar{u}(k) + W_x h_L(k) + b_x \\ y_{nn}(k) = Cx(k) + D\bar{u}(k) + b_y \end{cases} \quad (1.29)$$

où $\bar{u}(k) = [u^T(k) \dots u^T(k-n_{in})]^T$.

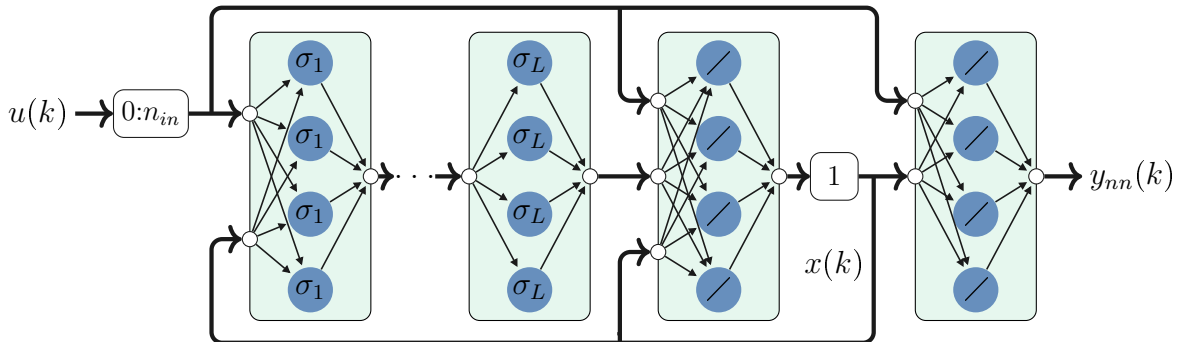


FIGURE 1.8 – Topologie du SSNN

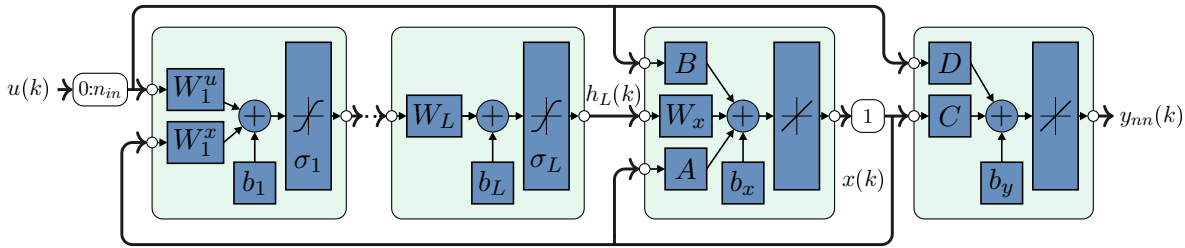


FIGURE 1.9 – Topologie du SSNN sous forme matricielle

Les réseaux SSNN présentent un nombre important d'avantages en comparaison à d'autres architectures récurrentes [Schenker et Agarwal, 1997, Zamarreño et Vega, 1998, Patan, 2008, Nelles, 2020] :

- Le modèle permet de représenter tous les systèmes dynamiques pouvant être décrits par l'équation d'état (1.23), dont les fonctions f_{SSNN} et g_{SSNN} existent, sont bornées et contiennent un nombre fini de discontinuités [Schenker et Agarwal, 1997, Hornik et al., 1989] (selon la démarche de nos travaux, g_{SSNN} doit être de plus linéaire). Cette classe couvre de nombreux systèmes pour lesquels une relation entrée-sortie d'ordre fini n'existe pas, en plus de tous les systèmes pour lesquels elle existe (ceux qui peuvent être approchés par un NNOE (1.22)).
- L'ordre dynamique qui correspond au nombre d'états du système peut être déterminé indépendamment du nombre de neurones cachés considéré. Bien que la complexité du modèle croisse à mesure que ces deux critères augmentent, leurs impacts peuvent être ajustés de façon séparée. D'une part, la dimension de l'état est liée à l'ensemble des informations stockées par le système à un instant donné. L'état dépend de la taille de la couche d'état mais également de la fenêtre d'entrée considérée (si $n_{in} > 1$). D'autre part, le nombre de neurones doit être sélectionné afin de conférer de bonnes capacités de généralisation au modèle. Un réseau comportant un nombre insuffisant de neurones peut ne pas être en mesure de refléter correctement la dynamique d'un système tandis qu'un réseau surdimensionné pourrait souffrir de mauvaises performances de généralisation résultant d'un sur-apprentissage.
- La structure bénéficie d'une contribution linéaire et non-linéaire ce qui peut améliorer les performances de modélisation puisque les systèmes dynamiques respectent souvent ce comportement hybride.
- La modélisation s'accorde avec les formalismes rencontrés en automatique. De plus, comme nous le présenterons dans le chapitre 3, un lien peut être établi avec les systèmes Linéaires à Paramètres Variants (LPV).

Devant les principaux avantages listés ci-dessus, l'architecture SSNN semble très prometteuse pour l'identification de systèmes dynamiques. Néanmoins, de nombreuses difficultés peuvent survenir dans la pratique si les mesures de l'état du système ou ses conditions initiales ne sont pas disponibles :

- L'apprentissage n'échappe pas aux difficultés rencontrés par les algorithmes d'optimisation relatifs aux réseaux récurrents.
- Bien que l'état du système soit facilement accessible depuis l'extérieur, il ne correspond pas à l'état du processus réel et n'a pas de signification physique.

- Une mauvaise initialisation peut détériorer de façon significative les performances du réseau.

Les inconvénients de l'architecture général des SSNN sont caractéristiques de l'identification par une approche RNN c'est-à-dire avec un réseau comprenant une dynamique interne. Afin de surmonter ces difficultés deux architectures plus spécifiques sont proposés dans ces travaux et présentées dans la section suivante.

1.4.5 Topologie des SSNNARX et SSNNOE

À l'image des NNARX présentés dans la section 1.4.2 et des NNOE de la section 1.4.3, deux architectures incluses dans le formalisme général des SSNN sont introduites dans cette section. Ces structures s'inspirent des concepts présentés dans les travaux de [Nørgård *et al.*, 2000] qui ont été largement repris comme par exemple dans [Abbas et Werner, 2008b].

Topologie des SSNNOE. Les SSNNOE (*State-Space Neural Network Output Error*) représentent un sous-ensemble des architectures SSNN. Ce type de réseaux se caractérise principalement par la contrainte imposée à l'équation de sortie afin que l'état soit défini selon :

$$x(k) = [y^T(k) \ \cdots \ y^T(k-n_{out})]^T \quad (1.30)$$

L'architecture SSNNOE est réduite d'une couche en comparaison aux SSNN (1.29) et la couche d'état correspond alors à la couche de sortie du réseau comme le montre la figure 1.10.

Définition 1.5 : SSNNOE

Un réseau de type SSNNOE à L couches cachées se définit par la dynamique :

$$\begin{cases} h_0(k) = [x^T(k) \ \bar{u}^T(k)]^T \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 1, \dots, L \\ x(k+1) = Ax(k) + B\bar{u}(k) + W_x h_L(k) + b_x \\ y_{nn}(k) = Cx(k) \end{cases} \quad (1.31)$$

où il est imposé que $x(k) = [y_{nn}^T(k) \ \cdots \ y_{nn}^T(k-n_{out})]^T$ et $\bar{u}(k) = [u^T(k) \ \cdots \ u^T(k-n_{in})]^T$.

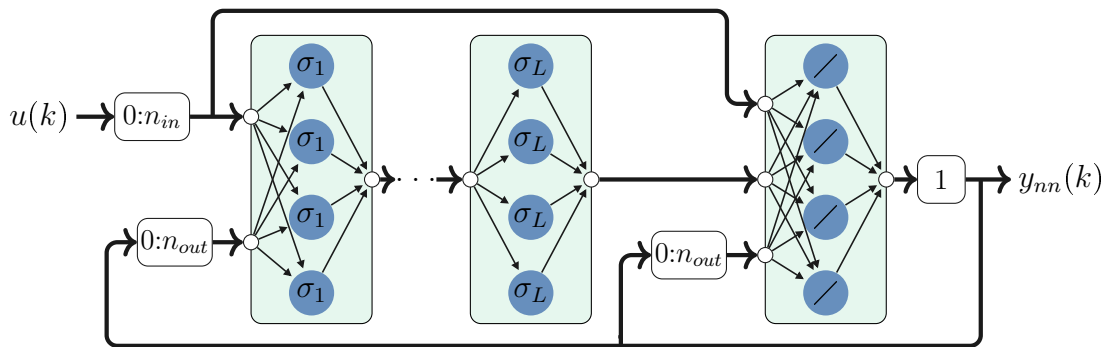


FIGURE 1.10 – Topologie du SSNNOE

Les réseaux SSNNOE reprennent les principaux avantages et inconvénients des SSNN généraux listés dans la section 1.4.4 précédente. Néanmoins ils se différencient sur les caractéristiques suivantes :

- Le modèle permet de modéliser à l'aide d'une représentation d'état les systèmes dynamiques restreints à ceux pour lesquels il existe une relation entrée-sortie d'ordre fini :

$$\begin{cases} x^{1:n_y}(k+1) = f_{\text{SSNNOE}}(x(k), u(k)) \\ y_{nn}(k) = Cx(k) \end{cases} \quad (1.32)$$

où la notation $x^{1:n_y} \in \mathbb{R}^{n_y}$ désigne le vecteur composé des n_y premières lignes de l'état $x \in \mathbb{R}^{n_x}$ de dimension $n_x = n_y(n_{out} + 1)$, ce qui correspond au vecteur de sortie estimé futur par le réseau c'est-à-dire $x^{1:n_y}(k+1) = y_{nn}(k+1)$. Le reste du vecteur d'état est donné suivant $x(k+1) = [y_{nn}^T(k+1) \cdots y_{nn}^T(k-n_{out}+1)]^T$.

- La définition de l'état (1.30) impose que certaines matrices soient par définition creuses. Pour une matrice X de dimensions appropriées et selon la notation $X^{1:n_y}$ désignant la matrice constituée des n_y premières lignes de X , la structure des matrices A , B , W_x , b_x et C de l'équation (1.31), peut être détaillée selon :

$$A = \begin{bmatrix} A^{1:n_y} & \\ I_{n_y n_{out}} & 0_{n_y n_{out} \times n_y} \end{bmatrix}, \quad B = \begin{bmatrix} B^{1:n_y} \\ 0 \end{bmatrix}, \quad W_x = \begin{bmatrix} W_x^{1:n_y} \\ 0 \end{bmatrix}, \quad b_x = \begin{bmatrix} b_x^{1:n_y} \\ 0 \end{bmatrix} \quad (1.33)$$

$$C = [I_{n_y} \quad 0_{n_y \times n_y n_{out}}]$$

- L'état possède une signification physique puisqu'il est constitué des sorties retardées (1.30), il peut de ce fait être plus facilement initialisé.

Remarque 1.2. Lors de l'implémentation du SSNNOE, il est parfois plus simple de faire remonter le retard imposé à la sortie dans les autres TDL du réseau. Ainsi la TDL unitaire du vecteur de sortie $y_{nn}(k)$ peut être retirée en considérant les TDL(1 : $n_{out}+1$) pour les sorties rebouclées et la TDL(1 : $n_{in}+1$) appliquée au vecteur d'entrée $u(k)$.

Remarque 1.3. Il est possible d'écrire un NNOE sous forme d'un SSNNOE si au minimum un retard est imposé à l'entrée c'est-à-dire que la TDL(1 : n_{in}) d'entrée est considérée. Néanmoins le modèle NNOE impose dans l'équation (1.33) que les matrices $A^{1:n_y} = 0$ et $B^{1:n_y} = 0$.

Topologie des SSNNARX. Les SSNNARX (*State-Space Neural Network AutoRegressive with eXternal input*) représentent l'architecture de type FNN associée au SSNNOE. L'état du système est externalisé, il correspond aux sorties passées mesurées ce qui permet de le considérer comme une entrée exogène pour l'apprentissage.

Définition 1.6 : SSNNARX

Un réseau de type SSNNARX à L couches cachées se définit par la dynamique :

$$\begin{cases} h_0(k) = [x^T(k) \bar{u}^T(k)]^T \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 1, \dots, L \\ x(k+1) = Ax(k) + B\bar{u}(k) + W_x h_L(k) + b_x \\ y_{nn}(k) = Cx(k) \end{cases} \quad (1.34)$$

où il est imposé que $x(k) = [y^T(k) \cdots y^T(k-n_{out})]^T$ et $\bar{u}(k) = [u^T(k) \cdots u^T(k-n_{in})]^T$.

Les réseaux SSNNARX couplent les avantages d'apprentissages des modèles FNN avec les autres points caractéristiques de l'architecture SSNNOE. Cependant comme les NNARX, la modélisation fidèle de systèmes dynamiques et la prédiction sur long terme ne sont pas assurées.

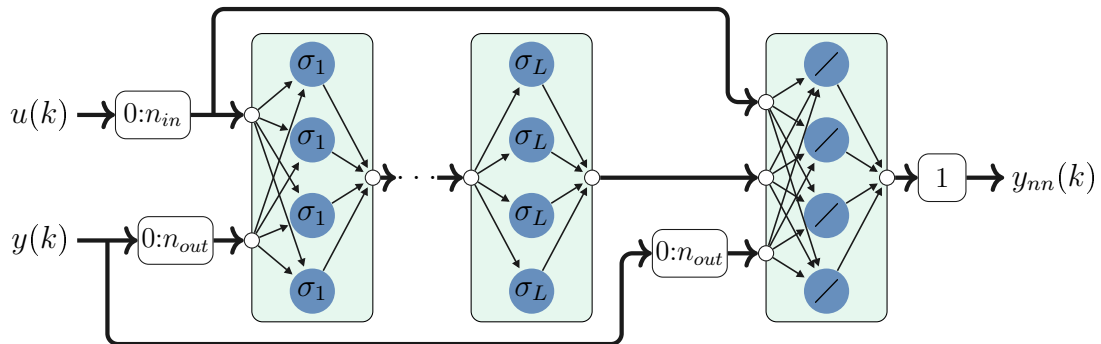


FIGURE 1.11 – Topologie du SSNNARX

SECTION 1.5

Procédure d'apprentissage

1.5.1 Expérimentation et acquisition des données

Génération des signaux d'excitation. Le choix des signaux d'excitation pour l'identification a un impact important sur la qualité du modèle identifié. L'objectif est de générer efficacement des données suffisamment riches en information pour la régression tout en tenant compte des contraintes du système et d'expérimentation.

Une littérature assez vaste couvre la problématique de plan d'expérience (en anglais *Design Of Experiments* ou DOE) et de son optimalité du moins dans le domaine linéaire. La problématique abordée est de concevoir l'expérience d'identification la moins coûteuse en moyen (nombre d'essais par exemple) tout en maximisant l'information contenue dans les données pour garantir un modèle suffisamment précis. Le choix des signaux d'excitation est donc un problème clé dans l'identification des systèmes dynamiques [Goodwin, 1977, Ljung, 1998, Godfrey et al., 2005]. Les signaux typiques de l'identification linéaire sont par exemple une somme de sinusoides de différentes amplitudes et fréquences ou la populaire PRBS (*Pseudo Random Binary Sequence*). Par ailleurs, lorsque des systèmes non-linéaires sont impliqués, l'utilisation des signaux binaires est insuffisante puisqu'il est important de représenter l'ensemble des fréquences et des amplitudes.

Pour l'identification de processus non-linéaires, il est important de déterminer le domaine d'opération du système afin d'agir en conséquence pour que l'ensemble de ce domaine soit représenté dans les données. De plus, dans la plupart des cas, le signal d'entrée doit mettre en évidence les propriétés de basse fréquence du système et donc avoir un contenu riche en basses fréquences [Söderström et Stoica, 1989]. Plusieurs signaux d'entrée ont été développés dans le but de respecter ces propriétés, les plus courants sont présentés dans [Nelles, 2020, Nørgård et al., 2000, Morelli et Klein, 2016].

Les signaux d'excitation utilisés dans nos travaux sont composés d'une succession d'échelons d'amplitudes et de durées aléatoires. Les amplitudes suivent une loi uniforme dont les grandeurs sont délimitées par des valeurs minimales et maximales, qui sont forcées d'apparaître dans le signal d'excitation au moins une fois pour s'assurer que le système soit excité sur la plage de valeurs désirée. De même, les durées totales des paliers sont déterminées de manière à ce que le

régime permanent soit atteint dans une proportion plus ou moins importante des cas. La durée des échelons est également un nombre aléatoire uniformément distribué qui est paramétré pour être compris entre zéro et une valeur choisie en fonction de la constante de temps du système. Un exemple de signal est présenté par la figure 1.12, pour une plage d'amplitude de $[-5 ; 10]$ et une durée n'excédant pas 2s.

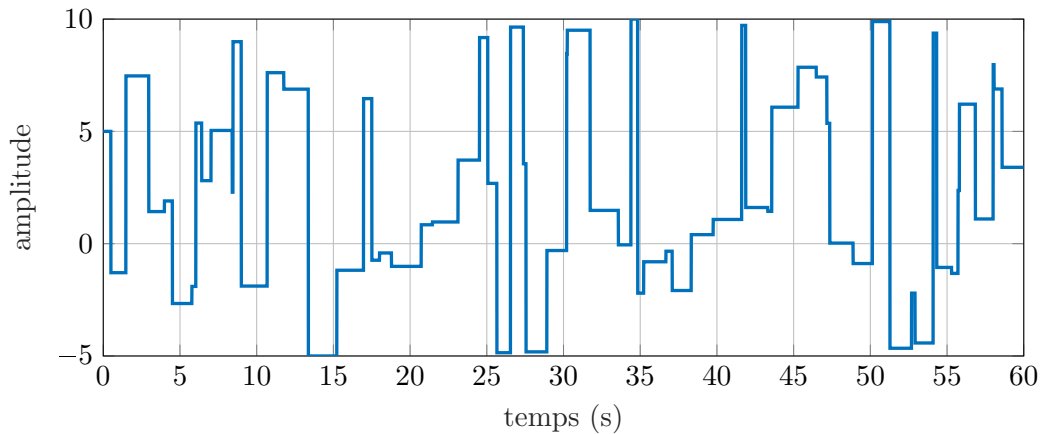


FIGURE 1.12 – Exemple de signal d'excitation

La quantité de données utilisées pour l'identification est également un point décisif au même titre que la qualité [Ljung, 1998]. La base de données doit être suffisamment informative et importante pour que le modèle appris reproduise fidèlement le comportement du système. La grandeur de la base de données doit alors être de manière utopique très importante pour aboutir à un apprentissage parfait. Néanmoins dans la pratique le temps de convergence des algorithmes d'apprentissage et les difficultés qu'ils peuvent rencontrer pour un trop grand nombre de données, pousse à utiliser des données de tailles raisonnables et dont l'information est suffisamment concentrée.

Fréquence d'échantillonnage. La détermination de la fréquence d'échantillonnage du modèle dynamique est rarement un choix facile alors que cette décision a pourtant un impact significatif pour l'apprentissage. Une période d'échantillonnage trop petite par rapport au système considéré encouragera un mauvais conditionnement des matrices ce qui se traduira par d'importants problèmes numériques lors de l'identification du modèle. À l'inverse, une période d'échantillonnage trop grande dissimulerait d'éventuelles dynamiques haute fréquence.

Néanmoins, l'identification est régulièrement utilisée à des fins de contrôle, ainsi la fréquence d'échantillonnage est impliquée dans les performances recherchées. Que l'identification concerne le système ou le contrôleur, le choix du pas d'échantillonnage doit donc être en accord avec la bande-passante désirée de la boucle fermée qui comprend ces deux éléments. Un cadencement suffisamment élevé permet d'imposer une certaine rapidité de réponse et un signal de contrôle plus lisse, mais les problèmes numériques de mauvais conditionnement peuvent être prononcés. La fréquence d'échantillonnage sélectionnée doit donc être un compromis entre privilégier l'identification et privilégier les performances du contrôle.

Finalement, le choix de la fréquence d'échantillonnage doit s'effectuer sur les connaissances *a priori* à disposition et les propriétés recherchées. Malgré un comportement non-linéaire du système à identifier, une approche linéaire permet souvent d'intuiter une période d'échantillonnage fonctionnelle. Selon les critères temporels et fréquentiels du linéaire, un bon point de départ est de considérer une dizaine de points au minimum pour caractériser une dynamique.

1.5.2 Pré-traitement des données

Filtrage. Le filtrage est couramment utilisé pour retirer les bruits de mesures, les perturbations périodiques et autres dynamiques non désirées. Pour la plupart des systèmes industriels dont le bruit est de haute fréquence, il est souvent judicieux d'utiliser un filtre passe-bas en prêtant attention aux phénomènes de repliement de spectre. Les filtres se comportent comme des systèmes dynamiques, par conséquent le filtre représente une dynamique de système supplémentaire introduite entre le système physique et le signal utilisé pour identifier les modèles dynamiques. Le filtrage doit donc être étudié pour ne pas fausser les résultats de la modélisation [Morelli et Klein, 2016].

Normalisation. L'étape de normalisation est essentielle dans la phase de pré-traitement des données. L'apprentissage sur une base non-normalisée peut aboutir à un comportement indésirable même si l'on dispose d'un ensemble de données comportant de nombreuses caractéristiques intéressantes. Le principal problème intervient lorsque les caractéristiques, c'est-à-dire les valeurs des données, sont à des échelles radicalement différentes. Le risque est qu'une caractéristique domine complètement les autres et qu'elle soit seule prise en compte lors de l'apprentissage. Un autre phénomène pouvant également poser problème est lié à l'emploi de fonctions d'activation qui sont souvent non-linéaires et généralement bornées. Des données concentrées sur une zone de non-linéarité ou dont les valeurs sont trop grandes par rapport aux bornes de la fonction d'activation pourraient être problématiques. La normalisation a finalement pour bénéfice de rendre le processus d'apprentissage plus robuste numériquement et d'accélérer sa convergence [LeCun et al., 1991]. De plus, l'expérience montre simplement que les modèles obtenus présentent de meilleures performances et leurs comportements se rapprochent davantage de ceux désirés. Il existe différentes techniques de normalisation dont les principales sont listées ci-dessous :

- **Normalisation *min-max*** : Cette méthode consiste à dimensionner l'amplitude des données entre $[-1 ; 1]$, la formule de normalisation est :

$$Z_{\min\text{-max}} = \frac{2(Z_{dt} - \min(Z_{dt}))}{\max(Z_{dt}) - \min(Z_{dt})} - 1 \quad (1.35)$$

La base de données possède donc les propriétés $\min(Z_{\min\text{-max}}) = -1$ et $\max(Z_{\min\text{-max}}) = 1$. Nous pouvons également évoquer la généralisation de cette méthode à la normalisation des données entre $[z_{\min} ; z_{\max}]$ par :

$$Z_{\min\text{-max}} = \frac{(z_{\max} - z_{\min})(Z_{dt} - \min(Z_{dt}))}{\max(Z_{dt}) - \min(Z_{dt})} + z_{\min} \quad (1.36)$$

- **Normalisation *z-score*** : Cette méthode impose aux données une moyenne nulle et un écart-type unitaire, le processus de normalisation est :

$$Z_{z\text{-score}} = \frac{Z_{dt} - \text{mean}(Z_{dt})}{\text{std}(Z_{dt} - \text{mean}(Z_{dt}))} \quad (1.37)$$

où mean est la fonction moyenne et std la fonction écart-type. La base de données vérifie alors les propriétés $\text{mean}(Z_{z\text{-score}}) = 0$ et $\text{std}(Z_{z\text{-score}}) = 1$.

- **Normalisation *max*** : Cette méthode consiste à imposer à la base de données une amplitude absolue inférieure à 1 en divisant par la valeur la plus élevée en absolu :

$$Z_{\text{max}} = \frac{Z_{dt}}{\max(|Z_{dt}|)} \quad (1.38)$$

Les données respectent donc $\max(|Z_{\text{max}}|) = 1$.

- **Normalisation *med-mad*** : Cette technique de normalisation se rapproche de la normalisation *z-score* mais utilise la médiane et la déviation absolue médiane (mad) pour transformer les données comme suggéré dans [Leys *et al.*, 2013]. L'expression mathématique de la normalisation est :

$$Z_{\text{med-mad}} = \frac{Z_{dt} - \text{med}(Z_{dt})}{\text{mad}(Z_{dt})} \quad (1.39)$$

où med est la fonction médiane et mad est la fonction déviation absolue médiane définie par $\text{mad}(Z_{dt}) = \text{med}(|Z_{dt} - \text{med}(Z_{dt})|)$. À l'issue de la normalisation les données vérifient $\text{med}(Z_{\text{med-mad}}) = 0$ et $\text{mad}(Z_{\text{med-mad}}) = 1$.

Les normalisations les plus communes sont *min-max* et *z-score*. À partir de ces deux techniques, il est possible de dresser deux grands types de normalisation. Les normalisations telles que *min-max* ou *max* impactent l'amplitude des données, elles garantissent donc que toutes les caractéristiques aient la même échelle. Néanmoins la distribution des données n'est pas modifiée ce qui peut aboutir à une mauvaise gestion des valeurs aberrantes. Les normalisations telles que *z-score* ou *med-mad* sont moins sensibles à la présence de valeurs aberrantes puisqu'elles agissent sur la distribution. Cependant les données ne partagent pas exactement une échelle commune.

L'ensemble des techniques de normalisation présentées sont en fait une transformation affine des données. La base de données normalisée Z_{norm} peut s'écrire selon l'équation :

$$Z_{\text{norm}} = A_{\text{norm}}Z^{dt} + B_{\text{norm}} \quad (1.40)$$

où les coefficients sont résumés par le tableau 1.2 pour chaque méthode. Notons qu'une fois le réseau entraîné avec les données normalisées, il est possible de modifier les poids d'entrée et de sortie afin d'utiliser le modèle sur les données réelles. La procédure de dénormalisation du réseau est présentée dans la section 1.5.4.

Séparation. Lors de tout apprentissage, il est primordial de conserver une partie des données hors du processus d'apprentissage afin de tester les performances du modèle sur un jeu de données indépendant. En plus de cet ensemble de test, un ensemble supplémentaire de validation est nécessaire afin d'éviter les phénomènes de sur-apprentissage comme présentés dans la section 1.3.3 abordant la généralisation. Les données sont donc séparées en trois ensembles :

- Ensemble d'apprentissage : Ensemble de données utilisé par l'algorithme d'apprentissage pour ajuster les poids du réseau.

TABLE 1.2 – Coefficient de normalisation pour les différentes normalisations affines

Normalisation	A_{norm}	B_{norm}
<i>min-max</i>	$A_{\min\text{-max}} = \frac{2}{\max(Z_{dt}) - \min(Z_{dt})}$	$B_{\min\text{-max}} = \frac{-\min(Z_{dt})}{\max(Z_{dt}) - \min(Z_{dt})} - 1$
<i>z-score</i>	$A_{z\text{-score}} = \frac{1}{\text{std}(Z_{dt} - \text{mean}(Z_{dt}))}$	$B_{z\text{-score}} = \frac{-\text{mean}(Z_{dt})}{\text{std}(Z_{dt} - \text{mean}(Z_{dt}))}$
<i>max</i>	$A_{\max} = \frac{1}{\max(Z_{dt})}$	$B_{\max} = 0$
<i>med-mad</i>	$A_{\text{med-mad}} = \frac{1}{\text{mad}(Z_{dt})}$	$B_{\text{med-mad}} = \frac{-\text{med}(Z_{dt})}{\text{mad}(Z_{dt})}$

- Ensemble de validation : Ensemble de données permettant d'éviter le sur-apprentissage et d'ajuster les hyperparamètres du réseau.
- Ensemble de test : Ensemble de données indépendantes permettant d'évaluer les performances et la généralisation du réseau.

La modélisation de systèmes dynamiques impose une division des données par blocs. Contrairement à d'autres applications comme la classification, il n'est pas possible de construire les trois ensembles de façon aléatoire à cause du caractère temporel des données. Communément l'ensemble d'apprentissage représente plus de la moitié des données disponibles, il est collecté en premier et suivi par celui de validation puis de test.

1.5.3 Sélection de la méthodologie d'apprentissage

Choix de la structure. La détermination de la structure du modèle passe par deux choix essentiels :

- L'architecture du réseau et les vecteurs de régression pris en compte via les TDL.
- Les hyperparamètres du réseau tels que le nombre de couches et de neurones ou les fonctions d'activation.

Les architectures envisagées dans ces travaux seront uniquement celles pouvant se décrire à l'aide d'une représentation d'état c'est-à-dire les SSNN. En effet l'utilisation ultérieure des modèles utilisera abondamment les liens de ces structures avec le domaine de l'automatique (voir chapitre 3). Lorsque les mesures du système à identifier (et l'approximation de leurs dérivées via les TDL) sont suffisantes pour constituer l'état du système, alors les architectures SSNNOE seront privilégiées. De plus, un apprentissage en deux temps à l'aide d'un SSNARX peut être réalisé afin d'améliorer les performances du réseau comme présenté ci-dessus. Les fenêtres de signaux retardés via les TDL doivent être choisies selon l'ordre dynamique du système souhaité. Dans le cas d'absence de bruit, il est possible d'utiliser le coefficient de Lipschitz [Nørgård *et al.*, 2000] pour guider le choix de ces paramètres.

Les hyperparamètres du réseau peuvent être déterminés dans un second temps afin de lui accorder de bonnes performances et de bonnes capacités de généralisation. Les fonctions d'activation non-linéaires employées dans ces travaux sont principalement la fonction tanh qui présente

une non-linéaire douce et une condition de secteur avantageuse (voir chapitre 3). Concernant les neurones cachés, sa quantité permet d'apporter de la complexité au modèle. La détermination du nombre de couches et de neurones cachés s'effectue en augmentant graduellement leurs nombres tout en évaluant l'erreur de test. La grandeur retenue est celle pour laquelle accroître davantage le nombre de neurones n'apporte qu'un gain de généralisation insignifiant.

Apprentissage en deux temps. Bien que les structures de réseau de type RNN soient préférées pour la modélisation de systèmes dynamiques, l'apprentissage de réseaux récurrents peut présenter des difficultés à converger. La fonction de coût considérée pour l'optimisation est fortement non-linéaire et elle peut comporter un grand nombre de minimums locaux peu performants. Les poids initiaux utilisés lors du problème d'optimisation fondé sur le gradient ont donc un rôle très influant sur la bonne convergence de l'algorithme.

La méthode proposée dans ces travaux est d'effectuer l'apprentissage d'un RNN en deux temps si la structure l'autorise. Les réseaux compatibles sont les RNN possédant une architecture FNN équivalente où la transformation des connexions récurrentes en des connexions reliées à de nouvelles entrées exogènes permet de passer d'une structure à l'autre. Les réseaux compatibles sont par exemple les associations NNOE \leftrightarrow NNARX et SSNNOE \leftrightarrow SSNNARX. Pour ces cas, la méthode d'apprentissage peut être divisée en deux étapes :

- Réaliser un premier apprentissage avec une architecture FNN équivalente à celle récurrente d'origine.
- Rétablir les connexions de récurrence puis exécuter l'apprentissage du réseau RNN dont l'initialisation des poids est issue des résultats de l'apprentissage précédent.

Cette procédure en deux temps a permis dans la pratique de résoudre de nombreux problèmes liés à l'apprentissage de certains systèmes dynamiques. Cette méthode peut présenter selon les cas, les avantages d'améliorer significativement la convergence des algorithmes d'optimisation et de réduire le temps nécessaire en comparaison à l'entraînement classique de RNN.

Choix de l'indice de performance. L'indice de performance retenu dans nos travaux sera le critère MSE (1.15) comme précédemment évoqué dans la section 1.3.1. Son utilisation est compatible avec l'ensemble des algorithmes d'optimisation fondés sur le gradient et plus particulièrement l'algorithme LM qui utilise le jacobien du critère. Si les signaux supervisés sont significativement bruités et que la matrice de covariance est connue, alors il peut être judicieux de transformer les sorties afin de considérer le critère MSE modifié de l'équation (1.16).

De plus, une méthode complémentaire peut être utilisée pour améliorer les capacités de généralisation de certains réseaux, notamment des structures FNN postérieurement rebouclées. Cette proposition qui dérive des travaux de [Nørgård *et al.*, 2000] consiste à superviser durant l'apprentissage une fenêtre glissante de sorties et non uniquement le vecteur de sortie à un moment donné. L'indice de performance minimisé pendant l'entraînement est alors le MSE défini sur une fenêtre de signaux retardés de taille n_{out} par :

$$J_{mse, n_{out}} = \frac{1}{2N_{dt}n_y n_{out}} \sum_{k=1}^{N_{dt}} [\bar{y}(k) - \bar{y}_{nn}(k)]^T [\bar{y}(k) - \bar{y}_{nn}(k)] \quad (1.41)$$

où $\bar{y}(k) = [y^T(k) \cdots y^T(k-n_{out}+1)]^T$ et $\bar{y}_{nn}(k) = [y_{nn}^T(k) \cdots y_{nn}^T(k-n_{out}+1)]^T$ sont respectivement les vecteurs de sorties mesurées et estimées. Les réseaux résultant d'un apprentissage en

FNN à l'aide de ce critère, ont exhibé dans la pratique, de meilleures propriétés de généralisation lorsqu'ils sont employés de façon récurrente pour la modélisation de systèmes dynamiques.

Choix de l'algorithme d'apprentissage. Pour les réseaux comportant jusqu'à quelques centaines de poids exploitables lors de l'entraînement alors l'algorithme LM est souvent le plus rapide et le plus performant. Cependant l'algorithme implique une étape d'inversion de matrices dont les dimensions sont directement reliées au nombre de paramètres du modèle. Par cette complexité de calcul, le LM peut grandement perdre en efficacité sur des grandes structures. En conséquence, lorsque le nombre de poids est de l'ordre du millier ou plus, les algorithmes tels que BFGS ou SCG seront privilégiés [Hagan *et al.*, 2014, Pouilly-Cathelain, 2020].

Critères d'arrêt de l'algorithme d'apprentissage. Les critères d'arrêt sont définis avec des valeurs scalaires qui indiquent des limites à partir desquelles continuer l'entraînement n'améliorera pas significativement les résultats. Une fois que l'une de ces conditions est satisfaite, l'apprentissage est interrompu automatiquement. Les critères suivants sont retenus pour arrêter les algorithmes d'apprentissage lorsqu'ils sont atteints :

- Nombre d'itérations maximum
- Temps de calcul maximum
- Coût minimum
- Valeur du gradient minimum
- Nombre d'itérations successives maximum sans amélioration globale de l'erreur de validation (arrêt prématuré, voir la section 1.3.3)

Le choix des conditions d'arrêt doit donc être adapté pour permettre l'obtention rapide de réseaux performants en limitant le sur-apprentissage.

Apprentissage sur plusieurs tirages. Les algorithmes d'apprentissage de descente de gradient sont par définition des méthodes de recherche locale. Ces algorithmes convergent alors vers l'un des minimums locaux de la surface de performance, ce qui dépend sensiblement des conditions initiales des paramètres. Ainsi un seul cycle d'exécution de l'entraînement c'est-à-dire un seul tirage peut ne pas garantir une performance optimale. C'est pourquoi dans le contexte de l'apprentissage hors ligne, il est préférable de recommencer l'apprentissage avec différentes initialisations des poids du réseau. Cinq à dix tirages sont réalisés afin de se rapprocher de l'optimum global [Hagan *et al.*, 2014] et le réseau sélectionné est celui qui produit les meilleures performances de validation.

1.5.4 Analyse des performances et post-traitement du réseau

Critères de validation d'un réseau. Une fois l'apprentissage effectué, il convient d'estimer si le modèle final obtenu est suffisamment fidèle au système identifié. L'objectif est donc de déterminer si le réseau reproduit le comportement du système dynamique avec un niveau de précision acceptable. Mesurer ou plus précisément intuitiver cet écart de comportement peut être réalisé des façons suivantes :

- Visualisation et simulation : Évaluer explicitement le comportement du modèle peut simplement être réalisé en visualisant les réponses temporelles du réseau. Les tracés de simulation fournissent souvent une représentation intuitive des attitudes du modèle, ils peuvent être de plus comparés avec d'éventuelles données mesurées à disposition.

- Estimation de la capacité de généralisation : Reproduire le comportement du système réel pour des données quelconques, revient à la problématique de généralisation. Ainsi la généralisation du réseau peut être estimée par l'évaluation de la performance sur l'ensemble des données de test.

L'identification implique bien souvent que différents modèles, architectures ou techniques vont être employés et comparés. Sélectionner le réseau et la méthode qui sont les plus appropriés passe par l'examen des critères ci-dessus auxquelles peuvent s'ajouter par exemples la performance d'apprentissage et de validation, le nombre de paramètres, le temps de calcul, le nombre d'itérations ou encore la répétabilité de la méthode.

Dénormalisation du réseau. L'utilisation de données normalisées est indispensable lors du processus d'apprentissage (voir la section 1.5.1), cependant l'usage du réseau final porte la plupart du temps sur les données d'origines. Conformément à cet objectif, les poids finaux du réseau peuvent être modifiés pour que le modèle utilise les données réelles.

Pour la dénormalisation du réseau l'ensemble des couches reliées à l'entrée ou à la sortie doit être modifié. La méthode considère qu'une normalisation affine des données à été employée, elle peut donc être décrite par l'équation (1.40). Les entrées normalisées u_{norm} et les sorties normalisées y_{norm} peuvent être définies séparément par :

$$u_{norm} = A_{norm}^u u + B_{norm}^u, \quad y_{norm} = A_{norm}^y y + B_{norm}^y \quad (1.42)$$

L'algorithme de dénormalisation se compose des trois étapes successives présentées ci-dessous. Les notations sont que pour une couche ℓ quelconque du réseau, W_ℓ désigne la matrice de poids et b_ℓ le vecteur de biais. Lorsque la couche est connectée à l'entrée ou à la sortie, les poids correspondant à ces connexions sont isolés respectivement dans les sous-matrices W_ℓ^u et W_ℓ^y . Finalement pour un réseau à L couches cachées les modifications suivantes sont effectuées :

- Pour toutes les couches ℓ du réseau possédant une connexion avec l'entrée, la matrice de connexion avec l'entrée et le vecteur de biais sont modifiés selon :

$$\begin{aligned} W_\ell^u &\leftarrow W_\ell^u A_{norm}^u \\ b_\ell &\leftarrow b_\ell + W_\ell^u B_{norm}^u \end{aligned} \quad (1.43)$$

- Pour toutes les couches ℓ du réseau possédant une connexion avec la sortie, la matrice de connexion avec la sortie et le vecteur de biais sont modifiés selon :

$$\begin{aligned} W_\ell^y &\leftarrow W_\ell^y A_{norm}^y \\ b_\ell &\leftarrow b_\ell + W_\ell^y B_{norm}^y \end{aligned} \quad (1.44)$$

- Pour la couche $L + 1$ de sortie linéaire, l'ensemble de la matrice de poids et le vecteur de biais sont modifiés selon :

$$\begin{aligned} W_{L+1} &\leftarrow (A_{norm}^y)^{-1} W_{L+1} \\ b_{L+1} &\leftarrow (A_{norm}^y)^{-1} (b_{L+1} - B_{norm}^y) \end{aligned} \quad (1.45)$$

Conclusions

Nous avons développé dans ce chapitre une méthodologie complète pour effectuer l'identification de systèmes dynamiques par réseaux de neurones à structure fixée. Au vue de l'utilisation des modèles pour l'automatique, l'architecture des réseaux employés correspond à celle de MLP récurrents. Plus spécifiquement, cette partie a permis de développer et de dresser un bref état de l'art de la topologie SSNN qui est le modèle neuronal retenu pour la suite des travaux. Nous avons également pu présenter une procédure d'apprentissage et ainsi détailler ses principales étapes. Premièrement, l'acquisition des données et les techniques de génération des signaux d'excitation ont été présentées. Par la suite, le pré-traitement de données pour l'apprentissage a pu être abordé notamment pour la normalisation et la séparation (selon les ensembles d'apprentissage, de validation et de test) des données. Après quoi, la méthodologie d'apprentissage a été détaillée, les topologies privilégiées sont celles des SSNNNOE dont une méthode d'apprentissage innovante en deux temps via l'initialisation des poids par un pré-apprentissage en SSNNARX a été proposée. Le critère de performance retenu pour les travaux correspond à l'indice MSE tandis que l'algorithme d'apprentissage majoritairement employé est celui de Levenberg-Marquardt. Les phases d'entraînement sont réalisées hors lignes et pour plusieurs tirages, c'est-à-dire pour plusieurs conditions initiales. La notion de généralisation du réseau a également été introduite et la méthode d'arrêt prématuré de l'apprentissage est utilisée afin d'éviter le sur-apprentissage lorsque l'erreur de validation n'est plus minimum depuis un certain nombre d'itérations donné. Finalement, la validation et le post-traitement du réseau ont été présentés dont spécifiquement la phase de dénormalisation des poids du réseau final qui a été formalisée.

Outils d'analyse des systèmes LPV

Sommaire

2.1	Introduction	44
2.2	Notions de stabilité et outils dans le cadre linéaire invariant	44
2.2.1	Réponse fréquentielle	45
2.2.2	Norme \mathcal{H}_∞	45
2.2.3	Estimation de la norme \mathcal{H}_∞ par le lemme borné réel	46
2.2.4	Transformation en w	47
2.3	Introduction aux systèmes LPV	49
2.3.1	Définition des systèmes LPV	49
2.3.2	Transformation linéaire fractionnaire (LFT)	50
2.4	Analyse de stabilité et de performance des systèmes LPV-LFT	53
2.4.1	Notions de stabilité des systèmes LPV	53
2.4.2	Théorème du petit gain	54
2.4.3	Théorème du petit gain avec multiplicateurs	55
2.4.4	Analyse de performance des systèmes LPV-LFT	57
2.5	Conclusions	60

SECTION 2.1

Introduction

Nous nous intéressons dans ce chapitre aux outils d'analyses de la commande robuste qui seront exploités dans la suite des travaux et pour lesquels nous présentons succinctement les principaux résultats. Les méthodes abordées portent sur l'étude des systèmes non-linéaires qui peuvent être modélisés par une représentation LPV. Comme nous l'aborderons plus tard dans le chapitre 3, les modèles SSNN peuvent être représentés selon ce type de formulation où les neurones non-linéaires sont appréhendés à l'aide de paramètres variants associés. De par les grandes dimensions des systèmes impliqués, nous limitons l'étude à ceux représentés par LPV-LFT. De plus, l'approche favorise les systèmes à temps discret au regard de la nature discrète des réseaux de neurones qui traitent des données également échantillonnées. Ainsi, bien que le domaine continu soit souvent privilégié, nous rappelons ici le volet discret des outils d'analyses de systèmes LPV-LFT. Les concepts et problématiques abordés restent néanmoins similaires pour le temps continu, le lecteur intéressé et souhaitant davantage de détails est invité à se référer aux références citées au fur et à mesure de ce chapitre.

Dans la section 2.2, nous introduisons les principales notions et outils relatifs aux systèmes LTI. Nous présentons par la suite dans la section 2.3, les systèmes LPV et une façon de les représenter selon le formalisme LFT. Les outils d'analyse de stabilité et de performance des systèmes LPV-LFT sont finalement formulés dans la section 2.4.

SECTION 2.2

Notions de stabilité et outils dans le cadre linéaire invariant

Nous présentons succinctement pour débiter les outils mathématiques indispensables permettant d'apprécier la robustesse des systèmes dynamiques. Soit un système linéaire invariant dans le temps (LTI) d'ordre n_x à n_u entrées et n_y sorties, dont le comportement dynamique est défini par un modèle dans l'espace d'état discret de la forme :

$$G := \begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases} \quad (2.1)$$

où $x(k) \in \mathbb{R}^{n_x}$ est l'état du système au pas d'échantillonnage $k \in \mathbb{N}$, $u \in \mathbb{R}^{n_u}$ est l'entrée du système, $y \in \mathbb{R}^{n_y}$ est la sortie et A, B, C, D sont des matrices de dimensions appropriées dénommées respectivement matrice d'évolution, de commande, d'observation et de transfert direct.

La matrice de transfert associée à la réalisation (A, B, C, D) est donnée dans le domaine de la transformée en Z en supposant une condition initiale nulle par :

$$G(z) = C(zI - A)^{-1}B + D \quad (2.2)$$

Lorsque le système est monovarié ou SISO (*Single Input and Single Output*), une seule fonction de transfert est nécessaire pour décrire la relation entre l'entrée ($n_u = 1$) et la sortie ($n_y = 1$). Dans les autres cas des systèmes multivariables ou MIMO (*Multiple Input and Multiple Output*), la matrice de transfert décrit terme à terme la relation entre chaque combinaison d'entrée et de sortie. De plus, le système est strictement propre si il n'existe pas de lien direct entre l'entrée et la sortie, c'est-à-dire que la matrice de transfert direct (D) est nulle.

2.2.1 Réponse fréquentielle

Dans le cas des systèmes discrets, l'évaluation de $G(z)$ pour $z = e^{j\omega}$ avec $\omega \in [0, \pi]$ définit la réponse fréquentielle du système. Pour un système monovariante, la notion de gain du système à la fréquence ω est définie par le module $|G(e^{j\omega})|$. Pour le cas multivariante, ce concept est étendu à l'aide des valeurs singulières.

Remarque 2.1. En faisant intervenir la fréquence d'échantillonnage T_e , il est également possible d'évaluer la réponse fréquentielle $G(z)$ pour $z = e^{jT_e\omega}$ avec $\omega \in [0, \pi/T_e]$.

Définition 2.1 : Valeurs singulières

Soit la matrice de transfert $G(z)$ de dimension $n_y \times n_u$, les valeurs singulières du système sont définies comme les racines carrées des valeurs propres de sa réponse fréquentielle $G(e^{j\omega})$ multipliée par sa transconjugée, c'est-à-dire pour $i = 1, \dots, \min(n_u, n_y)$, chaque valeur singulière est définie selon :

$$\sigma_i(G(e^{j\omega})) := \sqrt{\lambda_i(G(e^{j\omega})G(e^{-j\omega})^T)} = \sqrt{\lambda_i(G(e^{-j\omega})^T G(e^{j\omega}))} \quad (2.3)$$

Au regard de cette définition, les valeurs singulières sont des nombres réels positifs ou nuls, qui peuvent donc être classés de la plus grande, appelée valeur singulière supérieure $\bar{\sigma}(G)$, à la plus petite, appelée valeur singulière inférieure $\underline{\sigma}(G)$:

$$\bar{\sigma}(G(e^{j\omega})) = \sigma_1(G(e^{j\omega})) \geq \sigma_2(G(e^{j\omega})) \geq \dots \geq \underline{\sigma}(G(e^{j\omega})) \geq 0 \quad (2.4)$$

Les valeurs singulières constituent une généralisation aux systèmes multivariante de la notion de gain établie dans le cas des systèmes monovariante. En effet pour un système multivariante le gain de chaque transfert, et plus généralement de chaque valeur singulière d'une sous-matrice de transfert, à une fréquence donnée, sera compris entre les valeurs singulières inférieure et supérieure.

Remarque 2.2. Pour un système monovariante, il existe une seule valeur singulière qui est telle que :

$$\bar{\sigma}(G(e^{j\omega})) = \underline{\sigma}(G(e^{j\omega})) = |G(e^{j\omega})| \quad (2.5)$$

2.2.2 Norme \mathcal{H}_∞

L'approche \mathcal{H}_∞ propose un cadre théorique général pour concevoir des contrôleurs en manipulant des concepts fréquentiels [Doyle *et al.*, 1990, Duc et Font, 1999]. Il s'agit essentiellement d'un problème d'optimisation dont le but est de satisfaire des contraintes en boucle fermée en fonction de critère de performances et de robustesse.

Définition 2.2 : Ensemble \mathcal{RH}_∞

\mathcal{RH}_∞ définit le sous-espace des matrices de transfert rationnelles propres à coefficients réels et stables c'est-à-dire dans le cas discret n'ayant pas de pôles à l'extérieur du cercle unité.

Définition 2.3 : Norme \mathcal{H}_∞

Soit le système $G \in \mathcal{RH}_\infty$, la norme \mathcal{H}_∞ de G est définie par :

$$\|G(z)\|_\infty := \sup_{\omega \in [0; \pi]} \bar{\sigma}(G(e^{j\omega})) \quad (2.6)$$

Cette norme représente donc la valeur la plus élevée du gain du système sur l'ensemble des pulsations, dans le cas d'un système monovarié, c'est la valeur la plus élevée de $|G(e^{j\omega})|$.

2.2.3 Estimation de la norme \mathcal{H}_∞ par le lemme borné réel

A défaut de pouvoir calculer analytiquement la norme \mathcal{H}_∞ , la norme est caractérisée mathématiquement notamment en estimant une borne supérieure à l'aide de méthodes numériques variées. Nous pouvons notamment citer les méthodes de dichotomie [Boyd *et al.*, 1989], les algorithmes fondés sur la matrice hamiltonienne [Zhou *et al.*, 1996] ou les algorithmes basés sur les équations de Riccati et les inégalités matricielles affines. Ces dernières sont également désignées par leur sigle LMI provenant de l'anglais *Linear Matrix Inequalities*. Nous nous intéresserons particulièrement à une approche de cette catégorie qui est couramment dénommée lemme borné réel. Ce résultat fondamental constitue le point de départ de la résolution de nombreux problèmes en automatique par l'optimisation sous contrainte LMI [Boyd *et al.*, 1994].

Lemme 2.1 : Lemme borné réel discret [Gahinet et Apkarian, 1994]

Soit le système $G(z)$ ayant pour réalisation (A, B, C, D) et γ un réel positif, alors le système est stable et vérifie

$$\|G(z)\|_\infty < \gamma \quad (2.7)$$

si et seulement si il existe une matrice symétrique définie positive X solution de l'inégalité :

$$\begin{bmatrix} A^T X A - X & A^T X B & C^T \\ B^T X A & B^T X B - \gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} < 0 \quad (2.8)$$

Remarque 2.3. Comme exposé dans [Gahinet et Apkarian, 1994], une troisième affirmation également équivalente à celles présentées dans le lemme 2.1 est qu'il existe une matrice symétrique définie positive X solution de l'inégalité :

$$\begin{bmatrix} -X^{-1} & A & B & 0 \\ A^T & -X & 0 & C \\ B^T & 0 & -\gamma I & D^T \\ 0 & C & D & -\gamma I \end{bmatrix} < 0 \quad (2.9)$$

D'après ce lemme, la proposition suivante expose une méthode de calcul de la norme \mathcal{H}_∞ d'un système dynamique par le biais de la résolution d'un problème d'optimisation de type LMI.

Proposition 2.1

Soit le système $G(z)$ de réalisation (A, B, C, D) et dont la norme \mathcal{H}_∞ est $\|G(z)\|_\infty = \gamma^*$ alors on a également :

$$\gamma^* = \min_{\substack{\gamma > 0 \\ X > 0}} \left\{ \gamma \text{ sous (2.8)} \right\} \quad (2.10)$$

La proposition ci-dessus apporte une méthode de majoration de la norme \mathcal{H}_∞ d'un système. Ce problème de minimisation sous contraintes LMI d'une fonction qui dépend linéairement des inconnues est un problème d'optimisation convexe, qui a donc pour avantage de ne présenter aucun minimum local. La formulation ne fait également pas intervenir la fréquence, c'est-à-dire que le problème est donc de dimension finie. Les problèmes LMI surviennent dans de nombreux domaines de l'automatique comme celui du contrôle robuste [Scherer et Weiland, 2015, Boyd et al., 1994]. Ce type de problème peut être résolu numériquement avec des algorithmes d'optimisation convexe adaptés. La *Robust Control Toolbox* de MATLAB fournit des outils de résolution numériques pour traiter l'optimisation sous contrainte LMI [Gahinet et al., 1994]. D'autres outils performants et libres d'utilisation sont proposés sous MATLAB comme CVX [Grant et Boyd, 2008, Grant et Boyd, 2014] ou YALMIP [Lofberg, 2004].

Les problèmes d'optimisation sous contraintes LMI sont donc résolus à l'aide d'outils numériques comme évoqué précédemment. Cependant, suivant la complexité des problèmes abordés, des difficultés numériques peuvent émerger lors des résolutions. Une technique qui peut se révéler efficace est de résoudre le problème équivalent mais dans le domaine continu. Bien qu'ils ne soient pas présentés dans ce rapport, les problèmes LMI en temps discret peuvent être obtenus d'une manière similaire en temps continu. Les inégalités obtenues dans le cas continu procurent une formulation alternative du problème LMI initial, sa résolution peut se révéler dans la pratique moins sensible aux difficultés numériques. La section 2.2.4 qui suit présente la transformation en w qui est un outil intéressant pour commuter entre les domaines continus et discrets.

2.2.4 Transformation en w

Cette section est consacrée à l'étude de la transformation en w aussi appelé transformation bilinéaire. Cette méthode qui peut être utilisée de façon directe ou inverse, fournit un lien mathématique entre les systèmes à temps discret et ceux à temps continu.

Définition 2.4 : Transformation en w

La transformation en w est définie par la substitution :

$$z \rightarrow \frac{1+w}{1-w}, \quad w \neq 0 \quad (2.11)$$

Pour un système discret $G_d(z)$, son image en temps continu par la transformation bilinéaire est donnée par :

$$G_w(w) = G_d\left(\frac{1+w}{1-w}\right) \quad (2.12)$$

Il est important de mentionner que le système continu obtenu n'a pas d'interprétation physique. Néanmoins, les forts intérêts de la transformation bilinéaire se trouvent dans les propriétés suivantes.

Propriété 2.1

L'étude de la stabilité d'un système LTI en temps discret $G_d(z)$ est équivalente à l'étude de la stabilité de son image $G_w(w)$ par la transformation bilinéaire.

Démonstration. La démonstration est omise mais est disponible dans de nombreux travaux sur la commande \mathcal{H}_∞ tels que [Feyel, 2013]. \square

Propriété 2.2

La norme \mathcal{H}_∞ est conservée par la transformation bilinéaire, en effet :

$$\|G_d(z)\|_\infty \text{ (norme discrète)} = \|G_w(w)\|_\infty \text{ (norme continue)} \quad (2.13)$$

Démonstration.

$$\begin{aligned} \|G_d(z)\|_\infty \text{ (norme discrète)} &= \sup_{\omega \in \mathbb{R}} \sqrt{\bar{\lambda}(G_d(e^{j\omega})G_d(e^{-j\omega})^T)} \\ &= \sup_{\omega \in \mathbb{R}} \sqrt{\bar{\lambda}\left(G_d\left(\frac{1+w}{1-w}\right)G_d\left(\frac{1+w}{1-w}\right)^T\right)} \\ &= \sup_{\omega \in \mathbb{R}} \sqrt{\bar{\lambda}(G_w(j\omega)G_w(-j\omega)^T)} \\ &= \|G_w(w)\|_\infty \text{ (norme continue)} \end{aligned} \quad (2.14)$$

où $\bar{\lambda}$ désigne la plus grande des valeurs propres. \square

La transformation bilinéaire appliquée à un système discret permet ainsi d'obtenir l'image du système en temps continu. De plus, la transformée inverse permet de revenir au contexte discret initial. D'après les travaux [Chen et Francis, 1995], les relations des représentations d'état des systèmes discrets et continus sont formulées par les résultats suivants :

- Considérons le système discret $G_d(z) = (A_d, B_d, C_d, D_d)$ et supposons que la matrice $(A_d + I)$ est non-singulière, l'image de G_d par la transformation bilinéaire est le système continu $G_w(w) = (A_w, B_w, C_w, D_w)$ dont les matrices sont données par :

$$\begin{cases} A_w = (A_d - I)(A_d + I)^{-1} \\ B_w = (I - A_w)B_d \\ C_w = C_d(A_d + I)^{-1} \\ D_w = D_d - C_w B_d \end{cases} \quad (2.15)$$

- Considérons le système continu $G_w(w) = (A_w, B_w, C_w, D_w)$ et supposons que la matrice $(I - A_w)$ est non-singulière, l'image de G_w par l'inverse de la transformation bilinéaire est le système discret $G_d(z) = (A_d, B_d, C_d, D_d)$ dont les matrices sont :

$$\begin{cases} A_d = (I - A_w)^{-1}(I + A_w) \\ B_d = (I - A_w)^{-1}B_w \\ C_d = C_w(I + A_d)^{-1} \\ D_d = D_w + C_w B_d \end{cases} \quad (2.16)$$

Introduction aux systèmes LPV

Les systèmes Linéaires à Paramètres Variants (LPV) constituent une classe de modèles particulière de systèmes dynamiques. Le comportement du processus à l'étude est mathématiquement décrit comme un système linéaire paramétré dont les paramètres varient au cours du temps. Depuis leur introduction dans [Shamma, 1988, Shamma et Athans, 1990], ce type de représentation est devenu un formalisme standard de l'automatique que ce soit pour l'analyse, la synthèse, la modélisation ou l'identification. Les systèmes LPV offrent une généralisation très utile des systèmes Linéaires Temps-Invariant (LTI) tout en les distinguant des systèmes Linéaires Temps-Variant (LTV) [Shamma, 2012]. Cette classe de système à d'autant plus d'importance puisque de nombreux systèmes non-linéaires peuvent être reformulés dans ce cadre d'étude, les non-linéarités sont alors intégrés sous la forme de paramètres linéaires variants qui dépendent alors de signaux internes au système ou de ses entrées. Les systèmes sont alors qualifiés de systèmes quasi-LPV, pour faire la distinction par rapport aux systèmes LPV purs où les paramètres variants dépendent uniquement de signaux extérieurs.

2.3.1 Définition des systèmes LPV

Les systèmes Linéaires à Paramètres Variants (LPV) sont une classe de systèmes dynamiques dont le comportement peut être mathématiquement représenté à l'aide d'équations linéaires dépendant de paramètres qui changent de valeurs au cours du temps. Les systèmes LPV sont décrits par une représentation d'état de la forme :

$$\tilde{G}(\theta) := \begin{cases} x(k+1) = \tilde{A}(\theta(k))x(k) + \tilde{B}(\theta(k))u(k) \\ y(k) = \tilde{C}(\theta(k))x(k) + \tilde{D}(\theta(k))u(k) \end{cases} \quad (2.17)$$

où $x(k) \in \mathbb{R}^{n_x}$ est l'état du système au pas d'échantillonnage $k \in \mathbb{N}$, $u \in \mathbb{R}^{n_u}$ est l'entrée du système, $y \in \mathbb{R}^{n_y}$ est la sortie et A, B, C, D sont des matrices fixes de dimensions appropriées mais dépendantes du vecteur de paramètres $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$ qui varie dans le temps. Les paramètres sont généralement considérés comme bornés et évoluent dans l'espace des paramètres Θ considéré dans notre étude comme l'ensemble convexe suivant :

$$\Theta := \left\{ \theta = [\theta_1, \dots, \theta_{n_\theta}]^T, \quad \theta_i \in [\underline{\theta}_i, \bar{\theta}_i] \forall i = 1, \dots, n_\theta \right\} \quad (2.18)$$

Les paramètres affectent de façon interne le système en modifiant sa dynamique au cours du temps et ainsi son comportement entrée-sortie global. Si l'on fixe ou l'on « gèle » la valeur des paramètres à une valeur donnée, alors le système devient un système linéaire invariant dans le temps (LTI). Les paramètres sont généralement des variables exogènes au système, néanmoins dans le cas d'approximation de système non-linéaire par des systèmes quasi-LPV, les paramètres deviennent endogènes puisqu'ils dépendent finalement de l'état du système.

Remarque 2.4. Dans de nombreux travaux de la littérature, des bornes sur la vitesse de variations des paramètres sont aussi considérées ($\delta\theta = \theta(k+1) - \theta(k)$). Toutefois cette hypothèse n'est pas nécessaire dans le contexte de la stabilité quadratique [Apkarian et Gahinet, 1995, Briat, 2014] dans lequel s'inscrit ce travail où la vitesse de variation des paramètres n'est pas contrainte.

La représentation du système LPV (2.17) est difficilement exploitable directement sous cette forme pour l'analyse ou la synthèse de contrôleurs, c'est pourquoi, différentes modélisations ont été développées dans la littérature. Nous pouvons distinguer trois principales méthodes de représentations :

- l'approche polytopique : Cette méthode est utilisée lorsque les paramètres varient à l'intérieur d'un polytope et que la dépendance en ses paramètres est affine. Le système LPV est alors représenté par la combinaison convexe des systèmes LTI définis à chaque sommet du polytope [Apkarian *et al.*, 1995]. En effet, par la propriété de convexité, la stabilité des systèmes aux sommets du polytope garantit la stabilité de tout système à l'intérieur du polytope. Dans le cas le plus courant, les paramètres sont indépendants et évoluent dans un espace borné (2.18), cet espace est alors un hypercube défini par ses 2^{n_θ} sommets.
- l'approche par Transformations Linéaires Fractionnaires (LFT) : Cette représentation principalement employée lors de l'étude de la robustesse s'avère également adaptée aux systèmes LPV. La section 2.3.2 suivante aborde plus en détail cette méthode dont le principe est de séparer le système LPV en deux parties distinctes. La première partie comprend la dynamique du système sous la forme d'un système LTI tandis qu'une seconde partie réunit la dépendance du système vis-à-vis des paramètres variants.
- l'approche par maillage ou *gridding* : Cette méthode consiste à aborder l'ensemble des paramètres variants possibles sur un nombre fini de points de fonctionnement [Wu, 1995, Apkarian et Adams, 2000]. Ainsi l'espace des paramètres est discrétisé afin de représenter le système LPV par un ensemble de systèmes LTI. La synthèse et l'analyse sont réalisées sur chaque point du maillage qui peut être choisi plus ou moins dense.

L'approche polytopique ainsi que l'approche par maillage ne seront pas étudiées dans cette thèse, seule l'approche LFT sera considérée et décrite dans la section 2.3.2 qui suit. Cette restriction se justifie par la grande dimension du vecteur de paramètres des systèmes LPV qui seront considérés. En effet les méthodes d'analyses ou de synthèses des systèmes LPV sont principalement réalisées par des problèmes d'optimisation sous contrainte d'Inégalités Matricielles Linéaires (LMI). Chacune des trois approches possède ses propres algorithmes dont les complexités numériques ne peuvent être ignorées [Hoffmann et Werner, 2014a]. Les systèmes LPV abordés dans cette thèse possèdent potentiellement de nombreux paramètres variants, ainsi la représentation LFT se retrouve être la plus adaptée pour ce type de système [Hoffmann et Werner, 2014b].

2.3.2 Transformation linéaire fractionnaire (LFT)

Les transformations linéaires fractionnaires ou LFT (pour l'anglais *Linear Fractional Transformations*) représentent un formalisme mathématique standard dans les problèmes de modélisation et de commande des systèmes dynamiques. Dans un contexte général, cette forme permet de représenter une large classe de systèmes non-linéaires, non stationnaires ou incertains. Le principe de la représentation est d'isoler les éléments rendant la synthèse ou l'analyse plus difficile, du reste du système plus facilement abordable. La partie la plus accessible bénéficie souvent de propriétés utiles comme la linéarité, l'invariance dans le temps, etc. tandis que la partie plus complexe regroupe les termes non-linéaires, incertains ou variants à travers un opérateur généralement noté Δ .

Définition 2.5 : Transformation linéaire fractionnaire (LFT)

Soit M une matrice complexe partitionnée de la façon suivante :

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \in \mathbb{C}^{(p_1+p_2) \times (q_1+q_2)} \quad (2.19)$$

Pour deux matrices complexes $\Delta_u \in \mathbb{C}^{q_1 \times p_1}$, $\Delta_l \in \mathbb{C}^{q_2 \times p_2}$, les transformations linéaires fractionnelles supérieure $F_u(M, \Delta_u)$ et inférieure $F_l(M, \Delta_l)$ (respectivement pour l'anglais upper et lower) sont définies par :

$$\begin{aligned} F_u(M, \Delta_u) &= M_{22} + M_{21}\Delta_u(I - M_{11}\Delta_u)^{-1}M_{12} \\ F_l(M, \Delta_l) &= M_{11} + M_{12}\Delta_l(I - M_{22}\Delta_l)^{-1}M_{21} \end{aligned} \quad (2.20)$$

Ces représentations existent et sont dites bien posées si les matrices $(I - M_{11}\Delta_u)$ et $(I - M_{22}\Delta_l)$ sont inversibles.

Ces définitions correspondent aux formes standard illustrées par les deux schémas blocs de la figure 2.1.

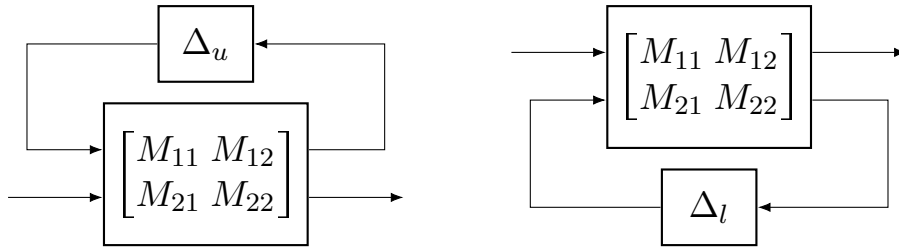


FIGURE 2.1 – Forme standard LFT supérieure et inférieure

Le formalisme LFT offre un cadre bien plus général pour la description des systèmes dynamiques que le cas matriciel présenté précédemment. Tout d'abord remarquons que cette définition inclut la représentation des systèmes linéaires invariants dans le temps, pour un système $G(z) = C(zI - A)^{-1}B + D$, une représentation du système discret est obtenu en considérant $\Delta = z^{-1}I$ ainsi :

$$G(z) = F_u \left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}, \Delta \right) \quad (2.21)$$

De même, la forme LFT se retrouve être un outil parfaitement adapté à la modélisation de systèmes incertains. En effet, il est généralement difficile de garantir l'exactitude entre le comportement du modèle mathématique et celui de processus physique. Ceci peut s'expliquer par la présence d'incertitude sur les paramètres ou par la simplification volontaire des équations mathématiques considérées (via des approximations ou tout simplement par d'insuffisances de modélisation). Les incertitudes de modélisation peuvent alors prendre la forme d'une matrice Δ matérialisant l'écart entre le comportement nominal et le comportement réel du processus. Afin de faciliter l'utilisation du modèle, il est habituel de représenter le modèle dynamique sous une forme telle que les incertitudes soient extraites du procédé. Le système physique incertain est alors représenté par une LFT sous la forme du système nominal bouclé sur une matrice d'incertitude représentant les erreurs de modélisation. Cette matrice peut-être de différentes natures et formes suivant les types d'incertitudes considérées : matrice de transfert

quelconque, diagonale par blocs, diagonale à deux blocs, réelles, complexes, etc. Finalement, les formes LFT offrent un formalisme standard très utilisé dans le domaine de la commande robuste.

Dans le cadre des systèmes LPV, l'intérêt de cette représentation a rapidement été mis en avant [Packard, 1994] pour devenir l'une des formes de modélisation les plus utilisées pour ce type de systèmes [Apkarian et Gahinet, 1995, El Ghaoui et Scorletti, 1996, Wu et Dong, 2006]. Cet éveil d'intérêt trouve son explication dans le fait que le formalisme LFT permet de représenter de façon exacte tout système dont les matrices $(A(\theta), B(\theta), C(\theta), D(\theta))$ dépendent de façon rationnelle ou polynomiale du vecteur de paramètres θ [Zhou *et al.*, 1996]. Le modèle LPV-LFT se décompose donc en deux éléments indissociables, où d'un côté le système LTI est interconnecté avec les paramètres variants qui sont regroupés d'un autre côté au sein de l'opérateur non-stationnaire $\Delta(\theta)$ qui est diagonal par construction.

Le formalisme LFT apporte finalement une représentation des systèmes LPV équivalente à la modélisation (2.17) précédemment introduite où les matrices dépendent du vecteur de paramètres. Considérons la forme standard $F_u(G, \Delta(\theta))$ où G est un système linéaire invariant défini par :

$$G := \begin{cases} x(k+1) = A x(k) + B_\Delta w(k) + B_u u(k) \\ z(k) = C_\Delta x(k) + D_{\Delta\Delta} w(k) + D_{\Delta u} u(k) \\ y(k) = C_y x(k) + D_{y\Delta} w(k) + D_{yu} u(k) \end{cases} \quad (2.22)$$

et l'interconnexion avec la matrice d'incertitude est donnée par :

$$w(k) = \Delta(\theta(k))z(k) \quad (2.23)$$

Le système est bien posée si $(I - D_{\Delta\Delta}\Delta(\theta))$ est non-singulière pour tout vecteur de paramètres $\theta \in \Theta$. Le système LPV-LFT (2.22)(2.23) est alors équivalent au système LPV de la forme de (2.17) dont les matrices sont données par :

$$\begin{cases} \tilde{A}(\theta) = A + B_\Delta\Delta(\theta)(I - D_{\Delta\Delta}\Delta(\theta))^{-1}C_\Delta \\ \tilde{B}(\theta) = B_u + B_\Delta\Delta(\theta)(I - D_{\Delta\Delta}\Delta(\theta))^{-1}D_{\Delta u} \\ \tilde{C}(\theta) = C_y + D_{y\Delta}\Delta(\theta)(I - D_{\Delta\Delta}\Delta(\theta))^{-1}C_\Delta \\ \tilde{D}(\theta) = D_{yu} + D_{y\Delta}\Delta(\theta)(I - D_{\Delta\Delta}\Delta(\theta))^{-1}D_{\Delta u} \end{cases} \quad (2.24)$$

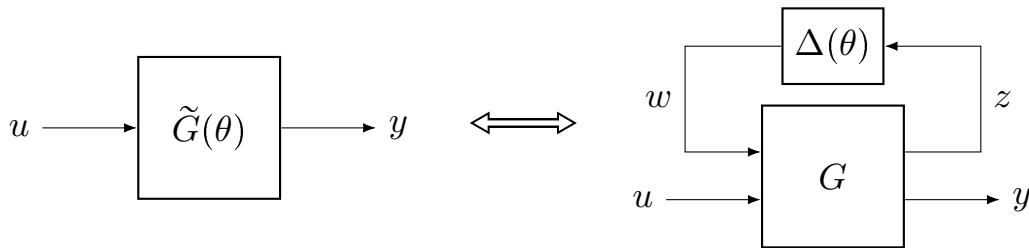


FIGURE 2.2 – Représentation LPV-LFT

SECTION 2.4

Analyse de stabilité et de performance des systèmes LPV-LFT

Cette section présente les principaux résultats d'analyse de stabilité et de performance dans le cadre discret. Le système LPV à l'étude est défini par sa représentation d'état donnée par :

$$\tilde{G}(\theta) = \left\{ \begin{array}{c} \left[\begin{array}{c} x(k+1) \\ y(k) \end{array} \right] \\ \left[\begin{array}{c|c} \tilde{A}(\theta(k)) & \tilde{B}(\theta(k)) \\ \tilde{C}(\theta(k)) & \tilde{D}(\theta(k)) \end{array} \right] \left[\begin{array}{c} x(k) \\ u(k) \end{array} \right] \end{array} \right. \quad (2.25)$$

où θ est le vecteur de paramètres variants.

Supposons qu'il existe un modèle LFT équivalent $F_u(G, \Delta(\theta))$ du système LPV $\tilde{G}(\theta)$ dont le comportement dynamique est décrit par :

$$F_u(G, \Delta(\theta)) = \left\{ \begin{array}{c} \left[\begin{array}{c} x(k+1) \\ z(k) \\ y(k) \end{array} \right] \\ \left[\begin{array}{c|c|c} A & B_\Delta & B_p \\ C_\Delta & D_{\Delta\Delta} & D_{\Delta p} \\ C_p & D_{p\Delta} & D_{pp} \end{array} \right] \left[\begin{array}{c} x(k) \\ w(k) \\ u(k) \end{array} \right] \\ w(k) = \Delta(\theta(k))z(k) \end{array} \right. \quad (2.26)$$

où G est un système LTI et la matrice $\Delta(\theta)$ contient les paramètres variants. De plus, la matrice $\Delta(\theta)$ respecte les propriétés suivantes :

- L'opérateur $\Delta(\theta)$ est bloc-diagonal tel que :

$$\Delta = \text{diag}(\delta_1 I_{r_1}, \dots, \delta_{n_\Delta} I_{r_{n_\Delta}}) \subset \mathbb{R}^{r \times r} \quad (2.27)$$

où r_i est le nombre d'occurrence du paramètre $\delta_i \in \mathbb{R}$ et on a alors $r = \sum_{i=1}^{n_\Delta} r_i$.

- En normalisant si besoin le système, les paramètres δ_i évoluent dans l'intervalle unitaire centré sur l'origine tel que $\delta_i \in [-1, 1]$. Ainsi, à l'aide du concept de gain \mathcal{L}_2 qui sera établi par la définition 2.7 de la section suivante, l'opérateur $\Delta(\theta)$ vérifie :

$$\|\Delta(\theta)\|_{\mathcal{L}_2} \leq 1 \quad (2.28)$$

L'objectif de cette section est, dans un premier temps, d'étudier la stabilité interne du système LPV (2.25) via sa forme LFT équivalente (2.26). Dans un second temps, le but est de garantir une performance entrée-sortie du système LPV-LFT (2.26). Un enjeu majeur de l'analyse est également de diminuer au maximum le conservatisme des résultats en tirant bénéfice des propriétés spécifiques (2.27) et (2.28) de l'opérateur $\Delta(\theta)$ associé aux paramètres variants.

Les résultats d'analyses de systèmes LPV présentés dans cette section peuvent être facilement implémentés sous MATLAB à l'aide des fonctions disponibles dans *LPVTools* [Balas *et al.*, 2015] développé par MUSYN Inc.

2.4.1 Notions de stabilité des systèmes LPV

Avant d'apporter des méthodes d'analyse des systèmes LPV, définissons dans un premier temps quelques concepts essentiels.

Définition 2.6 : Espace et norme ℓ_2

Soit un signal $s(k) : \mathbb{R}_+ \rightarrow \mathbb{R}^n$, la norme ℓ_2 du signal est définie par :

$$\|s(k)\|_2 := \sqrt{\sum_{k=0}^{\infty} s(k)^T s(k)} \quad (2.29)$$

Elle est associée à l'espace ℓ_2 qui regroupe l'ensemble des signaux causaux de carré sommable soit :

$$\ell_2 := \left\{ s(k) : \mathbb{R}_+ \rightarrow \mathbb{R}^n \mid \|s(k)\|_2 < \infty \right\} \quad (2.30)$$

Afin de quantifier les caractéristiques d'un système, nous nous intéresserons à la norme induite \mathcal{L}_2 ou gain \mathcal{L}_2 du système qui est une généralisation de la norme \mathcal{H}_∞ utilisée pour les systèmes LTI. Cette norme a une interprétation physique intéressante, puisqu'elle correspond à un gain énergétique entre les signaux d'entrées et ceux de sortie.

Définition 2.7 : Gain \mathcal{L}_2

Soit un système M d'entrée w et de sortie z , son gain \mathcal{L}_2 (ou sa norme \mathcal{L}_2 induite) est défini par :

$$\|M\|_{\mathcal{L}_2} := \sup_{\substack{w \in \ell_2 \\ \|w\|_2 \neq 0}} \frac{\|z\|_2}{\|w\|_2} \quad (2.31)$$

Le gain \mathcal{L}_2 est donc le plus grand gain possible entre l'énergie en entrée et l'énergie en sortie. Pour exemple, une entrée d'énergie unitaire injectée à un système M engendrera au plus une sortie d'énergie $\|M\|_{\mathcal{L}_2}$. Notons que le gain \mathcal{L}_2 est défini pour n'importe quel type de système stable, qu'il soit linéaire, non-linéaire, variant dans le temps, etc. La norme induite \mathcal{L}_2 est équivalente pour les systèmes linéaires invariants dans le temps à la norme \mathcal{H}_∞ .

Définition 2.8 : Ensemble \mathcal{RL}_2

\mathcal{RL}_2 définit le sous-espace des opérateurs de gain \mathcal{L}_2 fini.

2.4.2 Théorème du petit gain

Considérons l'interconnexion standard de deux systèmes comme le montre la figure 2.3. Le théorème du petit gain [Zames, 1966, Desoer et Vidyasagar, 1975, Zhou *et al.*, 1996] permet d'établir une condition suffisante de stabilité de l'interconnexion de l'opérateur M avec le second opérateur Δ .

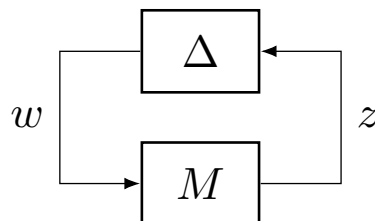


FIGURE 2.3 – Système d'interconnexion pour le théorème du petit gain

Théorème 2.1 : Théorème du petit gain [Zames, 1966]

Soient deux opérateurs $M \in \mathcal{RL}_2$ et $\Delta \in \mathcal{RL}_2$, alors l'interconnexion de la figure 2.3 est stable si

$$\|\Delta M\|_{\mathcal{L}_2} < 1 \quad (2.32)$$

Dans le cas de l'étude d'un opérateur M qui est un système LTI alors la norme \mathcal{L}_2 correspond à la norme \mathcal{H}_∞ . De plus, lorsque l'objectif est de garantir la stabilité pour tout opérateur Δ de gain \mathcal{L}_2 borné, le théorème du petit gain apporte une condition nécessaire et suffisante formulée par le corollaire suivant.

Corollaire 2.1 : Corollaire du théorème du petit gain [Hiret, 1999]

Soient $M \in \mathcal{RH}_\infty$, $\Delta \in \mathcal{RL}_2$ et $\gamma > 0$, le système en boucle fermée de la figure 2.3 est stable pour tout Δ tel que $\|\Delta\|_{\mathcal{L}_2} \leq \gamma$ si et seulement si $\|M\|_\infty < \gamma^{-1}$

Le théorème du petit gain apporte une preuve de stabilité dont la vérification est relativement accessible, cependant il mène généralement à des résultats pessimistes aussi caractérisés de conservatifs. Premièrement, aucune information sur la phase n'est prise en compte et seul le gain est examiné. Cette considération est un réel atout pour la généralité des résultats néanmoins elle contribue nécessairement à un certain conservatisme. En second lieu, dans une majorité des cas, la structure et la nature des opérateurs Δ sont connues et elles ne sont cependant pas prise en compte. La possibilité des matrices Δ est donc restreinte à un sous-ensemble de \mathcal{RL}_2 . Or ces informations ne sont pas exploitées dans les conditions de l'analyse.

Finalement, une première analyse de stabilité du système LPV (2.25) peut être effectuée à l'aide des résultats du théorème du petit gain. Nous considérons alors l'interconnexion définie par sa représentation LFT (2.26) et une classe d'opérateur Δ englobant la matrice des paramètres variants. D'après le corollaire 2.1, une condition de stabilité nécessaire et suffisante peut être formulée au sujet du sous transfert G_Δ reliant w à z définie par $G_\Delta(z) := C_\Delta(zI - A)^{-1}B_\Delta + D_{\Delta\Delta}$.

Corollaire 2.2 : Condition de stabilité interne des systèmes LPV-LFT

Soient $G_\Delta(z) = (A, B_\Delta, C_\Delta, D_{\Delta\Delta}) \in \mathcal{RH}_\infty$ et $\Delta(\theta) \in \mathcal{RL}_2$, le système LPV-LFT (2.26) est stable de manière interne, pour toute matrice Δ telle que $\|\Delta\|_{\mathcal{L}_2} \leq 1$ si et seulement si

$$\|G_\Delta\|_\infty < 1 \quad (2.33)$$

2.4.3 Théorème du petit gain avec multiplicateurs

Le principal objectif des multiplicateurs est d'améliorer les conclusions du théorème du petit gain en tenant compte de la structure de l'opérateur Δ et ainsi réduire le conservatisme des résultats. Les multiplicateurs qui commutent avec la structure de Δ sont alors insérés dans le système d'interconnexion sans changer la nature du problème. Néanmoins, ils introduisent des degrés de liberté supplémentaires dans sa résolution, permettant de réduire ainsi significativement le conservatisme [Doyle, 1982, Safonov, 1993].

La matrice d'incertitude est généralement de structure diagonale comme évoqué dans (2.27), ce qui limite l'étude à un sous-ensemble de possibilités. Définissons alors l'ensemble des matrices possédant cette même structure ainsi que l'ensemble des multiplicateurs ou matrices de *scalings* symétriques qui commutent avec elle.

Définition 2.9 : Matrices structurées

L'ensemble $\underline{\Delta}$ des matrices réelles de structure diagonale est défini tel que :

$$\underline{\Delta} := \left\{ \Delta = \text{diag}(\delta_1 I_{r_1}, \dots, \delta_{n_\Delta} I_{r_{n_\Delta}}), \delta_i \in \mathbb{R} \right\} \quad (2.34)$$

Définition 2.10 : Multiplicateurs associés à $\underline{\Delta}$

L'ensemble des matrices de scalings associé avec la structure $\underline{\Delta}$ est défini par :

$$L_{\underline{\Delta}} = \left\{ L = L^T > 0 \mid L\Delta = \Delta L, \forall \Delta \in \underline{\Delta} \right\} \quad (2.35)$$

Remarque 2.5. Le caractère variant dans le temps de Δ , impose que les matrices qui commutent avec cet opérateur soient constantes et non des matrices de transfert. De plus, en considérant la structure diagonale de la matrice $\Delta \in \underline{\Delta}$, les matrices de scalings sont définies selon une structure bloc diagonale. Les multiplicateurs sont donc de la forme :

$$\forall L \in L_{\underline{\Delta}} : L = \text{diag}(L_1, \dots, L_{n_\Delta}), L_i \in \mathbb{R}^{r_i \times r_i} \quad (2.36)$$

où chaque sous-matrice L_i est une matrice pleine mais symétrique par construction dont les dimensions correspondent au nombre d'occurrence du paramètre δ_i associé. Plus de détails sur ces matrices sont référés dans [Apkarian et Gahinet, 1995].

Selon les définitions précédentes, le rôle de la matrice L est d'inclure l'information de structure à propos de Δ à travers une propriété de commutation. Le problème abordé par le théorème du petit gain est alors remanié avec les multiplicateurs comme le montre la figure 2.4. En vertu des propriétés de la matrice non singulière $L \in L_{\underline{\Delta}}$, il apparaît l'identité suivante $\Delta = L^{-1}\Delta L$ et l'équivalence des problèmes considérés. D'après le corollaire 2.1, il est alors possible d'établir une formulation du théorème du petit gain équivalente en faisant intervenir les multiplicateurs.

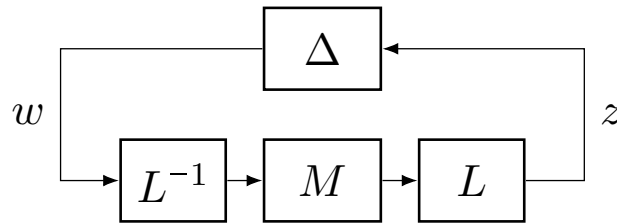


FIGURE 2.4 – Système d'interconnexion pour le théorème du petit gain avec multiplicateurs

Corollaire 2.3 : Corollaire du théorème du petit gain avec multiplicateurs

Soient $M \in \mathcal{RH}_\infty$ et $\gamma > 0$, le système en boucle fermée de la figure 2.4 est stable pour tout $\Delta \in \underline{\Delta}$ avec $\|\Delta\|_{\mathcal{L}_2} \leq \gamma$ si il existe $L \in L_{\underline{\Delta}}$ telle que $\|LML^{-1}\|_\infty < \gamma^{-1}$

En s'appuyant sur les multiplicateurs, le lemme 2.1 borné réel peut être reconsidéré afin d'apporter une caractérisation simple du critère \mathcal{H}_∞ . Cette condition malheureusement suffisante mais ayant pour avantage d'être simple est fondée sur l'existence d'une fonction de Lyapunov quadratique unique qui peut être obtenue sous la forme du lemme suivant.

Lemme 2.2 : Lemme borné réel discret avec multiplicateurs [Apkarian et Gahinet, 1995]

Soit le système $M(z) = (A, B, C, D)$, le système est asymptotiquement stable et il existe $L \in L_{\underline{\Delta}}$ telle que la norme \mathcal{H}_∞ vérifie

$$\left\| L^{1/2} M L^{-1/2} \right\|_\infty < \gamma \quad (2.37)$$

si et seulement si il existe une matrice symétrique définie positive X solution de l'inégalité :

$$\begin{bmatrix} A^T X A - X & A^T X B & C^T L \\ B^T X A & B^T X B - \gamma L & D^T L \\ LC & LD & -\gamma L \end{bmatrix} < 0 \quad (2.38)$$

Finalement, l'application du théorème 2.3 du petit gain avec multiplicateurs aboutit à la conclusion de stabilité suivante pour le système LPV-LFT (2.26).

Corollaire 2.4 : Condition de stabilité quadratique interne des systèmes LPV-LFT

Soient $G_\Delta(z) = (A, B_\Delta, C_\Delta, D_{\Delta\Delta}) \in \mathcal{RH}_\infty$ et $\Delta(\theta) \in \mathcal{RL}_2$, le système LPV-LFT (2.26) est asymptotiquement stable de manière interne, pour toute matrice structurée $\Delta(\theta) \in \underline{\Delta}$ telle que $\|\Delta(\theta)\|_{\mathcal{L}_2} \leq 1$ et pour toute trajectoire $\theta(k) \subset \Theta$ si il existe une matrice symétrique définie positive X et $L \in L_{\underline{\Delta}}$ solution de l'inégalité :

$$\begin{bmatrix} A^T X A - X & A^T X B_\Delta & C_\Delta^T L \\ B_\Delta^T X A & B_\Delta^T X B_\Delta - L & D_{\Delta\Delta}^T L \\ LC_\Delta & LD_{\Delta\Delta} & -L \end{bmatrix} < 0 \quad (2.39)$$

Démonstration. Ce corollaire est déduit en combinant le corollaire 2.3 et le lemme 2.2 pour le système LPV-LFT (2.26). \square

2.4.4 Analyse de performance des systèmes LPV-LFT

L'analyse de robustesse abordée jusqu'à maintenant, se concentre sur les conditions de stabilité interne des systèmes LPV. En effet, l'étude a été menée par la vérification que l'interconnexion $F_u(G_\Delta, \Delta(\theta))$ est stable pour toute matrice d'incertitude structurée et normalisée, c'est-à-dire $\Delta(\theta) \in \underline{\Delta}$ et $\|\Delta(\theta)\|_{\mathcal{L}_2} \leq 1$. Nous considérons dans cette partie un problème plus général qui est l'analyse de performance. L'étude porte alors sur le fait que non seulement la stabilité interne du système doit être garantie, mais également qu'un certain niveau de performance doit être assuré. Dans le cadre de l'analyse de performance, le critère utilisé est basé sur un indicateur entrée-sortie qui correspond généralement au gain \mathcal{L}_2 pour les systèmes LPV et à la norme \mathcal{H}_∞ dans le cas des systèmes LTI.

Cette partie porte donc sur l'analyse de performance d'un système LPV, c'est-à-dire la détermination du gain \mathcal{L}_2 entre l'entrée et la sortie d'un système M sous sa forme LFT. Le

principe de l'approche consiste à se ramener à un problème équivalent de stabilité robuste comme l'illustre la figure 2.5. Le problème d'analyse de performance est transformé en un problème d'analyse de stabilité en introduisant un bloc d'incertitudes fictif Δ_p non structuré appartenant à \mathcal{RH}_∞ . L'analyse exploite alors le principe suivant : si le système d'interconnexion de la figure 2.5b est stable pour tout bloc admissible tel que $\|\Delta\|_{\mathcal{L}_2} < 1$ et pour tout $\Delta_p \in \mathcal{RH}_\infty$ tels que $\|\Delta_p\|_{\mathcal{L}_2} < \gamma^{-1}$ alors l'interconnexion de la figure 2.5a est stable et $\|F_u(G, \Delta)\|_{\mathcal{L}_2} < \gamma$.

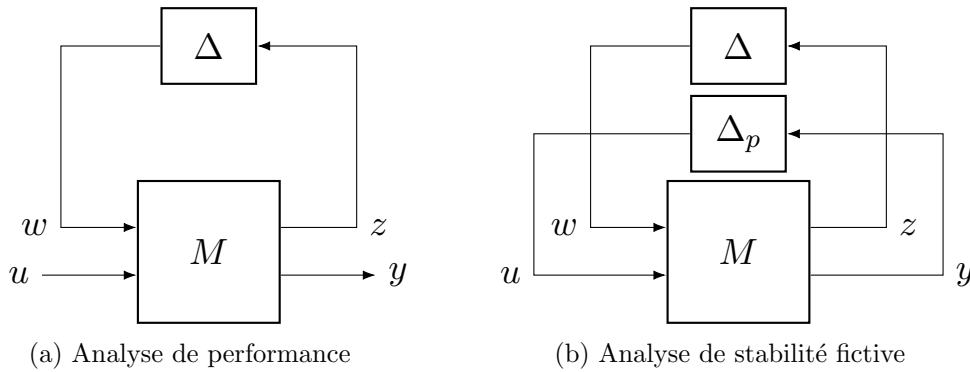


FIGURE 2.5 – Analyse de performance des systèmes LPV-LFT via une analyse de stabilité

L'objectif de l'analyse de performance consiste donc, par le biais d'une analyse de stabilité, à déterminer la plus petite incertitude $\Delta_p \in \mathcal{RH}_\infty$ au sens de sa norme \mathcal{H}_∞ qui déstabilise le système de l'interconnexion de la figure 2.5b. Ce problème peut être résolu à l'aide du corollaire 2.3 du théorème du petit gain avec multiplicateurs dont les multiplicateurs sont adaptés à l'opérateur diagonal par blocs $\text{diag}(\Delta, \Delta_p)$. En prenant en considération la nature structurée de $\Delta \in \underline{\Delta}$ via une matrice $L \in L_{\underline{\Delta}}$ et celle non structurée de Δ_p , l'analyse peut être menée selon le schéma et les multiplicateurs de la figure 2.6. Une extension du corollaire 2.3 du théorème du petit gain avec multiplicateurs permet alors d'énoncer le résultat suivant.

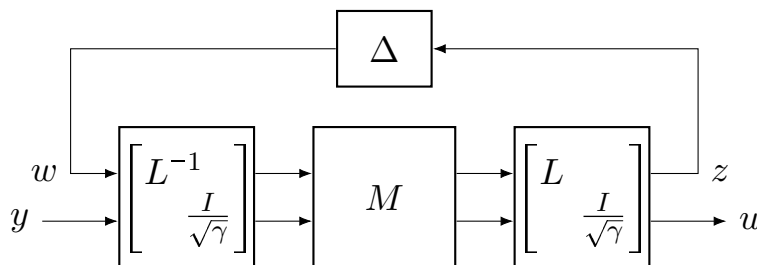


FIGURE 2.6 – Analyse de performance des systèmes LPV-LFT avec multiplicateurs

Corollaire 2.5 : Corollaire du théorème du petit gain pour l'analyse de performance

Soient $M \in \mathcal{RH}_\infty$ et $\gamma > 0$, la norme \mathcal{L}_2 du système $F_u(M, \Delta)$ vérifie :

$$\|F_u(M, \Delta)\|_{\mathcal{L}_2} < \gamma \quad (2.40)$$

si il existe une matrice $L \in L_\Delta$ telle que :

$$\left\| \begin{bmatrix} L & 0 \\ 0 & \frac{I}{\sqrt{\gamma}} \end{bmatrix} M \begin{bmatrix} L^{-1} & 0 \\ 0 & \frac{I}{\sqrt{\gamma}} \end{bmatrix} \right\|_\infty < 1 \quad (2.41)$$

Finalement, l'analyse de performance entrée-sortie du système LPV-LFT (2.26) peut-être évaluée selon le lemme 2.2 borné réel avec multiplicateurs.

Corollaire 2.6 : Analyse de performance quadratique des systèmes LPV-LFT

Soient $\Delta(\theta) \in \mathcal{RL}_2$ et $\gamma > 0$, le système LPV-LFT (2.26) est asymptotiquement stable et vérifie

$$\|F_u(G(z), \Delta(\theta))\|_{\mathcal{L}_2} < \gamma \quad (2.42)$$

pour toute matrice structurée $\Delta(\theta) \in \underline{\Delta}$ telle que $\|\Delta(\theta)\|_{\mathcal{L}_2} \leq 1$ et pour toute trajectoire $\theta(k) \in \Theta$ si il existe une matrice symétrique définie positive X et $L \in L_\Delta$ solution de l'inégalité :

$$\begin{bmatrix} A^T X A - X & A^T X B_\Delta & A^T X B_p & C_\Delta^T L & C_p^T \\ B_\Delta^T X A & B_\Delta^T X B_\Delta - L & B_\Delta^T X B_p & D_{\Delta\Delta}^T L & D_{p\Delta} \\ B_p^T X A & B_p^T X B_\Delta & B_p^T X B_p - \gamma I & D_{\Delta p}^T L & D_{pp} \\ LC_\Delta & LD_{\Delta\Delta} & LD_{\Delta p} & -L & 0 \\ C_p & D_{p\Delta} & D_{pp} & 0 & -\gamma I \end{bmatrix} < 0 \quad (2.43)$$

Démonstration. Ce corollaire est déduit en combinant le corollaire 2.5 et le lemme 2.2. En effet, à partir des multiplicateurs de la forme :

$$\begin{bmatrix} L & 0 \\ 0 & \frac{I}{\sqrt{\gamma}} \end{bmatrix} \quad (2.44)$$

La condition (2.37) du lemme bornée réel pour le système LPV-LFT (2.26) devient alors :

$$\begin{bmatrix} A^T X A - X & A^T X B_\Delta & \gamma^{-\frac{1}{2}} A^T X B_p & C_\Delta^T L & \gamma^{-\frac{1}{2}} C_p^T \\ B_\Delta^T X A & B_\Delta^T X B_\Delta - L & \gamma^{-\frac{1}{2}} B_\Delta^T X B_p & D_{\Delta\Delta}^T L & \gamma^{-\frac{1}{2}} D_{p\Delta} \\ \gamma^{-\frac{1}{2}} B_p^T X A & \gamma^{-\frac{1}{2}} B_p^T X B_\Delta & \gamma^{-1} B_p^T X B_p - I & \gamma^{-\frac{1}{2}} D_{\Delta p}^T L & \gamma^{-1} D_{pp} \\ LC_\Delta & LD_{\Delta\Delta} & \gamma^{-\frac{1}{2}} LD_{\Delta p} & -L & 0 \\ \gamma^{-\frac{1}{2}} C_p & \gamma^{-\frac{1}{2}} D_{p\Delta} & \gamma^{-1} D_{pp} & 0 & -I \end{bmatrix} < 0 \quad (2.45)$$

Finalement, l'équation (2.43) s'obtient en appliquant l'opération de normalisation par $\gamma^{\frac{1}{2}}$ sur le canal de performance. \square

Conclusions

Ce chapitre nous a permis de rappeler les principaux outils et résultats d'analyse de robustesse en stabilité et en performance des systèmes LPV-LFT. Les méthodes d'estimation de la robustesse reposent sur la résolution numérique de problèmes d'optimisation visant à minimiser un objectif linéaire sous contraintes LMI. Le conservatisme des outils présentés n'est cependant pas négligeable. En premier lieu, le théorème du petit gain procure des résultats souvent pessimistes de par sa généralité même si des restrictions supplémentaires peuvent être considérées à travers l'ajout de multiplicateurs. De plus, les problèmes LMI à résoudre qui sont fondés sur la théorie de la stabilité de Lyapunov, fournissent une condition suffisante puisqu'ils considèrent exclusivement une fonction de Lyapunov quadratique qui ne dépend pas des paramètres variants du système.

Une alternative intéressante consiste à utiliser des fonctions de Lyapunov paramétrées puisqu'elles permettent d'introduire des bornes sur les vitesses de variations des paramètres. Dans le cas contraire, la prise en compte de variations arbitrairement rapides impose de fortes contraintes et des résultats plus pessimistes. En contrepartie, une telle méthode conduit le plus souvent à une formulation des problèmes LMI dont la complexité de résolution surpasse largement celle de l'approche quadratique. La difficulté numérique est alors difficilement compatible avec la dimension des problèmes adressés dans nos travaux et les couches d'optimisation supplémentaires que nous proposerons dans le chapitre 4 pour l'apprentissage de contrôleurs robustes tenant compte de critère de stabilité.

Modélisation des SSNN pour la commande robuste

Sommaire

3.1	Introduction	62
3.2	Représentation des SSNN par des LPV-LFT	62
3.2.1	Formulation quasi-LPV	62
3.2.2	Première formulation LPV-LFT	65
3.2.3	Formulation quasi-LPV non-biaisée	67
3.2.4	Formulation LPV-LFT normalisée	69
3.3	Modélisation des associations de SSNN	73
3.3.1	La mise en série de SSNN est un SSNN	73
3.3.2	Prise en compte des retards dans la forme SSNN finale	74
3.3.3	L'interconnexion de SSNN est un SSNN	75
3.4	Analyse des SSNN via la formulation LPV-LFT	76
3.4.1	Analyse de stabilité et de performance des SSNN	77
3.4.2	Analyse des SSNN avec filtrage des paramètres	78
3.4.3	Analyse des SSNN par μ -analyse	79
3.5	Conclusions	81

SECTION 3.1

Introduction

Dans ce chapitre, nous nous intéressons à la modélisation des réseaux de neurones dans le cadre de l'automatique. Le travail se concentre sur les structures de réseaux pouvant être décrites par une représentation d'état c'est-à-dire les SSNN (*State-Space Neural Networks*) qui ont été introduits dans le chapitre 1. Par nature, ces structures se retrouvent privilégiées dans le cadre de notre étude puisqu'un premier pas dans le domaine du contrôle est déjà réalisé. La représentation d'état du réseau permet de l'appréhender comme un système dynamique décrit par des équations non-linéaires. Néanmoins l'approche peut être approfondie davantage, puisqu'un lien peut être établi avec les systèmes LPV qui ont été présentés au chapitre 2. Le rapprochement permet de bénéficier de l'abondante littérature sur les méthodes LPV en plus d'améliorer l'interprétation de ses objets hermétiques que peuvent être les réseaux de neurones.

Dans la section 3.2, nous présentons la méthodologie afin d'obtenir le modèle LPV-LFT équivalente d'un réseau de type SSNN. Nous développons dans la section 3.4, les outils d'analyse de stabilité et de performance des SSNN à partir des résultats du chapitre 2 sur les systèmes LPV. Nous modéliserons dans la section 3.3, différentes interconnexions de SSNN. Une application directe de cette partie est notamment la détermination du modèle du réseau incluant les systèmes de retards.

SECTION 3.2

Représentation des SSNN par des LPV-LFT

Les réseaux de topologie SSNN qui peuvent se représenter à l'aide d'une représentation d'état ont été introduits dans la section 1.4.4. Nous nous focalisons sur cette architecture dont la représentation pour un réseau à $L + 1$ couches cachées (c'est-à-dire L couches cachées non-linéaires) est rappelée ci-après avec $W_1 = [W_1^x \ W_1^u]$:

$$H^{\text{NN}} := \begin{cases} h_1(k) &= \sigma_1(W_1^x x_1(k) + W_1^u u(k) + b_1) \\ h_\ell(k) &= \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 2, \dots, L \\ x(k+1) &= Ax(k) + Bu(k) + W_x h_L(k) + b_x \\ y(k) &= Cx(k) + Du(k) + b_y \end{cases} \quad (3.1)$$

Dans cette section, nous expliquons comment les SSNN peuvent être interprétés comme des systèmes LPV. La représentation LFT issue de cette interprétation fait la passerelle avec les outils de la commande robuste. Les idées décrites ici ont été développées à partir de celles introduites par [Suykens *et al.*, 1995a] et élargies par la suite dans [Bendtsen et Trangbæk, 2000].

3.2.1 Formulation quasi-LPV

L'écart qui sépare le réseau de neurones du monde linéaire sont justement ses fonctions d'activation non-linéaires. En utilisant les méthodes de modélisation quasi-LPV, les dynamiques non-linéaires peuvent être interprétées à l'aide de systèmes linéaires dépendant de l'état. Les non-linéarités des fonctions d'activation de réseau peuvent alors être isolées de la partie linéaire afin de les appréhender en tant que paramètres variants. Une telle reformulation a déjà été

entreprise dans la littérature dans le cadre d'un réseau de neurones comportant une seule couche cachée. Nous pouvons par exemple évoquer les travaux de [Kretchmar, 2000] ou ceux de [Abbas et Werner, 2008a].

Pour le SSNN (3.1), définissons l'entrée de chaque couche cachée par :

$$\begin{cases} \xi_1(k) = W_1^x x(k) + W_1^u u(k) + b_1 \\ \xi_\ell(k) = W_\ell h_{\ell-1}(k) + b_\ell \quad \text{pour } \ell = 2, \dots, L \end{cases} \quad (3.2)$$

En se focalisant à l'échelle du neurone et en utilisant une notation terme à terme, l'équation de chaque $j^{\text{ème}}$ neurone non-linéaire de chaque couche cachée $\ell = 1, \dots, L$ peut se formuler comme suit :

$$\begin{aligned} h_\ell^j(k) &= \sigma_\ell(\xi_\ell^j(k)) \\ &= \frac{\sigma_\ell(\xi_\ell^j(k))}{\xi_\ell^j(k)} \xi_\ell^j(k) \\ &= \theta_\ell^j(k) \xi_\ell^j(k) \end{aligned} \quad (3.3)$$

où $\theta_\ell^j(k)$ représente le paramètre variant associé à la non-linéarité de chaque neurone :

$$\theta_\ell^j(k) = \frac{\sigma_\ell(\xi_\ell^j(k))}{\xi_\ell^j(k)} \quad (3.4)$$

Remarque 3.1. Notons que la présence d'un quotient peut poser problème lors du calcul des paramètres variants lorsque l'entrée devient nulle. Les paramètres variants sont donc ainsi définis :

$$\theta_\ell^j(k) = \begin{cases} \frac{\sigma_\ell(\xi_\ell^j(k))}{\xi_\ell^j(k)} & \text{si } \xi_\ell^j(k) \neq 0 \\ 1 & \text{sinon} \end{cases} \quad (3.5)$$

La valeur unitaire obtenue lorsque l'entrée s'annule est propre aux fonctions d'activation *tanh*, *saturation* et *RELU* qui sont présentées dans le tableau 3.1. Cette valeur peut être prouvée facilement par passage à la limite par la règle de l'Hôpital ou l'utilisation des développements limités.

Remarque 3.2. La fonction d'activation sigmoïde qui ne respecte pas les conditions de secteur borné peut tout de même être étudiée par une représentation à l'aide de la fonction tangente hyperbolique comme suit :

$$\text{sigmoïde}(x) = \frac{1}{2} + \frac{1}{2} \tanh\left(\frac{x}{2}\right) \quad (3.6)$$

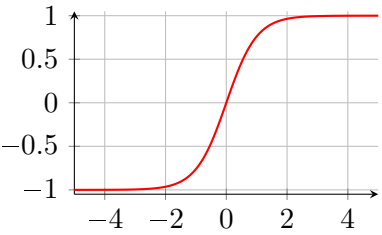
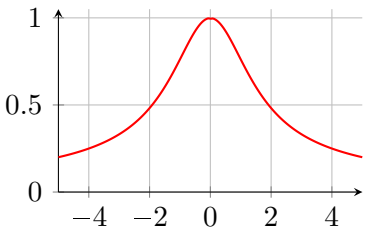
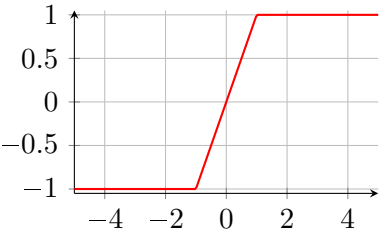
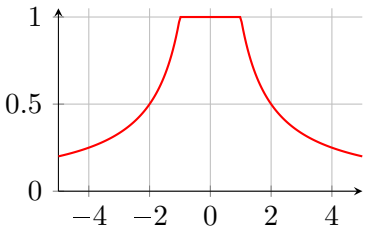
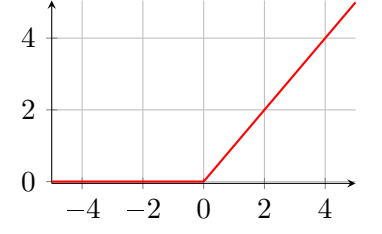
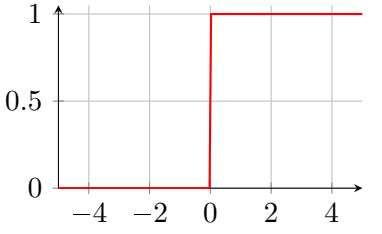
En revenant à une notation matricielle, les non-linéarités des couches cachées peuvent ainsi être caractérisées selon :

$$h_\ell(k) = \sigma_\ell(\xi_\ell(k)) = \Theta_\ell(k) \xi_\ell(k) \quad (3.7)$$

où $\Theta_\ell(k)$ est la matrice des paramètres variants associée à chaque couche cachée définie par :

$$\Theta_\ell(k) = \text{diag}\left(\theta_\ell^j(k)\right) \quad (3.8)$$

TABLE 3.1 – Fonctions d'activation et leur gain

Fonctions d'activation	Figure de la fonction $f(x)$	Figure du gain $f(x)/x$
tanh		
saturation		
RELU		

Finalement, en utilisant les paramètres variants, le SSNN (3.1) s'exprime selon un système LPV dont la forme est donnée par la proposition suivante.

Proposition 3.1 : Formulation quasi-LPV d'un SSNN

Soit un SSNN (3.1), le SSNN admet une représentation quasi-LPV telle que :

$$\begin{cases} h_1(k) = \Theta_1(W_1^x x(k) + W_1^u u(k) + b_1) \\ h_\ell(k) = \Theta_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 2, \dots, L \\ x(k+1) = Ax(k) + Bu(k) + W_x h_L(k) + b_x \\ y(k) = Cx(k) + Du(k) + b_y \end{cases} \quad (3.9)$$

où les paramètres variants sont définis selon les équations (3.4) et (3.2).

Cette représentation du réseau apporte plusieurs réflexions essentielles :

- Le modèle LPV est une reformulation exacte du calcul du comportement du SSNN, les équations dynamiques n'ont pas été altérées et la fonctionnalité du réseau reste inchangée.
- La forme LPV obtenue sépare distinctement les non-linéarités des couches cachées du reste des opérations linéaires du réseau. Les équations de dynamique se composent donc

d'application de matrices linéaires et des matrices non-linéaires Θ_ℓ , $\ell = 1, \dots, L$.

- Le nombre de paramètres variants correspond exactement au nombre de neurones non-linéaires présents dans le réseau. Cette équivalence peut avoir d'importantes conséquences quant à la dimension du système LPV. En effet, d'un côté, il est important que le réseau dispose de suffisamment de neurones afin d'identifier le processus avec un bon niveau de précision [Funahashi, 1989]. Cependant le modèle LPV de grande dimension résultant d'un réseau surdimensionné peut rapidement aboutir à des problèmes insolubles par la majorité des algorithmes d'analyses ou de synthèses basés sur l'optimisation semi-définie positive et les LMI [Balas *et al.*, 2015].
- De par la définition des paramètres variants, le modèle LPV ne peut être obtenu que pour des réseaux dont les fonctions d'activations respectent une condition de secteurs, c'est-à-dire pour lesquelles l'équation (3.4) a des valeurs finies et donc pour lesquelles les paramètres variants sont bornés.

Revenons sur la définition des paramètres variants donnée par l'équation (3.4) et leur interprétation. Pour chaque couche non-linéaire et chaque entité qui la compose, la fonction θ_ℓ^j calcule le rapport entre la sortie et l'entrée du neurone, ce qui correspond au gain de la fonction d'activation. Les figures présentées dans le tableau 3.1 illustre les différentes fonctions d'activation ainsi que leur gain. Les figures mettent en évidence l'importance de la propriété de secteur que les fonctions non-linéaires doivent respecter afin d'obtenir un gain fini. L'ensemble des fonctions présentées sont des fonctions à secteur borné appartenant au secteur $[0, 1]$, ainsi, on retrouve par définition cette même propriété pour les paramètres variants :

$$\theta_\ell^j \in [0, 1], \quad \forall \ell = 1, \dots, L \quad \forall j = 1, \dots, n_{h_\ell} \quad (3.10)$$

L'interprétation des paramètres variants peut être développée d'avantage afin de caractériser qualitativement les non-linéarités du réseau [Suykens *et al.*, 1995b]. En effet, la fonction θ_ℓ^j apporte une indication du degré de distorsion du neurone, c'est-à-dire de sa non-linéarité. Une valeur proche de 1 traduit un comportement avoisinant celui du linéaire tandis qu'une valeur proche de 0 témoigne d'un fort caractère non-linéaire puisque la fonction d'activation s'approche des saturations. Pour un signal d'entrée du réseau donné, les éléments θ_ℓ^j peuvent être calculés au cours du temps ce qui permet d'évaluer la contribution des aspects non-linéaires de chaque neurone. Cette information peut demeurer indicative, mais peut également par exemple être utilisée pour orienter le choix de la structure du réseau de neurones lors de la phase d'apprentissage.

3.2.2 Première formulation LPV-LFT

La représentation LPV déterminée par l'équation (3.9) expose une séparation apparente entre les applications linéaires et les fonctions d'activations non-linéaires. Ceci permet d'établir comme dans [Suykens *et al.*, 1995b], une représentation du système LPV sous forme d'une interconnexion exposée par la figure 3.1. Cette nouvelle représentation isole les non-linéarités du reste du système, elle correspond à une représentation LFT dont la matrice des paramètres variants est une matrice diagonale obtenue en posant :

$$w_\ell(k) = \Theta_\ell(k)z_\ell(k) \text{ avec } z_\ell(k) = \xi_\ell(k) \quad (3.11)$$

ainsi

$$w(k) = \begin{bmatrix} w_1(k) \\ \vdots \\ w_L(k) \end{bmatrix}, \quad z(k) = \begin{bmatrix} z_1(k) \\ \vdots \\ z_L(k) \end{bmatrix} \quad \text{et} \quad \Theta(k) = \begin{bmatrix} \Theta_1(k) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \Theta_L(k) \end{bmatrix}$$

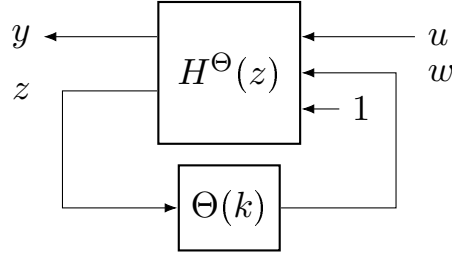


FIGURE 3.1 – Représentation LPV-LFT d'un SSNN isolant les non-linéarités

Finalement, le SSNN (3.1) admet une représentation LPV-LFT isolant les non-linéarités donnée par la proposition suivante.

Proposition 3.2 : Représentation LPV-LFT d'un SSNN biaisé

Soit un SSNN (3.1) et sa formulation LPV (3.9), le SSNN admet une représentation LPV-LFT donnée par :

$$\left\{ \begin{array}{l} \left[\begin{array}{c} x(k+1) \\ y(k) \\ z_1(k) \\ z_2(k) \\ \vdots \\ z_L(k) \end{array} \right] = \left[\begin{array}{c|c|c|c|c|c|c|c} A & B & 0 & \cdots & 0 & W_x & b_x \\ \hline C & D & 0 & \cdots & 0 & 0 & b_y \\ \hline W_1^x & W_1^u & 0 & \cdots & 0 & 0 & b_1 \\ 0 & 0 & W_2 & 0 & 0 & 0 & b_2 \\ \vdots & \vdots & 0 & \ddots & 0 & \vdots & \vdots \\ 0 & 0 & 0 & 0 & W_L & 0 & b_L \end{array} \right] \left[\begin{array}{c} x(k) \\ u(k) \\ w_1(k) \\ \vdots \\ w_{L-1}(k) \\ w_L(k) \\ 1 \end{array} \right] \\ w_\ell(k) = \Theta_\ell(k)z_\ell(k) \end{array} \right. \quad (3.12)$$

où les paramètres variants associés aux non-linéarités de chaque couche cachée $\ell = 1, \dots, L$ sont définis par :

$$\left\{ \begin{array}{l} \Theta_\ell(k) = \text{diag} \left(\theta_\ell^j(k) \right) \\ \theta_\ell^j(k) = \frac{\sigma_\ell(\xi_\ell^j(k))}{\xi_\ell^j(k)} \end{array} \right. \quad (3.13)$$

avec

$$\left\{ \begin{array}{l} \xi_1(k) = W_1^x x(k) + W_1^u u(k) + b_1 \\ \xi_\ell(k) = W_\ell \Theta_{\ell-1} \xi_{\ell-1} + b_\ell \quad \text{pour } \ell = 2, \dots, L \end{array} \right. \quad (3.14)$$

Cette formulation dite biaisée fait référence aux biais des différentes couches du réseau qui nécessite la présence d'une perturbation unitaire.

Rappelons que le modèle LPV possède autant de paramètres variants que le réseau possède de neurones non-linéaires. Ce nombre se retrouve dans la dimension de la matrice Θ de la forme LFT. Il est important de garder cette équivalence à l'esprit puisqu'un surdimensionnement du réseau aboutirait à une représentation LFT de grande dimension qui peut se révéler très contraignante par la suite.

La représentation du SSNN (3.12) obtenue est maintenant proche des représentations standards utilisées en automatique puisqu'il s'agit d'un système LPV représenté par une forme LFT.

Néanmoins deux axes d'amélioration peuvent être identifiés :

- La représentation implique une perturbation unitaire constante en entrée du système qui est utile pour la modélisation des biais de chaque couche du réseau. Cependant la présence de ce signal d'entrée particulier encombre pour utiliser les méthodes d'analyses ou de synthèses linéaires traditionnelles puisque le modèle est ici affine. Cette difficulté sera levée par les représentations non-biaisées présentées par la suite dans la section 3.2.3.
- De par leurs définitions, les paramètres variants sont bornés dans l'intervalle $\theta_\ell^j \in [0; 1]$. Une première remarque est que cet intervalle est asymétrique par rapport à zéro. Les méthodes pouvant être employées se limitent alors à celles supposant ce type d'intervalle ou celles qui supposent un intervalle symétrique mais en considérant un intervalle artificiellement élargi. De plus, lors de l'analyse ou la synthèse LPV, la propriété bornée des paramètres variants est exploitée. Ainsi, dans ce cas, l'ensemble de l'intervalle des paramètres est envisagé ce qui aboutit à des résultats souffrant d'un grand conservatisme. En effet, l'interprétation des non-linéarités est réduite à la propriété de secteur borné des fonctions d'activation, le modèle à l'étude englobe donc une classe de système bien plus importante que le comportement réel du réseau. Afin d'exploiter au mieux la représentation LPV du réseau ainsi que sa forme LFT, il est alors inévitable de restreindre la largeur des systèmes considérés et d'affiner la modélisation. Une méthode réduisant le conservatisme de la représentation est présentée dans la section 3.2.4 dans la suite du rapport.

3.2.3 Formulation quasi-LPV non-biaisée

Comme nous avons pu le constater, la présence des biais dans les différentes couches du SSNN impose une représentation LPV-LFT qui comporte une entrée supplémentaire constante et unitaire [Suykens *et al.*, 1997]. Nous présentons à présent une méthode permettant d'effectuer un changement de coordonnées autour d'un point d'équilibre. Outre le fait de fournir un moyen de déplacer le point de fonctionnement du réseau vers l'origine, la finalité de l'approche réside dans l'élimination des biais afin de ne pas devoir les considérer comme une perturbation constante. La démarche présentée dans la proposition suivante, s'inspire des travaux de [Bendtsen et Trangbæk, 2000, Bendtsen et Trangbæk, 2002] qui traite un réseau comportant une seule couche cachée.

Proposition 3.3 : Formulation non-biaisée d'un SSNN

Soit un SSNN (3.1) et un point d'équilibre associé (x^0, u^0) , le SSNN admet une représentation non-biaisée telle que :

$$\tilde{H}^{\text{NN}} := \begin{cases} \tilde{h}_1(k) &= \tilde{\sigma}_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k)) \\ \tilde{h}_\ell(k) &= \tilde{\sigma}_\ell(W_\ell \tilde{h}_{\ell-1}) \quad \text{pour } \ell = 2, \dots, L \\ \tilde{x}(k+1) &= A\tilde{x}(k) + B\tilde{u}(k) + W_x \tilde{h}_L(k) \\ \tilde{y}(k) &= C\tilde{x}(k) + D\tilde{u}(k) \end{cases} \quad (3.15)$$

où $\tilde{x}(k) = x(k) - x^0$, $\tilde{u}(k) = u(k) - u^0$ et $\tilde{y}(k) = y(k) - y^0$ correspondent au changement de coordonnées autour du point d'équilibre qui est déterminé par la résolution des équations :

$$\begin{cases} h_1^0 &= \sigma_1(W_1^x x^0 + W_1^u u^0 + b_1) \\ h_\ell^0 &= \sigma_\ell(W_\ell h_{\ell-1}^0 + b_\ell) \quad \text{pour } \ell = 2, \dots, L \\ x^0 &= Ax^0 + Bu^0 + W_x h_L^0 \\ y^0 &= Cx^0 + Du^0 + b_y \end{cases} \quad (3.16)$$

Les nouvelles fonctions d'activation sont également définies par :

$$\begin{cases} \tilde{\sigma}_\ell(W_\ell \tilde{h}_{\ell-1}) = \sigma_\ell(W_\ell \tilde{h}_{\ell-1} + \xi_\ell^0) & \text{pour } \ell = 1, \dots, L-1 \\ \tilde{\sigma}_L(W_L \tilde{h}_{L-1}) = \sigma_L(W_L \tilde{h}_{L-1} + \xi_L^0) - \sigma_L(h_L^0) \end{cases} \quad (3.17)$$

avec $\xi_1^0 = W_1^x x^0 + W_1^u u^0 + b_1$ et $\xi_\ell^0 = b_\ell$ pour $\ell = 2, \dots, L$.

Démonstration. Dans un souci de clarté des notations, la démonstration est présentée pour un réseau comportant une seule couche cachée non-linéaire ($L = 1$). Cependant, la généralisation à L couches peut s'en déduire aisément. Supposons qu'il existe un point d'équilibre (x^0, u^0) tels que :

$$x^0 = Ax^0 + Bu^0 + W_x \sigma_1(W_1^x x^0 + W_1^u u^0 + b_1) + b_x \quad (3.18)$$

On pose :

$$\xi^0 = W_1^x x^0 + W_1^u u^0 + b_1 \quad (3.19)$$

Un changement de coordonnées du réseau peut alors être effectué selon $x(k) = \tilde{x}(k) + x^0$ et $u(k) = \tilde{u}(k) + u^0$ et la dynamique du réseau devient :

$$\begin{aligned} \tilde{x}(k+1) + x^0 &= A(\tilde{x}(k) + x^0) + B(\tilde{u}(k) + u^0) \\ &\quad + W_x \sigma_1(W_1^x (\tilde{x}(k) + x^0) + W_1^u (\tilde{u}(k) + u^0) + b_1) + b_x \\ &= A(\tilde{x}(k) + x^0) + B(\tilde{u}(k) + u^0) \\ &\quad + W_x \sigma_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k) + \xi^0) + b_x \end{aligned} \quad (3.20)$$

On définit une nouvelle fonction d'activation telle que :

$$\tilde{\sigma}_1(X) = \sigma_1(X + \xi^0) - \sigma_1(\xi^0) \quad (3.21)$$

En retirant et ajoutant $W_x \sigma_1(\xi^0)$ à la dynamique du réseau (3.20) on obtient :

$$\begin{aligned} \tilde{x}(k+1) + x^0 &= A(\tilde{x}(k) + x^0) + B(\tilde{u}(k) + u^0) + W_x \sigma_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k) + \xi^0) \\ &\quad + b_x - W_x \sigma_1(\xi^0) + W_x \sigma_1(\xi^0) \\ &= A\tilde{x}(k) + B\tilde{u}(k) + W_x \left(\sigma_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k) + \xi^0) - \sigma_1(\xi^0) \right) \\ &\quad + Ax^0 + Bu^0 + W_x \sigma_1(\xi^0) + b_x \\ \tilde{x}(k+1) &= A\tilde{x}(k) + B\tilde{u}(k) + W_x \tilde{\sigma}_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k)) \end{aligned} \quad (3.22)$$

□

Conformément à cette formulation sans biais du SSNN, sa représentation par un modèle LFT ne nécessitera pas la présence d'une perturbation constante unitaire. Un modèle LFT de la forme la figure 3.2 peut alors être déduit en suivant l'approche d'isolation des non-linéarités présentée dans les sections 3.2.1 et 3.2.2.

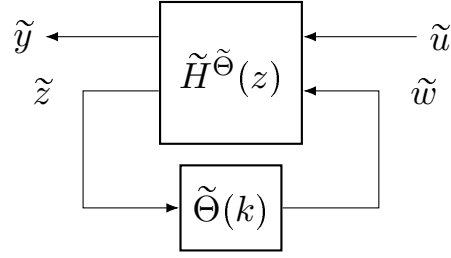


FIGURE 3.2 – Représentation LPV-LFT d'un SSNN non-biaisé isolant les non-linéarités

Proposition 3.4 : Représentation LPV-LFT d'un SSNN non-biaisé

Soit un SSNN non-biaisé (3.15), le SSNN admet une représentation LPV-LFT de la forme :

$$\left\{ \begin{array}{l} \begin{bmatrix} \tilde{x}(k+1) \\ \tilde{y}(k) \\ \tilde{z}_1(k) \\ \tilde{z}_2(k) \\ \vdots \\ \tilde{z}_L(k) \\ \tilde{w}_\ell(k) \end{bmatrix} = \begin{bmatrix} A & B & 0 & \cdots & 0 & W_x \\ C & D & 0 & \cdots & 0 & 0 \\ W_1^x & W_1^u & 0 & \cdots & 0 & 0 \\ 0 & 0 & W_2 & 0 & 0 & 0 \\ \vdots & \vdots & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & W_L & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ \tilde{u}(k) \\ \tilde{w}_1(k) \\ \vdots \\ \tilde{w}_{L-1}(k) \\ \tilde{w}_L(k) \end{bmatrix} \\ \tilde{w}_\ell(k) = \tilde{\Theta}_\ell(k) \tilde{z}_\ell(k) \end{array} \right. \quad (3.23)$$

où la matrice des paramètres variants associées aux nouvelles non-linéarités (3.17) de chaque couche cachée est définie telle que :

$$\left\{ \begin{array}{l} \tilde{\Theta}_\ell(k) = \text{diag}(\tilde{\theta}_\ell^j(k)) \\ \tilde{\theta}_\ell^j(k) = \frac{\tilde{\sigma}_\ell(\tilde{\xi}_\ell^j(k))}{\tilde{\xi}_\ell^j(k)} \end{array} \right. \quad (3.24)$$

avec

$$\left\{ \begin{array}{l} \tilde{\xi}_1 = W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k) \\ \tilde{\xi}_\ell = W_\ell \tilde{h}_{\ell-1}(k) \quad \text{pour } \ell = 2, \dots, L \end{array} \right. \quad (3.25)$$

Démonstration. Ce résultat est obtenu directement en reformulant le SSNN non-biaisé (3.15) selon un système quasi-LPV (section 3.2.1) puis en établissant une représentation LPV-LFT associée (section 3.2.1). \square

3.2.4 Formulation LPV-LFT normalisée

Cette partie aborde l'intervalle d'appartenance des paramètres variants et présente une méthode permettant d'ajuster cet intervalle dans la représentation LFT. La méthode est présentée à partir de la représentation LPV-LFT d'un SSNN non-biaisé (3.23) afin d'aboutir à une formulation sans perturbation constante. Néanmoins elle peut être appliquée de façon complètement analogue à la représentation LFT biaisée (3.12).

La matrice des paramètres variants $\tilde{\Theta}$ est une matrice diagonale dont les termes diagonaux $\tilde{\theta}_\ell^j$ sont les gains des fonctions d'activation non-linéaires. Selon la propriété de secteur borné des fonctions d'activation, chaque paramètre variant appartient à l'intervalle :

$$\tilde{\theta}_\ell^j(k) \in [\underline{\tilde{\theta}_\ell^j}; \overline{\tilde{\theta}_\ell^j}] \subset [0; 1] \quad (3.26)$$

L'idée est alors d'exploiter les valeurs limites de ces paramètres afin d'obtenir des paramètres variants normalisés dans l'intervalle $[-1; 1]$. Les paramètres variants $\tilde{\theta}_\ell^j$ sont alors considérés autour d'une valeur nominale définie comme le point milieu de l'intervalle $(\underline{\tilde{\theta}_\ell^j} + \overline{\tilde{\theta}_\ell^j})/2$ et dont les variations autour de cette valeur centrale peuvent aller jusqu'à $\pm(\underline{\tilde{\theta}_\ell^j} - \overline{\tilde{\theta}_\ell^j})/2$. Par le biais d'une fonction affine, définissons de nouveaux paramètres variants normalisés $\tilde{\delta}_\ell^j$ tels que :

$$\tilde{\theta}_\ell^j(k) = \frac{\underline{\tilde{\theta}_\ell^j} + \overline{\tilde{\theta}_\ell^j}}{2} + \frac{\tilde{\theta}_\ell^j - \overline{\tilde{\theta}_\ell^j}}{2} \tilde{\delta}_\ell^j(k) \iff \tilde{\delta}_\ell^j(k) = \frac{2}{\underline{\tilde{\theta}_\ell^j} - \overline{\tilde{\theta}_\ell^j}} \left(\tilde{\theta}_\ell^j(k) - \frac{\underline{\tilde{\theta}_\ell^j} + \overline{\tilde{\theta}_\ell^j}}{2} \right) \quad (3.27)$$

L'intervalle de variations des paramètres $\tilde{\theta}_\ell^j$ défini par (3.26) est conservé par la propriété d'appartenance :

$$\tilde{\delta}_\ell^j(k) \in [-1; 1] \quad (3.28)$$

De telles sortes que pour la matrice des paramètres normalisés de chaque couche non-linéaire, il suit que :

$$\tilde{\Theta}_\ell(k) = \underline{\tilde{\Theta}_\ell} + \tilde{\Theta}_\ell \tilde{\Delta}_\ell(k) \iff \tilde{\Delta}_\ell(k) = \underline{\tilde{\Theta}_\ell}^{-1} (\tilde{\Theta}_\ell(k) - \underline{\tilde{\Theta}_\ell}) \quad (3.29)$$

avec

$$\underline{\tilde{\Theta}_\ell} = \text{diag} \left(\frac{\underline{\tilde{\theta}_\ell^j} + \overline{\tilde{\theta}_\ell^j}}{2} \right) \text{ et } \tilde{\Theta}_\ell = \text{diag} \left(\frac{\tilde{\theta}_\ell^j - \overline{\tilde{\theta}_\ell^j}}{2} \right) \quad (3.30)$$

et

$$\|\tilde{\Delta}_\ell(k)\| \leq 1 \quad (3.31)$$

Concernant les intervalles des paramètres variants (3.26), le choix de bornes est fonction du degré de conservatisme qui sera prise en compte lors de l'analyse du modèle LPV-LFT. Le choix le plus conservatif est de fixer l'intervalle $\tilde{\theta}_\ell^j(k) \in [0; 1]$ comme l'illustre la figure 3.3a, puisque cela revient à ne considérer les non-linéarités qu'en termes de fonctions de secteur bornées c'est-à-dire que très peu de connaissances sur les non-linéarités réelles sont exploitées dans la pratique. Une autre approche est de tirer profit des données d'apprentissage afin de déterminer l'intervalle effectif de chaque neurone non-linéaire. À proprement parler, le réseau est reconnu comme une modélisation fiable essentiellement dans la région de l'espace d'état où il a été entraîné. Ainsi, à partir des séquences, des entrées et de l'état, qui peuvent être obtenues avec les données d'apprentissage, il est possible de déterminer l'intervalle d'appartenance de chaque paramètre variant selon :

$$\underline{\tilde{\theta}_\ell^j} = \inf_{1 \leq k \leq N_{\text{dt}}} (\tilde{\theta}_\ell^j(k)), \quad \overline{\tilde{\theta}_\ell^j} = \sup_{1 \leq k \leq N_{\text{dt}}} (\tilde{\theta}_\ell^j(k)) \quad (3.32)$$

La figure 3.3b illustre ainsi l'intervalle restreint d'appartenance des sorties d'un neurone non-linéaire tangente hyperbolique en fonction de ses entrées qui sont supposées comprises entre $\underline{\xi_\ell^j} = \inf(|\xi_\ell^j|)$ et $\overline{\xi_\ell^j} = \sup(|\xi_\ell^j|)$.

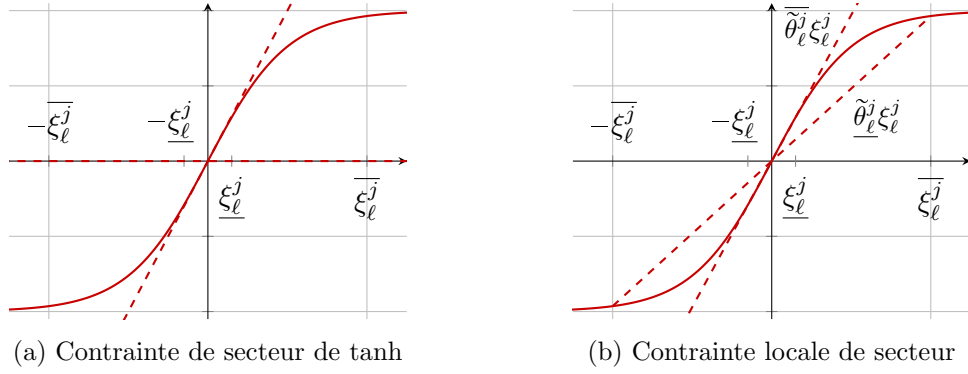


FIGURE 3.3 – Intervalle de la sortie d'un neurone non-linéaire tangente hyperbolique en fonction de son entrée

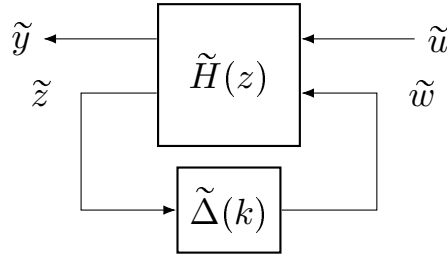


FIGURE 3.4 – Représentation LPV-LFT normalisée d'un SSNN non-biaisé

Finalement, à partir des intervalles des non-linéarités choisis puis des paramètres variants normalisés résultants, une représentation LFT normalisée (figure 3.4) peut être déduite.

Proposition 3.5 : Représentation LPV-LFT normalisée d'un SSNN non-biaisé

Soit un SSNN non-biaisé (3.15), le SSNN admet une représentation LPV-LFT normalisée de la forme :

$$\begin{aligned}
 \left[\begin{array}{c} \tilde{x}(k+1) \\ \tilde{y}(k) \\ \tilde{z}_1(k) \\ \tilde{z}_2(k) \\ \tilde{z}_3(k) \\ \vdots \\ \tilde{z}_L(k) \end{array} \right] &= \left[\begin{array}{c|c|c|c} \begin{array}{c} A + \\ W_x M_L W_1^x \end{array} & \begin{array}{c} B + \\ W_x M_L W_1^u \end{array} & \begin{array}{c} W_x M_1 \quad W_x M_2 \quad \cdots \quad W_x M_{L-1} \quad W_x M_L \end{array} & \\ \hline C & D & \begin{array}{c} 0 \quad 0 \quad \cdots \quad 0 \quad 0 \end{array} & \\ \hline W_1^x & W_1^u & \begin{array}{c} 0 \quad 0 \quad \cdots \quad 0 \quad 0 \end{array} & \\ \hline \overline{N}_2 W_1^x & \overline{N}_2 W_1^u & \begin{array}{c} W_2 \tilde{\Theta}_1 \quad 0 \quad 0 \quad 0 \quad 0 \end{array} & \\ \hline \overline{N}_3 W_1^x & \overline{N}_3 W_1^u & \begin{array}{c} \overline{N}_3 \quad W_3 \tilde{\Theta}_2 \quad 0 \quad 0 \quad 0 \end{array} & \\ \hline \vdots & \vdots & \begin{array}{c} \vdots \quad \ddots \quad \ddots \quad 0 \quad \vdots \end{array} & \\ \hline \overline{N}_L W_1^x & \overline{N}_L W_1^u & \begin{array}{c} \overline{N}_L \quad \cdots \quad \overline{N}_L \quad W_L \tilde{\Theta}_{L-1} \quad 0 \end{array} & \end{array} \right] \left[\begin{array}{c} \tilde{x}(k) \\ \tilde{u}(k) \\ \tilde{w}_1(k) \\ \tilde{w}_2(k) \\ \vdots \\ \tilde{w}_{L-1}(k) \\ \tilde{w}_L(k) \end{array} \right] \\
 \tilde{w}(k) &= \tilde{\Delta}(k) \tilde{z}(k), \quad \tilde{\Delta} = \text{diag}(\tilde{\Delta}_1, \dots, \tilde{\Delta}_L), \quad \|\tilde{\Delta}\| \leq 1
 \end{aligned} \tag{3.33}$$

avec

$$\begin{aligned}\overline{M}_\ell &= \prod_{i=\ell,\dots,2} \overline{\Theta}_i W_i \overline{\Theta}_1 & \underline{M}_\ell &= \prod_{i=L,\dots,\ell+1} \overline{\Theta}_i W_i \underline{\Theta}_\ell \\ \overline{N}_\ell &= \prod_{i=\ell,\dots,2} W_i \overline{\Theta}_{i-1} & \underline{N}_\ell &= W_\ell \prod_{i=\ell-1,\dots,2} \overline{\Theta}_i W_i \underline{\Theta}_1\end{aligned}\quad (3.34)$$

Démonstration. Ce résultat peut se déduire par récurrence. Considérons le SSNN non-biaisé (3.15) selon sa formulation quasi-LPV (similaire à l'équation (3.9)), la dynamique de la première couche est alors donnée par :

$$\tilde{h}_1(k) = \tilde{\Theta}_1(W_1^x \tilde{x}(k) + W_1^u \tilde{u}(k)) = \tilde{\Theta}_1 \xi_1(k) \quad (3.35)$$

En considérant l'expression des paramètres variants selon (3.29), il suit que :

$$\tilde{h}_1(k) = (\overline{\Theta}_1 + \underline{\Theta}_1 \tilde{\Delta}_1) \xi_1(k) = \overline{\Theta}_1 \tilde{\xi}_1(k) + \underline{\Theta}_1 \tilde{w}_1(k) \quad (3.36)$$

avec $\tilde{w}_1(k) = \tilde{\Delta}_1 \tilde{z}_1(k)$ et $\tilde{z}_1(k) = \tilde{\xi}_1(k)$. En réitérant la démarche pour la deuxième couche et en posant $\underline{M}_\ell^j = \prod_{i=j,\dots,\ell+1} \overline{\Theta}_i W_i \underline{\Theta}_\ell$, celle-ci s'exprime alors :

$$\begin{aligned}\tilde{h}_2(k) &= \tilde{\Theta}_2 W_2 \tilde{h}_1(k) = (\overline{\Theta}_2 + \underline{\Theta}_2 \tilde{\Delta}_2) (W_2 \overline{\Theta}_1 \tilde{\xi}_1(k) + W_2 \underline{\Theta}_1 \tilde{w}_1(k)) \\ &= \overline{\Theta}_2 W_2 \overline{\Theta}_1 \tilde{\xi}_1(k) + \overline{\Theta}_2 W_2 \underline{\Theta}_1 \tilde{w}_1(k) + \underline{\Theta}_2 \tilde{w}_2(k) \\ &= \overline{M}_2 \tilde{\xi}_1 + \underline{M}_1^2 \tilde{w}_1(k) + \underline{M}_2^2 \tilde{w}_2(k)\end{aligned}\quad (3.37)$$

avec $\tilde{w}_2(k) = \tilde{\Delta}_2 \tilde{z}_2(k)$ et $\tilde{z}_2(k) = W_2 \overline{\Theta}_1 \tilde{\xi}_1(k) + W_2 \underline{\Theta}_1 \tilde{w}_1(k) = \overline{N}_2 \tilde{\xi}_1 + W_2 \underline{\Theta}_1 \tilde{w}_1(k)$. Finalement, considérons la couche $L - 1$ pour laquelle :

$$\begin{aligned}\tilde{h}_{L-1}(k) &= \overline{M}_{L-1} \tilde{\xi}_1 + \sum_{j=1}^{L-1} \underline{M}_j^{L-1} \tilde{w}_j(k) \\ \tilde{z}_{L-1}(k) &= \overline{N}_{L-1} \tilde{\xi}_1 + \sum_{j=1}^{L-2} \underline{N}_j \tilde{w}_j(k) + W_{L-1} \underline{\Theta}_{L-2} \tilde{w}_{L-2}(k)\end{aligned}\quad (3.38)$$

La couche L est alors donnée par :

$$\begin{aligned}\tilde{h}_L(k) &= \tilde{\Theta}_L W_L \tilde{h}_{L-1}(k) = (\overline{\Theta}_L + \underline{\Theta}_L \tilde{\Delta}_L) (W_L \overline{M}_{L-1} \tilde{\xi}_1 + W_L \sum_{j=1}^{L-1} \underline{M}_j^{L-1} \tilde{w}_j(k)) \\ &= \overline{\Theta}_L W_L \overline{M}_{L-1} \tilde{\xi}_1 + \overline{\Theta}_L W_L \sum_{j=1}^{L-1} \underline{M}_j^{L-1} \tilde{w}_j(k) + \underline{\Theta}_L \tilde{w}_L \\ &= \overline{M}_L \tilde{\xi}_1 + \sum_{j=1}^L \underline{M}_j^L \tilde{w}_j(k)\end{aligned}\quad (3.39)$$

avec $\tilde{w}_L(k) = \tilde{\Delta}_L \tilde{z}_L(k)$ et après quelques calculs, il advient également :

$$\begin{aligned}\tilde{z}_L &= W_L \overline{M}_{L-1} \tilde{\xi}_1 + W_L \sum_{j=1}^{L-1} \underline{M}_j^{L-1} \tilde{w}_j(k) \\ &= \overline{N}_L \tilde{\xi}_1 + \sum_{j=1}^{L-2} \underline{N}_j \tilde{w}_j(k) + W_L \underline{\Theta}_{L-1} \tilde{w}_{L-1}(k)\end{aligned}\quad (3.40)$$

□

Lorsque le réseau SSNN non-biaisé (3.15) ne possède qu'une seule couche cachée non-linéaire

($L = 1$) alors la représentation LPV-LFT normalisée se simplifie telle que :

$$\left\{ \begin{array}{l} \left[\begin{array}{c} \tilde{x}(k+1) \\ \tilde{y}(k) \\ \tilde{z}_1(k) \end{array} \right] = \left[\begin{array}{c|c|c} A + W_x \tilde{\Theta}_1 W_1^x & B + W_x \tilde{\Theta}_1 W_1^u & W_x \tilde{\Theta}_1 \\ \hline C & D & 0 \\ \hline W_1^x & W_1^u & 0 \end{array} \right] \left[\begin{array}{c} \tilde{x}(k) \\ \tilde{u}(k) \\ \tilde{w}_1(k) \end{array} \right] \\ \tilde{w}_1(k) = \tilde{\Delta}_1(k) \tilde{z}_1(k), \quad \tilde{\Delta}_1(k) = \tilde{\Theta}_1^{-1} (\tilde{\Theta}_1(k) - \tilde{\Theta}_1) \in [-1; 1] \end{array} \right. \quad (3.41)$$

SECTION 3.3

Modélisation des associations de SSNN

Nous nous intéressons dans cette section à la modélisation de combinaisons de plusieurs réseaux de type SSNN. Comme nous l'établirons par la suite, plusieurs types d'interconnexions de SSNN peuvent être exprimés selon un seul modèle SSNN équivalent. Ainsi, il est souvent avantageux de manipuler un SSNN global plutôt que plusieurs sous-systèmes. Le SSNN global peut par la suite être analysé au besoin par sa formulation LPV-LFT comme nous le présenterons dans la section 3.4 suivante.

Remarque 3.3. Une propriété fondamentale des formes LFT est qu'une interconnexion linéaire de plusieurs LFT peut toujours être reformulée selon une seule LFT équivalente. Néanmoins, il est préférable de déterminer la représentation LFT d'un seul modèle SSNN plutôt que de réunir plusieurs LFT en une forme équivalente. En effet, lors de la détermination de la formulation non-biaisée des SSNN (voir la proposition 3.3), le changement de coordonnées autour du point d'équilibre (x^0, u^0) impose également au réseau un changement des entrées et des sorties telles que $\tilde{u}(k) = u(k) - u^0$ et $\tilde{y}(k) = y(k) - y^0$. Ainsi, lors de l'interconnexion de plusieurs LFT, les signaux d'entrées-sorties des modèles respectifs subissent une perturbation constante qui doit être modélisée à l'aide d'une méthode annexe.

3.3.1 La mise en série de SSNN est un SSNN

Soient deux réseaux SSNN dont le nombre de couches cachées non-linéaires sont respectivement L_1 et L_2 , la connexion série (figure 3.5) obtenue en connectant la sortie du premier réseau à l'entrée du second réseau est un SSNN dont le nombre de couches cachées non-linéaires L est égale à celle du réseau le plus profond, soit $L = \max\{L_1, L_2\}$.

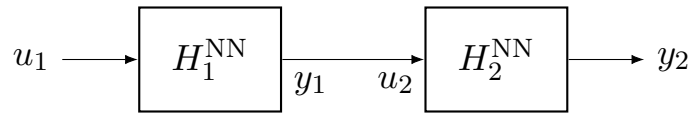


FIGURE 3.5 – Connexion en série de deux SSNN

En effet, de façon plus explicite, pour deux réseaux H_1^{NN} et H_2^{NN} comportant une seule couche cachée non-linéaire et exprimés de la manière suivante :

$$H_1^{NN} : \begin{cases} h_1(k) &= \sigma_{h_1}(W_1^x x_1(k) + W_1^u u_1(k) + b_1) \\ x_1(k+1) &= A_1 x_1(k) + B_1 u_1(k) + W_{x_1} h_1(k) + b_{x_1} \\ y_1(k) &= C_1 x_1(k) + D_1 u_1(k) + b_{y_1} \end{cases} \quad (3.42)$$

$$H_2^{\text{NN}} : \begin{cases} h_2(k) &= \sigma_2(W_2^x x_2(k) + W_2^u u_2(k) + b_2) \\ x_2(k+1) &= A_2 x_2(k) + B_2 u_2(k) + W_{x_2} h_2(k) + b_{x_2} \\ y_2(k) &= C_2 x_2(k) + D_2 u_2(k) + b_{y_2} \end{cases} \quad (3.43)$$

Le système de mise en série des réseaux conformément à la figure 3.5 est décrit selon un SSNN équivalent à une couche cachée non-linéaire de la forme :

$$\begin{cases} \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \left(\begin{bmatrix} W_1^x & 0 \\ W_2^u C_1 & W_2^x \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} W_1^u \\ W_2^u D_1 \end{bmatrix} u_1(k) + \begin{bmatrix} b_1 \\ b_2 + W_2^u b_{y_1} \end{bmatrix} \right) \\ \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ B_2 C_1 & A_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 D_1 \end{bmatrix} u_1(k) + \begin{bmatrix} W_{x_1} & 0 \\ 0 & W_{x_2} \end{bmatrix} \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} + \begin{bmatrix} b_{x_1} \\ b_{x_2} + B_2 b_{y_1} \end{bmatrix} \\ y_2(k) = \begin{bmatrix} D_2 C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} D_2 D_1 \end{bmatrix} u_1(k) + \begin{bmatrix} b_{y_2} + D_2 b_{y_1} \end{bmatrix} \end{cases} \quad (3.44)$$

Remarque 3.4. La généralisation à plusieurs couches cachées s'obtient aisément puisque seules la première couche et les deux dernières couches (couche cachée linéaire et couche de sortie linéaire) sont impliquées dans l'interconnexion. Ainsi, pour des couches non-linéaires définies pour le réseau H_i^{NN} , $i = \{1, 2\}$ selon :

$$h_{[i, \ell_i]}(k) = \sigma_{[i, \ell_i]}(W_{[i, \ell_i]} h_{[i, \ell_i - 1]}(k) + b_{[i, \ell_i]}) \quad \text{pour } \ell_i = 2, \dots, L_i \quad (3.45)$$

alors les couches cachées additionnelles du système d'interconnexion équivalent sont obtenues par la concaténation suivante :

$$\begin{bmatrix} h_{[1, \ell]}(k) \\ h_{[2, \ell]}(k) \end{bmatrix} = \begin{bmatrix} \sigma_{[1, \ell]} \\ \sigma_{[2, \ell]} \end{bmatrix} \left(\begin{bmatrix} W_{[1, \ell]} & 0 \\ 0 & W_{[2, \ell]} \end{bmatrix} \begin{bmatrix} h_{[1, \ell - 1]}(k) \\ h_{[2, \ell - 1]}(k) \end{bmatrix} + \begin{bmatrix} b_{[1, \ell]} \\ b_{[2, \ell]} \end{bmatrix} \right) \quad \text{pour } \ell = 2, \dots, \max\{L_1, L_2\} \quad (3.46)$$

Notons que lors de l'interconnexion de deux SSNN, le réseau le plus petit est fictivement agrandi avec des couches vides prenant la forme mathématique de matrices et de fonctions d'activation de dimensions nulles afin que les deux réseaux soient de même profondeur $L_1 = L_2 = L$.

3.3.2 Prise en compte des retards dans la forme SSNN finale

Comme nous l'avons vu dans le chapitre 1, lors de l'apprentissage, l'entrée du réseau de neurones $u(k)$ n'est généralement pas directement injectée au réseau mais elle est augmentée par le biais d'un système de retard TDL (voir la définition 1.1). L'entrée effective consiste en la concaténation du signal d'entrée avec une fenêtre de ses valeurs antérieures c'est-à-dire que pour la TDL($n_{in}^0 : n_{in}$) l'entrée du réseau est $\bar{u}(k) = [u^T(k - n_{in}^0) \cdots u^T(k - n_{in})]^T$. Néanmoins lors de l'analyse ou l'interconnexion de SSNN il est souvent utile d'étudier l'entrée d'origine u et non le vecteur de retards successifs \bar{u} . L'objectif est alors d'inclure le système TDL dans la description de la dynamique du SSNN.

Tout d'abord il est possible d'exprimer les systèmes TDL à l'aide de représentations d'état, deux réalisations peuvent alors être distinguées selon que le vecteur de retards successifs comporte le signal à l'instant présent ou qu'il impose au minimum un retard.

Définition 3.1 : Système TDL

Soit la TDL($n_1 : n_2$) qui est la fonction de concaténation du signal $s(k) \in \mathbb{R}^{n_s}$ en un vecteur de signaux retardés $\bar{s}(k) = [s^T(k-n_1) \cdots s^T(k-n_2)]^T \in \mathbb{R}^{m_s}$ de dimension $m_s = n_s(n_2 - n_1 + 1)$, le système de retards d'état $x_{tdl} \in \mathbb{R}^{n_s n_2}$ s'exprime de la manière suivante :

- si $n_1 = 0$, la dynamique du système de retards est donnée par :

$$\begin{cases} x_{tdl}(k+1) = \begin{bmatrix} 0_{n_s \times n_s(n_2-1)} & 0_{n_s \times n_s} \\ I_{n_s(n_2-1)} & 0_{n_s(n_2-1) \times n_s} \end{bmatrix} x_{tdl}(k) + \begin{bmatrix} I_{n_s} \\ 0_{n_s(n_2-1) \times n_s} \end{bmatrix} s(k) \\ \bar{s}(k) = \begin{bmatrix} 0_{n_s \times n_s n_2} \\ I_{n_s n_2} \end{bmatrix} x_{tdl}(k) + \begin{bmatrix} I_{n_s} \\ 0_{n_s n_2 \times n_s} \end{bmatrix} s(k) \end{cases} \quad (3.47)$$

- si $n_1 > 0$, la dynamique s'obtient par retrait du transfert direct, soit :

$$\begin{cases} x_{tdl}(k+1) = \begin{bmatrix} 0_{n_s \times n_s(n_2-1)} & 0_{n_s \times n_s} \\ I_{n_s(n_2-1)} & 0_{n_s(n_2-1) \times n_s} \end{bmatrix} x_{tdl}(k) + \begin{bmatrix} I_{n_s} \\ 0_{n_s(n_2-1) \times n_s} \end{bmatrix} s(k) \\ \bar{s}(k) = \begin{bmatrix} 0_{m_s \times n_s(n_1-1)} & I_{m_s} \end{bmatrix} x_{tdl}(k) + \begin{bmatrix} 0_{m_s \times n_s} \end{bmatrix} s(k) \end{cases} \quad (3.48)$$

Finalement pour un SSNN à L couches cachées non-linéaires défini par (3.1), la TDL du signal d'entrée de réalisation ($A_{tdl}, B_{tdl}, C_{tdl}, D_{tdl}$) est également un SSNN particulier (qui ne comporte aucune couche cachée et dont les biais sont nuls). À partir de la mise en série de deux SSNN (3.44), une forme SSNN global incluant le système de retards successifs est obtenue selon :

$$\begin{cases} h_1(k) = \sigma_{h_1} \left([W_h^u C_{tdl} \ W_h^x] \begin{bmatrix} x_{tdl}(k) \\ x(k) \end{bmatrix} + [W_h^u D_{tdl}] u(k) + b_h \right) \\ h_\ell(k) = \sigma_\ell (W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell=2, \dots, L \\ \begin{bmatrix} x_{tdl}(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} A_{tdl} & 0 \\ BC_{tdl} & A \end{bmatrix} \begin{bmatrix} x_{tdl}(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} B_{tdl} \\ BD_{tdl} \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ W_x \end{bmatrix} h_L(k) + \begin{bmatrix} 0 \\ b_x \end{bmatrix} \\ y(k) = [DC_{tdl} \ C] \begin{bmatrix} x_{tdl}(k) \\ x(k) \end{bmatrix} + [DD_{tdl}] u(k) + b_y \end{cases} \quad (3.49)$$

3.3.3 L'interconnexion de SSNN est un SSNN

Soient deux réseaux SSNN définis par (3.1) comportant respectivement L_1 et L_2 couches cachées non-linéaires, le système d'interconnexion des réseaux par rétroaction de la figure 3.6 est également un SSNN à L couches cachées non-linéaires dont la profondeur est égale à celle du réseau le plus profond, $L = \max\{L_1, L_2\}$.

Explicitement, pour deux réseaux H_1^{NN} et H_2^{NN} comportant une seule couche cachée non-linéaire et définis de la façon suivante :

$$H_1^{\text{NN}} : \begin{cases} h_1(k) = \sigma_1 (W_1^x x_1(k) + W_1^p w_p(k) + W_1^u u_1(k) + b_1) \\ x_1(k+1) = A_1 x_1(k) + B_p w_p(k) + B_1 u_1(k) + W_{x_1} h_1(k) + b_{x_1} \\ z_p(k) = C_p x_1(k) + D_{pp} w_p(k) + D_{pu} u_1(k) + b_p \\ y_1(k) = C_1 x_1(k) + D_p w_p(k) + D_1 u_1(k) + b_{y_1} \end{cases} \quad (3.50)$$

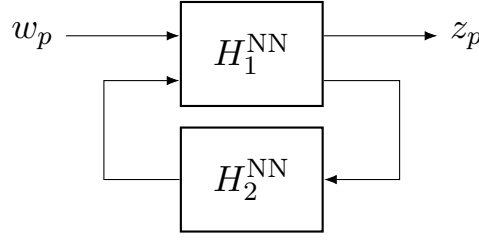


FIGURE 3.6 – Système en boucle fermée composé de deux SSNN

$$H_2^{\text{NN}} : \begin{cases} h_2(k) &= \sigma_2(W_2^x x_2(k) + W_2^u u_2(k) + b_2) \\ x_2(k+1) &= A_2 x_2(k) + B_2 u_2(k) + W_{x_2} h_2(k) + b_{x_2} \\ y_2(k) &= C_2 x_2(k) + D_2 u_2(k) + b_{y_2} \end{cases} \quad (3.51)$$

Le système en boucle fermée de la figure 3.6 se décrit selon SSNN équivalent à une couche cachée non-linéaires de la forme :

$$\left\{ \begin{array}{l} \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \left(\begin{bmatrix} W_1^x + W_1^u D_2 E C_1 & W_1^u (C_2 + D_2 E D_1 C_2) \\ W_2^u E C_1 & W_2^x + W_2^u E D_1 C_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \right. \\ \quad \left. + \begin{bmatrix} W_1^p + W_1^u D_2 E D_p \\ W_2^u E D_p \end{bmatrix} w_p(k) + \begin{bmatrix} b_1 + W_1^u D_2 E (b_{y_1} + D_1 b_{y_2}) \\ b_2 + W_2^u E (b_{y_1} + D_1 b_{y_2}) \end{bmatrix} \right) \\ \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} A_1 + B_1 D_2 E C_1 & B_1 (C_2 + D_2 E D_1 C_2) \\ B_2 E C_1 & A_2 + B_2 E D_1 C_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} B_p + B_1 D_2 E D_p \\ B_2 E D_p \end{bmatrix} w_p(k) \\ \quad + \begin{bmatrix} W_{x_1} & 0 \\ 0 & W_{x_2} \end{bmatrix} \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} + \begin{bmatrix} b_{x_1} + B_1 D_2 E (b_{y_1} + D_1 b_{y_2}) \\ b_{x_2} + B_2 E (b_{y_1} + D_1 b_{y_2}) \end{bmatrix} \\ z_p(k) = \begin{bmatrix} C_p + D_{pu} D_2 E C_1 & D_{pu} (C_2 + D_2 E D_1 C_2) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} D_{pp} + D_{pu} D_2 E D_p \end{bmatrix} w_p(k) \\ \quad + \begin{bmatrix} b_p + D_{pu} D_2 E (b_{y_1} + D_1 b_{y_2}) \end{bmatrix} \end{array} \right. \quad (3.52)$$

avec $E = (I - D_1 D_2)^{-1}$. La formulation étendue à plusieurs couches cachées s'obtient également à partir de la remarque 3.4.

SECTION 3.4

Analyse des SSNN via la formulation LPV-LFT

Nous développons dans cette section l'analyse de robustesse en stabilité et en performance d'un modèle H^{NN} de type SSNN dont la représentation d'état respecte l'équation (3.1). Au vu des résultats présentés dans les sections précédentes, il est supposé que le réseau à analyser admette une représentation LPV-LFT normalisée (et non-biaisée) équivalente $F_u(\tilde{H}(z), \tilde{\Delta})$ dont le comportement dynamique est donné par :

$$F_u(\tilde{H}(z), \tilde{\Delta}) = \begin{cases} \begin{bmatrix} \tilde{x}(k+1) \\ \tilde{z}(k) \\ \tilde{y}(k) \end{bmatrix} = \begin{bmatrix} \tilde{A} & \tilde{B}_\Delta & \tilde{B}_p \\ \tilde{C}_\Delta & \tilde{D}_{\Delta\Delta} & \tilde{D}_{\Delta p} \\ \tilde{C}_p & \tilde{D}_{p\Delta} & \tilde{D}_{pp} \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ \tilde{w}(k) \\ \tilde{u}(k) \end{bmatrix} \\ \tilde{w}(k) = \tilde{\Delta}(k) \tilde{z}(k), \quad \|\tilde{\Delta}(k)\| \leq 1 \end{cases} \quad (3.53)$$

3.4.1 Analyse de stabilité et de performance des SSNN

L'objectif de cette partie est d'exploiter les résultats introduits dans le chapitre 2 d'analyse de systèmes LPV-LFT. Les SSNN sont étudiés selon leurs représentations LPV-LFT qui permettent d'isoler les non-linéarités au moyen de paramètres variants. Ainsi, il est possible d'établir des conditions de stabilité pour toute trajectoire de paramètres ce qui permet également de garantir en particulier la trajectoire des paramètres variants propre aux non-linéarités au regard d'un certain conservatisme. De plus, l'analyse des SSNN peut être résolue numériquement à travers un problème d'optimisation sous contraintes LMI.

Théorème 3.1 : Stabilité d'un SSNN

Soit \tilde{H}^{NN} un SSNN et $F_u(\tilde{H}(z), \tilde{\Delta})$ sa représentation LPV-LFT avec $\|\tilde{\Delta}\|_{\mathcal{L}_2} \leq 1$, le réseau \tilde{H}^{NN} est stable si il existe $L \in L_{\underline{\Delta}}$ telle que la norme \mathcal{H}_{∞} vérifie

$$\|L^{1/2}\tilde{H}L^{-1/2}\|_{\infty} = \gamma < 1 \quad (3.54)$$

Démonstration. Cette condition de stabilité est directement déduit du théorème du petit gain avec multiplicateurs (corollaire 2.3). Si la condition est vérifiée, alors la stabilité est obtenue pour tout $\tilde{\Delta}(\tilde{\Theta}(k)) \in \underline{\Delta}$ avec $\|\tilde{\Delta}\|_{\mathcal{L}_2} \leq 1$ et pour toute trajectoire de $\tilde{\Theta}(k)$. Elle est donc également obtenue pour la trajectoire particulière des paramètres variants correspondant aux non-linéarités. \square

Théorème 3.2 : Stabilité quadratique d'un SSNN

Soit \tilde{H}^{NN} un SSNN et $F_u(\tilde{H}(z), \tilde{\Delta})$ sa représentation LPV-LFT (3.53), le réseau \tilde{H}^{NN} est asymptotiquement stable si il existe une matrice symétrique définie positive X et $L \in L_{\underline{\Delta}}$ solution de l'inégalité :

$$\begin{bmatrix} \tilde{A}^T X \tilde{A} - X & \tilde{A}^T X \tilde{B}_{\Delta} & \tilde{C}_{\Delta}^T L \\ \tilde{B}_{\Delta}^T X \tilde{A} & \tilde{B}_{\Delta}^T X \tilde{B}_{\Delta} - L & \tilde{D}_{\Delta}^T L \\ L \tilde{C}_{\Delta} & L \tilde{D}_{\Delta} & -L \end{bmatrix} < 0 \quad (3.55)$$

Démonstration. Ce théorème se déduit directement du corollaire 2.4 d'après le lemme bornée réel avec multiplicateurs. \square

Théorème 3.3 : Performance d'un SSNN

Soit \tilde{H}^{NN} un SSNN et $F_u(\tilde{H}(z), \tilde{\Delta})$ sa représentation LPV-LFT avec $\|\tilde{\Delta}\|_{\mathcal{L}_2} \leq 1$, la norme \mathcal{L}_2 du réseau \tilde{H}^{NN} vérifie

$$\|\tilde{H}^{\text{NN}}\|_{\mathcal{L}_2} < \gamma \iff \|F_u(\tilde{H}(z), \tilde{\Delta})\|_{\mathcal{L}_2} < \gamma \quad (3.56)$$

si il existe une matrice $L \in L_{\underline{\Delta}}$ telle que :

$$\left\| \begin{bmatrix} L & 0 \\ 0 & \frac{I}{\sqrt{\gamma}} \end{bmatrix} \tilde{H} \begin{bmatrix} L^{-1} & 0 \\ 0 & \frac{I}{\sqrt{\gamma}} \end{bmatrix} \right\|_{\infty} < 1 \quad (3.57)$$

Démonstration. Ce théorème se déduit du corollaire 2.5 d'après le théorème du petit gain pour l'analyse de performance. Une fois encore, la stabilité est obtenue pour tout $\tilde{\Delta}(\tilde{\Theta}(k)) \in \underline{\Delta}$ avec $\|\tilde{\Delta}\|_{\mathcal{L}_2} \leq 1$ et pour toute trajectoire de $\tilde{\Theta}(k)$, elle est donc également obtenue pour la trajectoire particulière des paramètres correspondant aux non-linéarités. \square

Théorème 3.4 : Performance quadratique d'un SSNN

Soit \tilde{H}^{NN} un SSNN et $F_u(\tilde{H}(z), \tilde{\Delta})$ sa représentation LPV-LFT (3.53), le réseau \tilde{H}^{NN} est asymptotiquement stable et vérifie

$$\|\tilde{H}^{\text{NN}}\|_{\mathcal{L}_2} < \gamma \quad (3.58)$$

si il existe une matrice symétrique définie positive X et $L \in L_{\underline{\Delta}}$ solution de l'inégalité :

$$\begin{bmatrix} \tilde{A}^T X \tilde{A} - X & \tilde{A}^T X \tilde{B}_{\Delta} & \tilde{A}^T X \tilde{B}_p & \tilde{C}_{\Delta}^T L & \tilde{C}_p^T \\ \tilde{B}_{\Delta}^T X \tilde{A} & \tilde{B}_{\Delta}^T X \tilde{B}_{\Delta} - L & \tilde{B}_{\Delta}^T X \tilde{B}_p & \tilde{D}_{\Delta\Delta}^T L & \tilde{D}_{p\Delta} \\ \tilde{B}_p^T X \tilde{A} & \tilde{B}_p^T X \tilde{B}_{\Delta} & \tilde{B}_p^T X \tilde{B}_p - \gamma I & \tilde{D}_{\Delta p}^T L & \tilde{D}_{pp} \\ L \tilde{C}_{\Delta} & L \tilde{D}_{\Delta\Delta} & L \tilde{D}_{\Delta p} & -L & 0 \\ \tilde{C}_p & \tilde{D}_{p\Delta} & \tilde{D}_{pp} & 0 & -\gamma I \end{bmatrix} < 0 \quad (3.59)$$

Démonstration. Ce théorème se déduit du corollaire 2.6 d'après le lemme borné réel avec multiplicateurs. \square

3.4.2 Analyse des SSNN avec filtrage des paramètres

Un inconvénient majeur des outils d'analyse de SSNN présentés jusqu'à présent est qu'aucune limite sur les vitesses de variations des paramètres n'est considérée. Par conséquent, les propriétés de stabilité et de performance sont garanties y compris pour des évolutions des paramètres variant extrêmement rapides. Ceci s'explique par les fondements de la méthode qui exploite des fonctions de Lyapunov quadratique. Une possibilité est alors d'établir les résultats à partir de fonctions de Lyapunov dépendant des paramètres, néanmoins cela conduit souvent à des algorithmes plus complexes et coûteux en temps de calcul.

Une alternative est alors d'employer la solution proposée dans les travaux de [Biannic, 2010] qui consiste à filtrer les variations des paramètres les plus rapides à l'aide d'un filtre passe-bas. Comme l'illustre la figure 3.7, un filtre $F(z)$ strictement propre et de gain statique unitaire est introduit en sortie de l'opérateur regroupant les paramètres variants. Ce filtre prend une forme diagonale de plusieurs fonctions passe-bas dont les bandes passantes sont choisies largement supérieur à celle du système. Néanmoins, une approche judicieuse doit être suivie afin d'atténuer significativement les vitesses de variations rapides et introduire une borne peu conservatrice, sans pour autant affecter la validée du modèle.

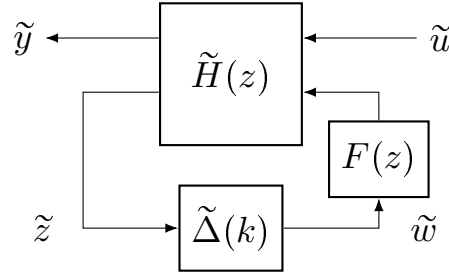


FIGURE 3.7 – Représentation LPV-LFT d'un SSNN avec filtrage des paramètres

Pour l'analyse des SSNN, cette méthode permet finalement de borner les vitesses de variations des paramètres et de supprimer des contraintes très exigeantes de stabilité sur la trajectoire des paramètres (par exemple des trajectoires non-différentiables ou discontinues). Les approches d'analyses de stabilité et de performance des SSNN (section 3.4.1) peuvent être directement appliquées au système incluant les filtres. En notant la représentation d'état du filtre (A_F, B_F, C_F) et x_F le vecteur d'état, la représentation LPV-LFT du système (3.53) incluant les filtres est définie telle que :

$$\left\{ \begin{array}{l} \begin{bmatrix} \tilde{x}(k+1) \\ x_F(k+1) \\ \tilde{z}(k) \\ \tilde{y}(k) \end{bmatrix} = \begin{bmatrix} \tilde{A} & \tilde{B}_\Delta C_F & 0 & \tilde{B}_p \\ 0 & A_F & B_F & 0 \\ \tilde{C}_\Delta & \tilde{D}_{\Delta\Delta} C_F & 0 & \tilde{D}_{\Delta p} \\ \tilde{C}_p & \tilde{D}_{p\Delta} C_F & 0 & \tilde{D}_{pp} \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ x_F(k) \\ \tilde{w}(k) \\ \tilde{u}(k) \end{bmatrix} \\ \tilde{w}(k) = \tilde{\Delta}(k)\tilde{z}(k), \quad \|\tilde{\Delta}(k)\| \leq 1 \end{array} \right. \quad (3.60)$$

Remarque 3.5. À l'image des bornes des paramètres variants, les vitesses de ces mêmes paramètres peuvent être également appréciées avec les données d'apprentissages. La détermination des filtres peut alors être guidée en évaluant son impact sur les signaux des paramètres variants qui sont issus des séquences d'apprentissage.

3.4.3 Analyse des SSNN par μ -analyse

Malgré les techniques mises en œuvre pour réduire le conservatisme des résultats d'analyse de SSNN, leur résolution numérique représente également un réel challenge. En effet, les problèmes LMI impliqués présentent une complexité numérique importante du fait du nombre important de paramètres variants (dont le nombre est directement relié au nombre de neurones non-linéaire que comporte le réseau). La résolution numérique devient alors rapidement délicate

lorsque les LMI sont issues de structures neuronales de trop grande dimension. Au vu de cette difficulté, nous proposons alors d'étudier la stabilité et les performances des SSNN à l'aide de la μ -analyse [Packard et Doyle, 1993] sous certaines limitations qui nous évoquerons par la suite.

La μ -analyse est un outil pour effectuer des analyses de robustesse en s'appuyant sur la valeur singulière structurée. Cette dernière permet de tenir compte de la structure des incertitudes du modèle dans l'objectif d'apporter des résultats d'analyse moins pessimistes que ceux du théorème du petit gain (voir le corollaire 2.1). Le problème à l'étude est, comme pour ce dernier théorème, le système d'interconnexion standard de deux sous-systèmes qui a été présenté par la figure 2.3. Nous définissons à présent, l'ensemble des matrices complexes structurées dans le cas général ainsi que la valeur singulière relative à cet ensemble.

Définition 3.2 : Matrices complexes structurées

L'ensemble $\underline{\Delta}^*$ des matrices complexes de structure diagonale en bloc est défini tel que :

$$\underline{\Delta}^* := \left\{ \begin{array}{l} \Delta = \text{diag}\{\Delta_1, \dots, \Delta_q, \delta_1 I_{r_1}, \dots, \delta_r I_{r_r}, \varepsilon_1 I_{c_1}, \dots, \varepsilon_c I_{c_c}\} \\ \Delta_i \in \mathbb{C}^{k_i \times k_i} \quad ; \quad \delta_i \in \mathbb{R} \quad ; \quad \varepsilon_i \in \mathbb{C} \end{array} \right\} \quad (3.61)$$

Définition 3.3 : Valeur singulière structurée

Soit la matrice de transfert M , la valeur singulière structurée du système c'est-à-dire relative à l'ensemble $\underline{\Delta}^*$ est définie comme :

$$\mu_{\underline{\Delta}}(M) := \left(\inf_{\Delta \in \underline{\Delta}^*} (\bar{\sigma}(\Delta) : \det(I - \Delta M) = 0) \right)^{-1} \quad (3.62)$$

Si $(I - \Delta M)$ n'est pas singulière pour tout $\Delta \in \underline{\Delta}^*$, alors $\mu_{\underline{\Delta}}(M) := 0$.

La définition de $\mu_{\underline{\Delta}}(M)$ conduit au théorème du petit gain généralisé. Ce théorème est le cœur de l'analyse de robustesse par μ -analyse qui se fonde sur la valeur singulière structurée.

Théorème 3.5 : Théorème du petit gain généralisé [Doyle, 1985, Zhou et al., 1996]

Soient $M(z) \in \mathcal{RH}_\infty$ et $\Delta \in \mathcal{RH}_\infty$, l'interconnexion de la figure 2.3 est stable pour toutes incertitudes structurées $\Delta \in \underline{\Delta}^*$ telles que $\|\Delta\|_\infty < \gamma$ si et seulement si

$$\forall \omega \quad \mu_{\underline{\Delta}}(M(e^{j\omega})) \leq \gamma^{-1} \quad (3.63)$$

Sur la base du théorème du petit gain généralisé, il est alors possible calculer la valeur singulière structurée et de déduire les bornes des paramètres pour lesquelles le système d'interconnexion reste stable. Dans le contexte des SSNN, ces résultats fournissent une méthode subsidiaire d'analyse de stabilité et de performance pour la représentation LPV-LFT. Nous introduisons alors un critère de performance similaire au gain \mathcal{L}_2 , mais dont les garanties sont limitées aux hypothèses d'étude propres à la μ -analyse.

Définition 3.4 : Performance par μ -analyse d'un SSNN

Soit \tilde{H}^{NN} un SSNN et $F_u(\tilde{H}(z), \tilde{\Delta})$ sa représentation LPV-LFT avec $\|\tilde{\Delta}\|_{\mathcal{L}_2} \leq 1$, la performance par μ -analyse du réseau \tilde{H}^{NN} est définie telle que :

$$\left\| \tilde{H}^{\text{NN}} \right\|_{\mu} := \mu_{\underline{\Delta}}(\tilde{H}(z)) \quad (3.64)$$

Par ses hypothèses, la μ -analyse n'apporte que des conclusions pour des paramètres Δ incertains et non-variants. Ainsi, les garanties théoriques de stabilité sont invalidées lors de l'étude de SSNN qui comportent par définition des paramètres variants au cours du temps. Cependant, pour le type de système que nous rencontrons dans nos travaux, la μ -analyse a permis d'apporter des résultats pertinents, comme nous le verrons dans le chapitre 6, là où les autres approches n'ont pu fournir de résultats à cause des difficultés numériques.

De plus, cet outil d'analyse requiert également moins de temps de calcul par rapport à d'autres méthodes lorsque les systèmes sont relativement complexes. Cette caractéristique représente un réel avantage si ce calcul est itérativement effectué pour prendre part à une optimisation plus haut-niveau comme nous le proposerons dans le chapitre 4. Pour conclure, nous privilégierions alors par défaut dans nos travaux les méthodes fondées sur la gain \mathcal{L}_2 et les LMI, mais nous nous orienterons vers la μ -analyse si ces dernières rencontrent d'importants problèmes de résolution numériques.

SECTION 3.5

Conclusions

Nous avons présenté dans ce chapitre une méthode permettant d'établir une représentation LPV des réseaux de type SSNN qui comportent des fonctions d'activations non-linéaires respectant une condition de secteur. Puisque ce processus est une transformation plutôt qu'une linéarisation locale, le modèle LPV résultant est exactement égal au système non-linéaire original. Nous avons déterminé, par la suite, la formulation LFT du réseau qui consiste à séparer le modèle en une partie linéaire et une partie diagonale non-linéaire, mais pour laquelle les bornes de chaque paramètre sont connues. Deux méthodes complémentaires ont également été développées afin d'affiner la modélisation. Tout d'abord, la présence des biais nécessite de considérer ces derniers comme des perturbations d'entrées constantes. Pour cette raison, nous avons établi une formulation analogue du SSNN, mais dont les biais ont été retirés à l'aide d'un changement de coordonnées autour du point d'équilibre. De plus, nous avons détaillé une technique permettant de normaliser les paramètres variants. En tirant profit des données d'apprentissage, les bornes des paramètres peuvent être déterminées afin de ne considérer qu'uniquement la région pertinente de l'espace d'état des non-linéarités plutôt que la limite de secteur dans son intégralité. D'un point de vue pratique, ces approches contribuent grandement à améliorer les résultats d'analyses ou éventuellement de synthèses reposant sur le modèle LPV-LFT.

Dans un second temps, nous avons proposé des outils d'analyse de stabilité et de performance de SSNN via le modèle LPV-LFT. Nous avons au préalable déterminé la modélisation de certains types d'interconnexion de SSNN afin de pouvoir les manipuler selon un modèle équivalent. Les

méthodes d'analyses reposent sur la résolution numérique de problème d'optimisation de type LMI, dont les résultats ont été introduits dans le chapitre précédent (chapitre 2) et sont à présent employés pour l'étude de SSNN. Néanmoins, lors de l'analyse, le modèle à l'étude englobe une classe de systèmes bien plus importante que le comportement réel du réseau ce qui peut limiter les résultats. Afin de restreindre l'ensemble de systèmes abordé, nous avons également proposé l'ajout d'un filtrage sur les paramètres variants afin de conditionner la trajectoire des paramètres et notamment les vitesses de variations envisagées. De même, nous avons présenté une méthode subsidiaire par μ -analyse dans le but de faire face aux systèmes plus complexes qui occasionnent des difficultés numériques de résolution.

Deuxième partie

**Méthodes de synthèse de contrôleurs
neuronaux robustes**

Identification neuronale de contrôleurs robustes

Sommaire

4.1	Introduction	86
4.2	Apprentissage de contrôleurs et enjeu de robustesse	87
4.2.1	Méthodologie d'apprentissage de contrôleur	87
4.2.2	Problème de la stabilité	88
4.2.3	Application au chariot mobile avec pendule	89
4.2.3.1	Présentation du système et du contrôleur considéré	89
4.2.3.2	Identification neuronale du contrôleur	90
4.2.3.3	Simulation de la boucle d'asservissement	92
4.3	Méthodes d'évaluation des marges de stabilité	94
4.3.1	Marges de stabilité dans le cadre linéaire invariant	94
4.3.2	Marges de stabilité de SSNN	101
4.3.3	Méthodologie d'apprentissage et d'évaluation des marges	101
4.3.3.1	Identification neuronale du contrôleur et du système	101
4.3.3.2	Évaluation des marges de stabilité via le modèle LPV-LFT associé	103
4.3.4	Application	104
4.4	Méthodes d'apprentissage avec prise en compte de stabilité	105
4.4.1	Notions de problème d'optimisation multi-objectifs	106
4.4.2	Formulation multi-objectifs de l'apprentissage de contrôleurs robustes	107
4.4.3	Optimisation multi-objectifs sans gradient	108
4.4.3.1	Apprentissage par algorithmes génétiques	109
4.4.3.2	Formulation multi-objectifs	109
4.4.3.3	Méthodologie d'implémentation	110
4.4.3.4	Techniques d'amélioration de la convergence pour l'apprentissage neuronal	111
4.4.4	Application	112
4.5	Conclusions et perspectives	117
4.5.1	Conclusions	117
4.5.2	Perspectives	118

SECTION 4.1

Introduction

Nous nous intéressons dans ce chapitre à l'apprentissage de contrôleurs neuronaux en tenant compte de critères de robustesse. Nous développons alors des méthodes en complément des techniques d'apprentissage présentées dans le chapitre 1. Comme nous l'introduisons dans la section 4.2, l'apprentissage de contrôleur suscite en effet une problématique quant à la stabilité lorsqu'il sera introduit dans la boucle d'asservissement. En l'état, il est alors difficile à l'issue de l'apprentissage et au seul regard des performances d'apprentissages, de préjuger des aptitudes du contrôleur en boucle fermée et des propriétés de robustesse de la loi de commande. Dans ce sens, nous présentons dans la section 4.3 une méthode afin de caractériser et d'évaluer *a posteriori* les marges de stabilité au vu du contrôleur neuronal obtenu. Différentes marges sont présentées et leur évaluation est fondée sur les méthodes d'analyses de réseaux selon leurs représentations LPV-LFT qui ont été développées dans le chapitre 3. Nous proposons par la suite, dans la section 4.4, une méthode permettant d'inclure *a priori* les critères de robustesse dans la phase d'apprentissage. La synthèse du contrôleur robuste s'effectue alors selon une formulation multi-objectifs et des méthodes d'apprentissages s'appuyant sur des algorithmes génétiques.

La méthodologie proposée dans ce chapitre pour le développement d'un contrôleur neuronal robuste par imitation à partir de donnée peut être résumée sur la figure 4.1. Cette méthodologie s'appuie sur les différentes méthodes et outils qui seront développés dans ce chapitre.

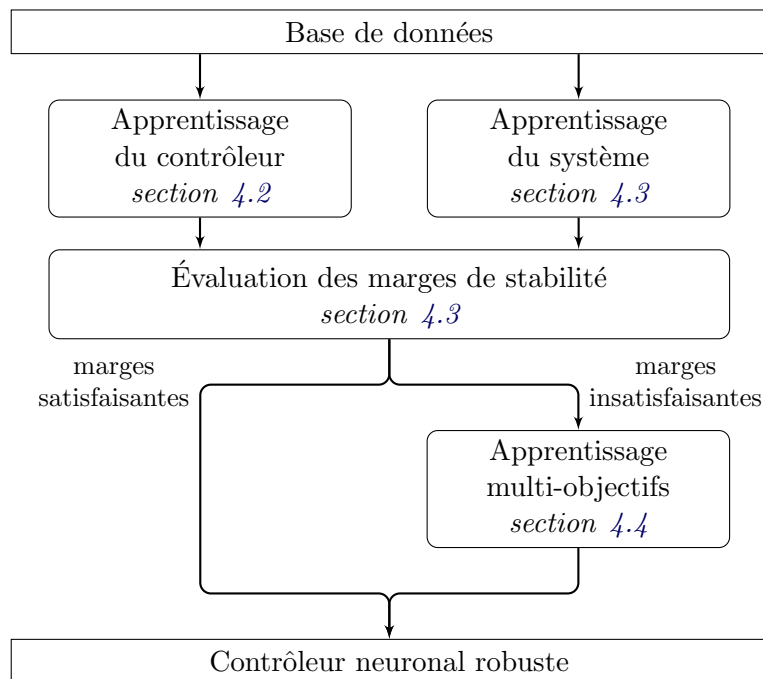


FIGURE 4.1 – Méthodologie de développement de contrôleurs neuronaux robustes à partir de données

Apprentissage de contrôleurs et enjeu de robustesse

4.2.1 Méthodologie d'apprentissage de contrôleur

Considérons la boucle d'asservissement classique d'un système physique représentée par la figure 4.2 où le système est de façon générale multivariable et potentiellement non-linéaire. Nous supposons qu'un contrôleur à disposition est capable de piloter le système afin que les sorties soient asservies de manière satisfaisante vis-à-vis d'un cahier des charges potentiellement complexe. Cet agent, assurant la régulation, peut être un correcteur déjà existant ou simplement un opérateur capable de piloter manuellement le système. L'objectif est alors d'imiter le contrôleur par les méthodes neuronales. Dans un contexte d'apprentissage supervisé ou d'identification de systèmes dynamiques, une manière de procéder intuitive est alors de collecter les signaux d'entrées-sorties du contrôleur puis d'élaborer un modèle à partir d'une régression temporelle des données.

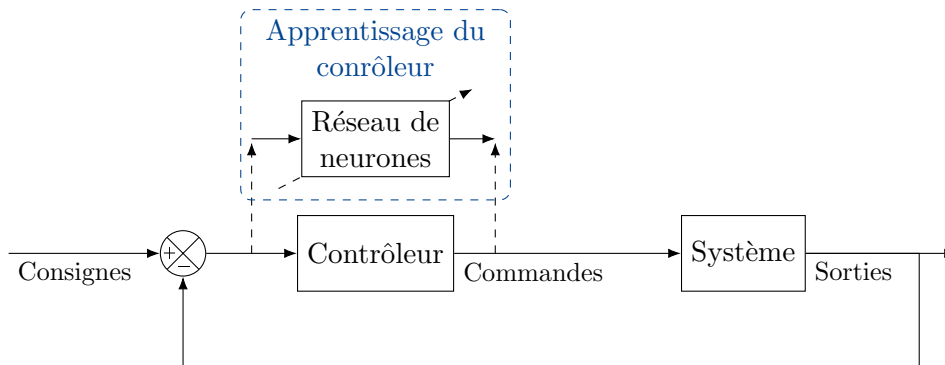


FIGURE 4.2 – Schéma d'apprentissage de contrôleur

Les données ont un rôle majeur dans le cadre de l'apprentissage supervisé et de l'identification. Il est nécessaire qu'elles contiennent suffisamment d'information pour que les résultats soient pertinents c'est-à-dire généralisables, en ce sens que l'utilisation du correcteur appris permette d'obtenir un pilotage satisfaisant et robuste pour une gamme de sollicitations la plus large possible. Ainsi, un point important est de s'assurer que l'ensemble du domaine d'opération soit représenté dans les scénarios d'apprentissage. De plus, nous savons depuis longtemps avec la théorie de l'identification des systèmes linéaires que les données doivent également caractériser les propriétés fréquentielles du système à identifier. L'identification s'effectue ici en boucle fermée, ce qui est un sujet récurrent de la littérature, nous pouvons par exemple citer [Söderström et Stoica, 1989, Nelles, 2020] même si le problème d'identification est adressé pour le système et non le contrôleur qui reste un système dynamique potentiellement instable. Dans ce contexte, il a été démontré que la corrélation des signaux peut amener certaines gammes de fréquence à disparaître ce qui peut provoquer des soucis pour l'identification notamment si le contrôleur est linéaire et invariant dans le temps (LTI). Une manière de procéder est d'insérer un signal d'excitation additionnel dans la boucle d'asservissement, en entrée du système comme l'illustre la figure 4.3. Comme il est indiqué dans [Nørgård *et al.*, 2000], ce signal supplémentaire peut permettre d'accorder une part plus importante aux hautes fréquences qui peuvent ne pas être excitées correctement à cause de la présence du contrôleur qui engendre des commandes souvent

basses fréquences. Nous voyons là aussi un moyen d'exciter certaines aptitudes du contrôleur comme son comportement intégrateur qui ne peut éventuellement pas être mis en évidence par l'excitation de la consigne si le système est lui même intégrateur. Ainsi, lorsque que cela est possible, l'ajout de cette perturbation d'excitation offre un degré de liberté supplémentaire afin d'obtenir des données comportant toutes les gammes de fréquence.

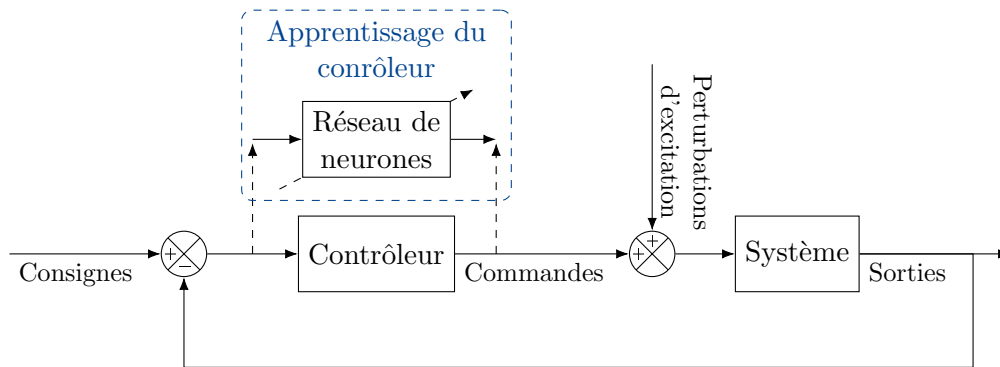


FIGURE 4.3 – Schéma d'apprentissage de contrôleur avec signal perturbateur d'excitation

4.2.2 Problème de la stabilité

À l'issue du processus d'identification et lorsque l'entraînement du contrôleur neuronal présente des performances satisfaisantes, il est alors escompté que le contrôleur identifié offre un comportement similaire par rapport à celui réel lorsqu'il sera introduit dans la boucle d'asservissement. Néanmoins, dans la pratique, ce comportement attendu n'est pas toujours observé et peut même, dans le pire cas, conduire à un comportement instable.

Lorsque le contrôleur neuronal est impliqué dans la boucle d'asservissement, aucune garantie ne peut être avancée a priori concernant la stabilité de la boucle globale sans une étude préalable. Le contrôleur doit faire face aux différentes incertitudes auxquelles sera soumise la boucle et les marges de robustesse doivent être suffisantes afin d'assurer la stabilité. Plusieurs sources d'erreurs peuvent être à l'origine de ces incertitudes. D'une part, le contrôleur élaboré suite à l'identification présente une erreur résiduelle par rapport au contrôleur réel. D'une autre part, un écart entre le système réel et celui ayant permis le recueil des données d'entraînement est également inévitable. Même si ces écarts de comportement sont faibles, leur évolution n'est pas caractérisée lors de l'apprentissage, ainsi, une fois le contrôleur interconnecté en boucle fermée avec le système, les erreurs peuvent rapidement s'accumuler, voire même s'auto-alimenter ce qui peut aboutir à l'instabilité de la boucle fermée. De plus, lors de l'apprentissage, l'erreur à minimiser est calculée selon les données entrées-sorties du contrôleur ce qui correspond à une évaluation de son comportement en boucle ouverte. Ainsi, il n'est pas possible de préjuger du comportement du contrôleur une fois en boucle fermée même s'il présente une erreur d'apprentissage minimale.

4.2.3 Application au chariot mobile avec pendule

4.2.3.1 Présentation du système et du contrôleur considéré

Tout au long de ce chapitre, le chariot mobile avec pendule présenté par la figure 4.4 est choisi pour illustrer l'approche. L'objectif est d'asservir la position du chariot, qui est pilotée par un moteur à courant continu, tout en stabilisant le pendule, ici en position basse, qui est monté sur le chariot.

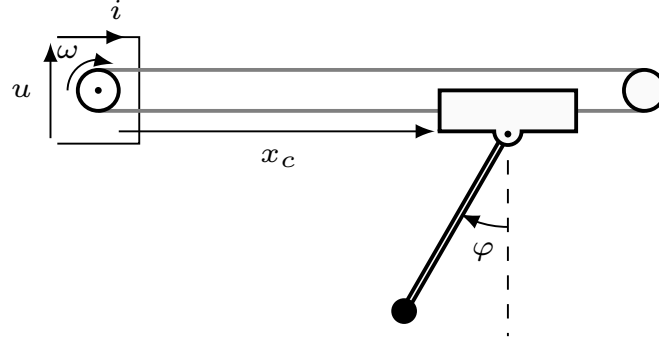


FIGURE 4.4 – Chariot mobile avec pendule

Le système considéré peut être modélisé de façon simplifiée par les équations suivantes en négligeant le couple de rappel du pendule sur le chariot :

$$\begin{cases} L \frac{di(t)}{dt} + Ri(t) + \phi \omega(t) = u(t) \\ J \frac{d\omega(t)}{dt} + f\omega(t) + \gamma(t) = \phi i(t) \\ \frac{dx_c(t)}{dt} = \frac{r}{n} \omega(t) \\ \cos(\varphi(t)) \frac{d^2 x_c(t)}{dt^2} + l \frac{d^2 \varphi(t)}{dt^2} + f_\alpha \frac{d\varphi(t)}{dt} + g \sin(\varphi(t)) = 0 \end{cases} \quad (4.1)$$

où $i(t)$, $u(t)$ sont le courant et la tension du moteur, $\omega(t)$ est la vitesse de rotation du moteur, $x_c(t)$ est la position du chariot, $\varphi(t)$ est l'angle du pendule et $\gamma(t)$ est un couple perturbateur. Le tableau 4.1 présente les valeurs numériques nominales de cet exemple.

Afin de disposer d'un système linéaire pour l'analyse et la synthèse, les équations du système peuvent être linéarisées en position basse ($\varphi \approx 0$) et en négligeant la constante de temps électrique ($L/R \approx 10^{-4}$ s) suivant la représentation d'état définie par :

$$\frac{d}{dt} \begin{bmatrix} x_c(t) \\ \omega(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} 0 & r/n & 0 & 0 \\ 0 & -a & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & ad & -g/l & -f_\alpha/l \end{bmatrix} \begin{bmatrix} x_c(t) \\ \omega(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ b \\ 0 \\ -bd \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ -c \\ 0 \\ cd \end{bmatrix} \gamma(t) \quad (4.2)$$

$$a = \frac{\phi^2}{RJ} + \frac{f}{J}, \quad b = \frac{\phi}{RJ}, \quad c = \frac{1}{J}, \quad d = \frac{r}{ln}$$

Afin de disposer d'un contrôleur à imiter, nous choisissons de synthétiser un contrôleur K_{syn} à trois degrés de liberté dont les entrées sont l'écart entre la consigne et la position, la position du chariot et l'angle du pendule conformément au schéma de la figure 4.5. Le système G utilisé

TABLE 4.1 – Valeurs nominales des paramètres

Paramètre	Signification	Valeur	Unité
R	résistance de l'induit du moteur	2.3	Ω
L	inductance de l'induit du moteur	2e-4	H
ϕ	coefficient de couple et de f.c.e.m	0.0162	Nm/A
J	inertie ramenée sur l'axe moteur	5e-6	kg.m ²
f	coefficient de frottement (moteur)	6e-5	N·m/rad.s
r	rayon de la poulie	0.022	m
n	rapport de réduction	17	-
l	longueur du pendule	0.275	m
f_α	coefficient de frottement (pendule)	0.3	m/s
g	accélération de la pesanteur	9.81	m/s ²

pour la synthèse est le modèle linéarisé autour du point $\varphi \approx 0$ présenté par les équations (4.2). Le calcul du contrôleur multivariable est réalisé à l'aide d'une approche \mathcal{H}_∞ selon la méthode *hinfsyn* [Doyle *et al.*, 1988] de MATLAB. Le schéma de synthèse suivi et le cahier des charges sont présentés dans les travaux de [Feyel, 2017]. Les filtres de pondération utilisés pour la synthèse sont déterminés afin d'élaborer un contrôleur présentant des marges de stabilité et des performances appropriées aux besoins de démonstrations. Ces marges de stabilité seront étudiées dans la section 4.3.1 abordant ce sujet pour les systèmes linéaires et dans la section 4.3.4 de mise en pratique. La synthèse d'ordre plein pour les filtres considérés mène à un contrôleur d'ordre 6, assurant un suivi de consigne sans erreur statique, un rejet de perturbation et un signal de commande limité peu sensible aux bruits de mesure.

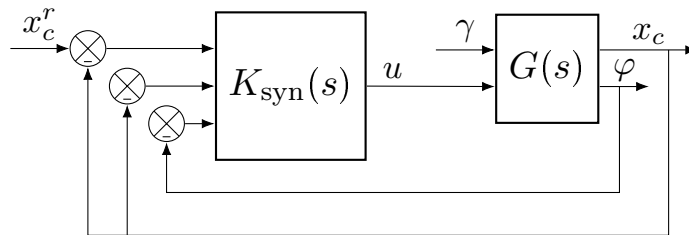


FIGURE 4.5 – Contrôleur à trois degrés de liberté

4.2.3.2 Identification neuronale du contrôleur

Nous nous consacrons dans cette partie à l'apprentissage du contrôleur K_{syn} à trois degrés de liberté (figure 4.5) du chariot mobile avec pendule. Nous appliquons la méthode d'écrite en 4.2.1 afin de générer les données qui seront utilisées pour l'apprentissage supervisé. Pour cet exemple d'illustration, nous générons alors deux bases de données Z_1^{dt} et Z_2^{dt} qui sont obtenues, dans le premier cas par la seule excitation de la consigne (schéma 4.2) puis dans le second par l'ajout du signal perturbateur d'excitation simultanément de la consigne (schéma 4.3). L'identification neuronale du contrôleur est finalement réalisée conformément à la procédure d'apprentissage présentée dans la section 1.5. Les paramètres retenus pour la collecte des données et ceux de

l'algorithme d'optimisation sont résumés dans le tableau 4.2.

TABLE 4.2 – Paramètres utilisés pour l'apprentissage dans le cas d'étude du chariot mobile avec pendule

Paramètre	Valeur
Période d'échantillonnage	1e-2s
Nombre de périodes d'échantillonnages	50 000
Intervalle des consignes	[-1, 1] m
Durée maximale des consignes	9s
Intervalle des perturbations	[-5, 5] V
Durée maximale des perturbations	6s
Algorithme d'apprentissage	Levenberg-Marquardt
Normalisation	<i>min-max</i>
Nombre de tirages	10
Taux de séparation des données	50%, 25%, 25%
Critère d'arrêt prématuré	500 itérations

Les résultats d'apprentissage des meilleurs réseaux obtenus pour le jeu de données sans perturbation (K_1^{NN}) et avec perturbation d'excitation (K_2^{NN}), sont respectivement présentés dans le tableau 4.3. La structure est détaillée selon la topologie employée, les dimensions des TDL d'entrée et de sortie (n_{in} et n_{out}), la fonction d'activation, le nombre de neurones des couches cachées (avec dans notre cas une seule couche cachée σ_1 de dimension n_1), ainsi que le nombre de paramètres que comporte la structure, c'est-à-dire à la dimension du vecteur de paramètres ($n_{\mathcal{W}}$) qui est donnée à titre indicatif. Les résultats présentent le nombre d'itérations et la durée de l'algorithme avant son arrêt ainsi que l'erreur de régression ou indice de performance (MSE) obtenu sur les jeux de données normalisés d'apprentissage, de validation et de test. Les résultats exposés sont seulement ceux du tirage retenu qui correspond à celui présentant l'erreur de validation la plus faible. Les structures des réseaux ont été sélectionnées itérativement afin d'obtenir des résultats satisfaisants et pour lesquelles une augmentation des paramètres de la structure ne permettait pas d'améliorer significativement les performances. Nous pouvons remarquer que malgré une structure de dimension légèrement plus élevée, le réseau entraîné sur les données Z_2^{dt} avec perturbations a une performance de régression bien plus faible. Les sorties estimées des contrôleurs neuronaux sont données par la figure 4.6 en comparaison des sortie des différentes bases de données pour l'ensemble de test.

TABLE 4.3 – Résultats d'apprentissage du contrôleur sur les bases de données obtenues par l'excitation de la consigne seule (K_1^{NN}) et avec la perturbation d'excitation supplémentaire (K_2^{NN})

	Structure						Résultats					
	topologie	n_{in}	n_{out}	σ_1	n_1	$n_{\mathcal{W}}$	iter.	durée	err. app.	err. val.	err. test	
K_1^{NN}	SSNNOE	2	6	tanh	5	101	3483	1h2m	2.1e-11	8.4e-10	4.0e-11	
K_2^{NN}	SSNNOE	5	6	tanh	5	155	1076	24m	7.7e-05	7.5e-05	4.4e-04	

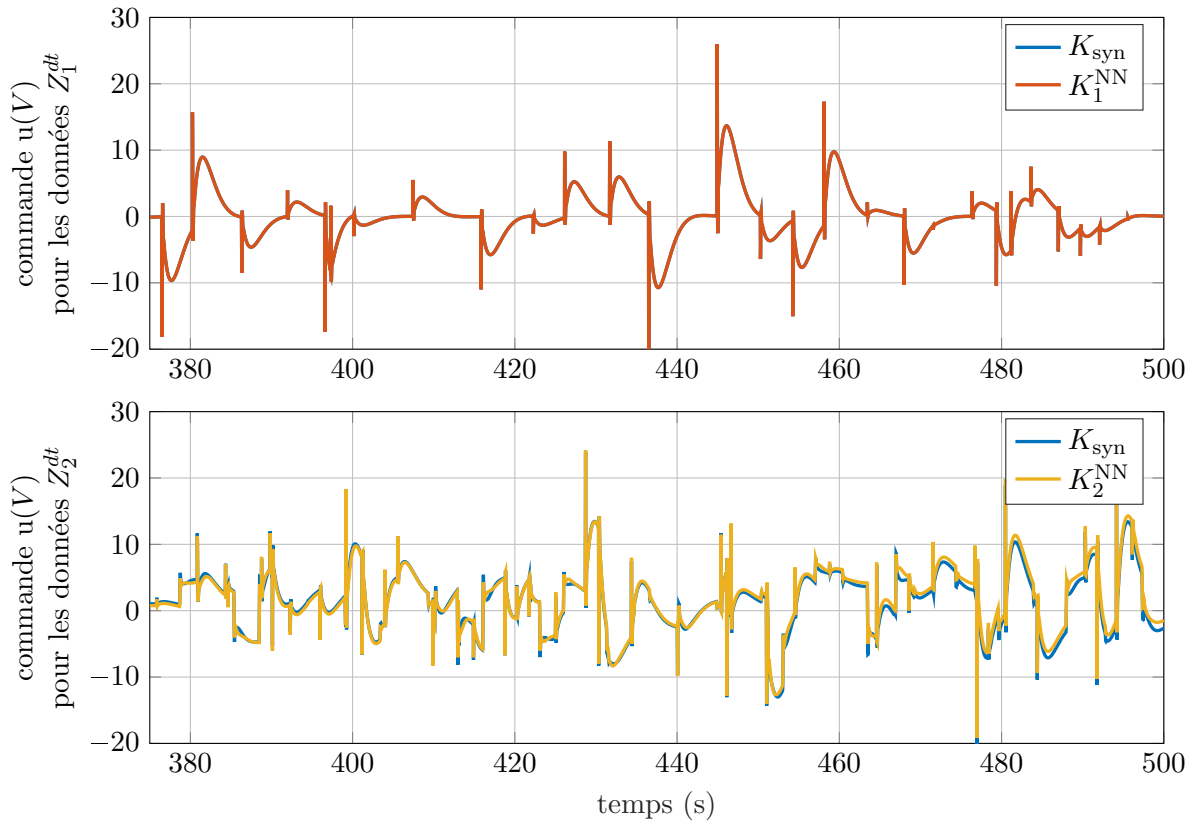


FIGURE 4.6 – Résultats d'apprentissage des contrôleurs neuronaux sur les données de l'ensemble de test

4.2.3.3 Simulation de la boucle d'asservissement

Nous développons à présent dans cette partie une série d'expérimentations à partir des deux contrôleurs neuronaux correspondant au tableau 4.3 obtenus dans la partie précédente. L'objectif est alors d'introduire les contrôleurs dans l'asservissement du système afin de s'assurer du bon comportement de la boucle fermée. Les simulations sont réalisées avec le système décrit par ses équations non-linéaires (4.1). La première expérience menée est la simulation en régime nominal c'est-à-dire sans variation du système, la figure 4.7 expose la réponse du contrôleur d'origine K_{syn} ainsi que celles des contrôleurs neuronaux K_1^{NN} et K_2^{NN} à un échelon de consigne puis de perturbation en entrée. Rappelons que l'étude du respect du cahier des charges des contrôleurs s'effectue non pas sur les performances dynamiques des asservissements mais bien sur leurs capacités à imiter le contrôleur d'origine.

Sur la première partie des simulations sans perturbation, les résultats démontrent un comportement nominal très satisfaisant puisque que la réponse de K_2^{NN} apparaît quasiment confondue avec celle de K_{syn} tandis que K_1^{NN} s'en approche également. Pour la seconde partie des simulations, l'apparition de la perturbation engendre l'instabilité dans le cas de K_1^{NN} alors que le contrôleur K_2^{NN} démontre l'assimilation de l'intégrateur du contrôleur d'origine. Notons que, contrairement aux valeurs de performances d'apprentissage à l'avantage de K_1^{NN} (voir le tableau 4.3), les performances d'imitation en boucle fermée sont au contraire à l'avantage de

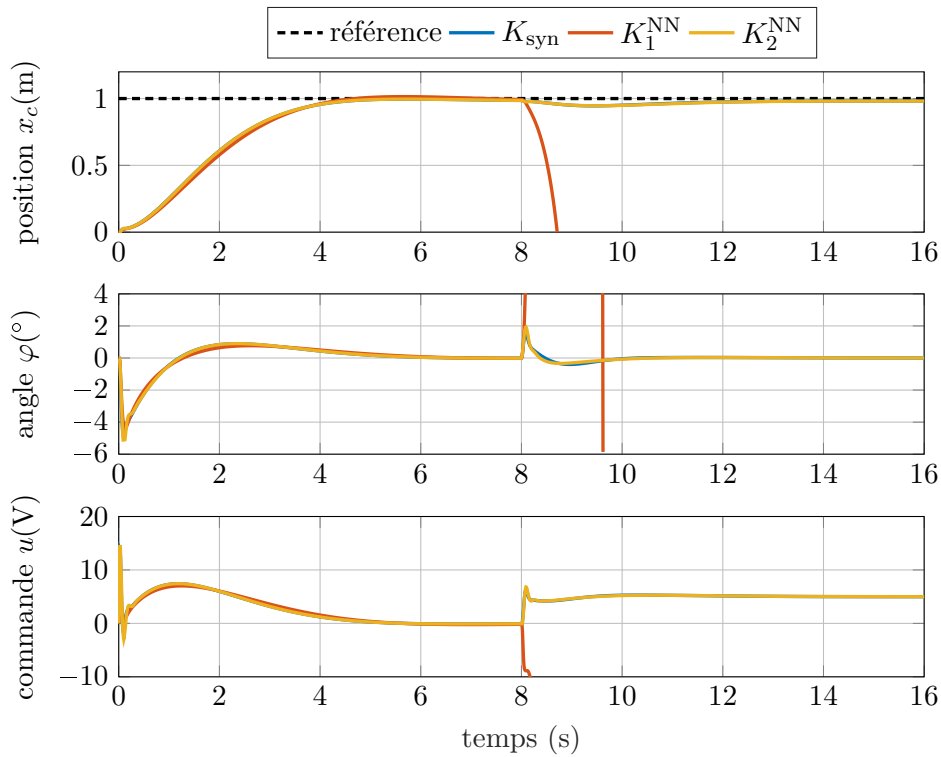


FIGURE 4.7 – Simulation des contrôleurs d’origine et neuronaux dans le cas nominal et avec rejet d’une perturbation d’entrée de -5V à partir de 8s

K_2^{NN} . De plus, ce constat intervient également sur les parties de simulations qui sont effectuées sans perturbation en entrée de système comme ce fut le cas pour les scénarios d’apprentissage du premier contrôleur.

Afin de témoigner des propriétés de robustesse des différents asservissements, nous envisageons par exemple des variations de gain du système comme l’illustre la figure 4.8. Ces variations surviennent en entrée de système par le biais du facteur multiplicatif positif Λ_u ou en sortie via Λ_y . Les réponses des boucles d’asservissement pour une légère variation d’entrée de $\Lambda_u = 1.1$ sont présentées par la figure 4.9a. Une seconde expérience est présentée par la figure 4.9b où une variation plus importante est effectuée, mais cette fois-ci en sortie $\Lambda_y = 2$. Dans le cas du premier contrôleur neuronal K_1^{NN} , les courbes de simulations témoignent de l’absence de marge de robustesse de l’asservissement aussi bien en entrée que en sortie de modèle. L’instabilité intervient de façon prématurée dès l’apparition de faibles variations par rapport aux scénarios appris. Concernant le second contrôleur à l’étude K_2^{NN} , ce dernier expose des résultats plus robustes, mais qui présentent tout de même des fortes oscillations dans la seconde simulation, c’est-à-dire dans le cas de variations plus importantes. Le comportement du contrôleur d’origine semble donc convenablement assimilé, néanmoins l’imitation c’est-à-dire l’identification du contrôleur, trouve ses limites pour un éloignement plus important du cas nominal d’apprentissage.

Ce cas d’étude démontre le challenge que représente l’assimilation des propriétés propres au contrôleur à identifier. Tout d’abord, le premier point soulevé par l’étude est qu’il n’existe pas de garantie évidente pour une erreur d’apprentissage faible impliquant un bon comportement

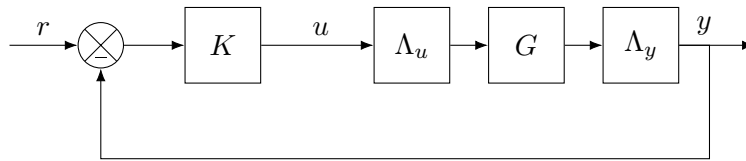


FIGURE 4.8 – Schéma de simulation de la boucle d'asservissement

du contrôleur dans la boucle d'asservissement. Une erreur temporelle faible en boucle ouverte ne garantit pas que l'erreur en boucle fermée l'est également. De plus, même si ce comportement se rapproche de celui à reproduire, la question de la robustesse en stabilité émerge alors dans le cas de nouveaux scénarios comme l'apparition de variations ou de perturbations. Finalement, l'intérêt que nous portons à cet exemple est de mettre en évidence le besoin d'un outil mathématique permettant de caractériser les propriétés de robustesse des contrôleurs neuronaux.

SECTION 4.3

Méthodes d'évaluation des marges de stabilité

Nous développons dans cette section une méthode permettant d'apprécier les marges de stabilité que peut conférer un contrôleur neuronal. En effet, les systèmes de contrôle sont toujours sujets à des effets potentiellement déstabilisants et c'est pourquoi ils nécessitent des marges de robustesse suffisamment importantes. D'une part, le système à contrôler est sujet à l'incertitude, il diffère plus ou moins de celui rencontré lors de l'élaboration de la base de données d'apprentissage. D'autre part, le contrôleur identifié pourra également être soumis à des scénarios non rencontrés lors de l'apprentissage pour lesquels il devra alors extrapoler le comportement à adopter. De plus, il est inévitable que le contrôleur appris et celui d'origine diffère d'une erreur d'identification, bien que cette différence soit quantifiable, il est difficile d'estimer son impact sur le comportement du système bouclé et ce même dans des conditions nominales.

Pour toutes les raisons influant sur la boucle fermée, il apparaît important de caractériser la stabilité émanant du contrôleur neuronal. De plus, la stabilité est rarement suffisante et c'est pourquoi des marges de stabilité sont nécessaires. Cette section passe brièvement en revue certains concepts importants pour évaluer la robustesse des systèmes en boucle fermée et propose une généralisation lorsque des structures neuronales sont impliquées. Les outils ainsi présentés donnent la possibilité d'évaluer la robustesse *a posteriori* du contrôleur par le biais de différentes marges de stabilité. Cela permettra également de les prendre en compte si nécessaire comme contraintes *a priori* dans le processus d'imitation comme nous le verrons dans la section 4.4.

4.3.1 Marges de stabilité dans le cadre linéaire invariant

Lorsque le système et le contrôleur sont linéaires et invariants dans le temps, les outils de l'automatique sont utilisés pour s'assurer que la boucle fermée possède des marges de stabilité suffisantes. D'une part, les marges de stabilité fournissent un indicateur de la qualité des réponses attendues, des marges élevées permettent de se prémunir d'éventuels comportements non désirés liés à l'instabilité (résonances ou dépassements importants par exemple). D'autre part, ces marges traduisent la robustesse de stabilité de l'asservissement, ainsi par mesure de précaution, des marges suffisantes préviennent des risques d'instabilité liés par exemple à une

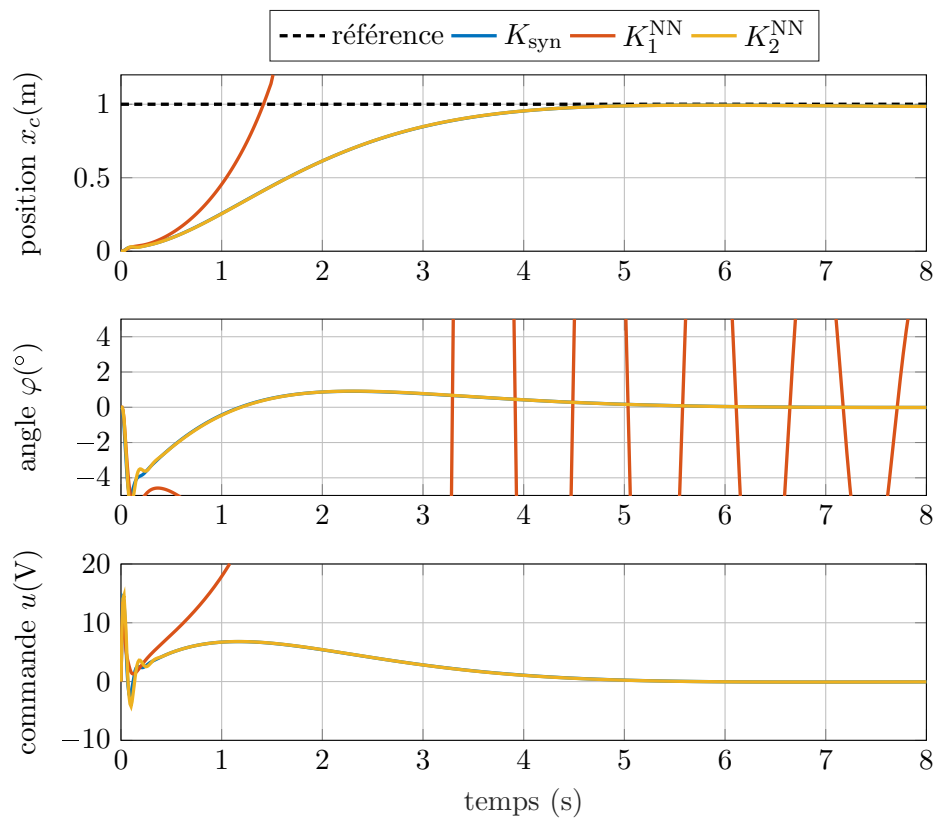
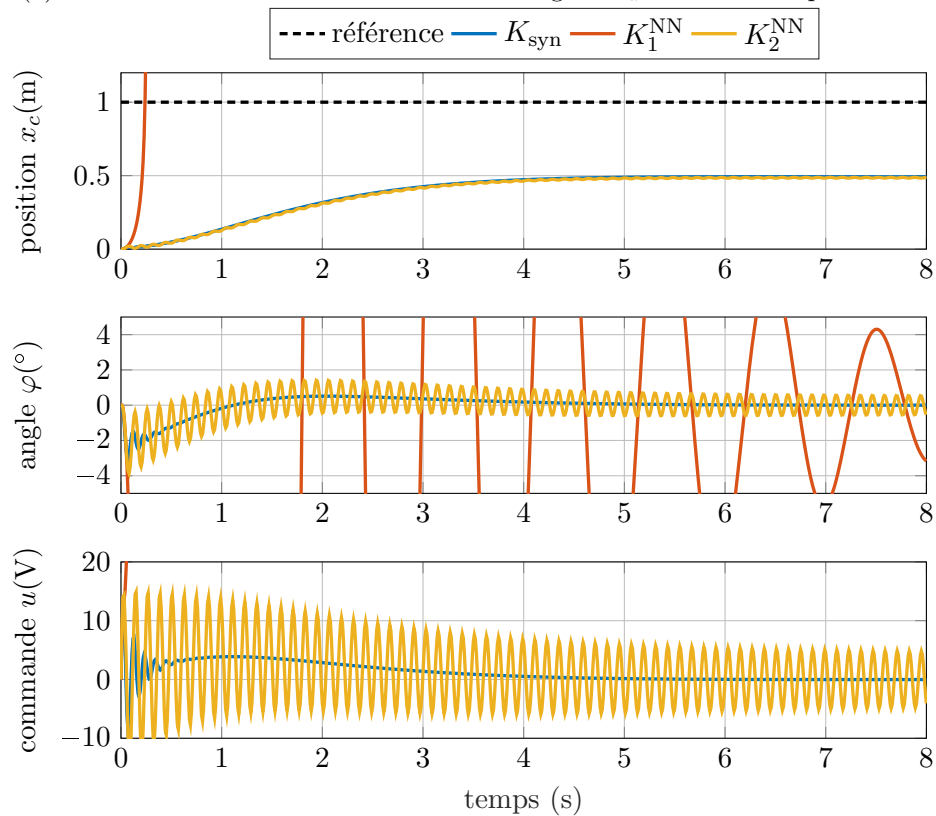
(a) Simulation dans le cas d'une variation de gain $\Lambda_u = 1.1$ et sans perturbation(b) Simulation dans le cas d'une variation de gain $\Lambda_y = 2$ et sans perturbation

FIGURE 4.9 – Simulation des contrôleurs d'origine et neuronaux dans des cas de variations de gain

erreur de modélisation du système.

Les marges les plus courantes pour quantifier la robustesse d'une loi de commande linéaire impliquant un contrôleur K avec un système G sont les traditionnelles marges de gain et marge de phase qui permettent de caractériser un transfert monovarié. Ces critères mesurent la quantité de variation de gain pour le premier cas et de perte de phase pour le second cas, que le transfert peut tolérer avant l'apparition d'une instabilité. Dans la continuité de l'approche d'analyse en boucle ouverte, les concepts de marges de gain et de phase peuvent être étendus au cas des systèmes multivariés (MIMO). Ainsi, dans le cas général, il convient de considérer séparément des marges en entrée ainsi que des marges en sortie du fait de la perte de la propriété de commutation du produit des fonctions de transfert si elles sont multivariées ($KG \neq GK$). Les marges de gain et de phase généralisées sont obtenues par l'introduction de facteurs multiplicatifs (réels ou complexes) en entrée $\Lambda_u = \text{diag}(\lambda_1, \dots, \lambda_{n_u})$ ou en sortie $\Lambda_y = \text{diag}(\lambda_1, \dots, \lambda_{n_y})$ comme le montre la figure 4.10.

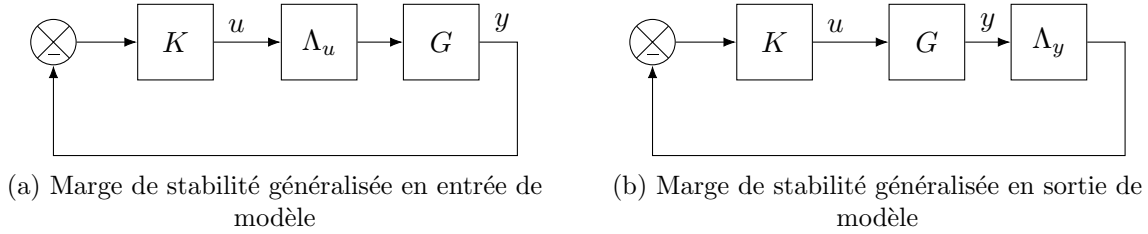


FIGURE 4.10 – Marges de stabilité généralisées

En fonction du type de facteurs multiplicatifs abordé, il est possible de déduire de différentes façons, les marges de gain et de phase généralisées. L'étude porte sur les variations maximales admissibles de gain et de phase afin que les boucles fermées des figures 4.10a et 4.10b restent stables. Pour ce faire, les facteurs multiplicatifs considérés sont de la forme :

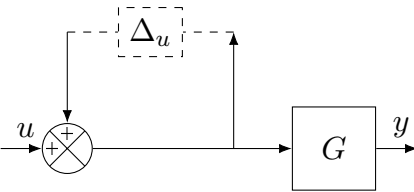
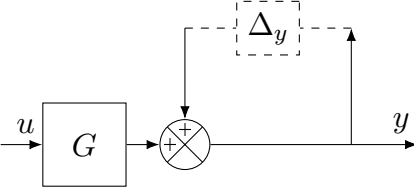
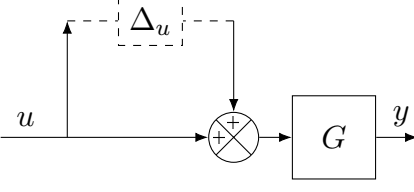
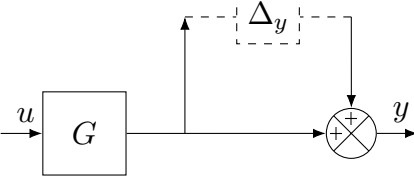
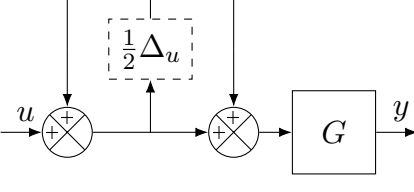
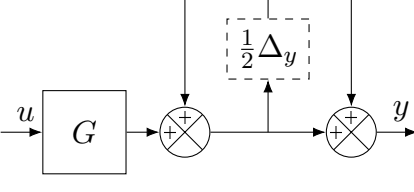
$$\Lambda_u = \text{diag}(k_i e^{j\phi_i}, 1, \dots, n_{n_u}) \quad \text{et} \quad \Lambda_y = \text{diag}(k_i e^{j\phi_i}, 1, \dots, n_{n_y}) \quad (4.3)$$

Les méthodes sont similaires selon que l'entrée ou la sortie soit considérée, c'est pourquoi nous omettrons l'indice dans les résultats présentés même s'ils sont obtenus dans le cas multivarié en fonction de l'entrée u ou la sortie y . La marge de gain généralisée GM indique que la boucle fermée est stable pour tous les gains réels de Λ dans l'intervalle $[\underline{\text{GM}}, \overline{\text{GM}}]$ qui est souvent formulé en unité décibels, c'est-à-dire $\text{GM} = 20 \log_{10}(\overline{\text{GM}}) = 20 \log_{10}(1/\underline{\text{GM}})$. De façon similaire, la marge de phase généralisée PM qui est souvent exprimée en degré assure la stabilité pour toute phase additionnelle de Λ dans l'intervalle $[-\text{PM}, +\text{PM}]$. Ces intervalles peuvent être déterminés en prenant en considération Λ à l'aide d'une matrice d'incertitude Δ de différentes façons, celles abordées dans ces travaux sont présentées dans le tableau 4.4. L'analyse des marges de stabilité est assimilée à l'analyse des fonctions de sensibilité directe S et complémentaire T qui sont impliquées dans de nombreuses modélisations de transfert en boucle fermée. Les fonctions de sensibilités directes et complémentaires sont respectivement définies en entrée et en sortie selon :

$$\begin{aligned} S_u &= (I_{n_u} + KG)^{-1}, & T_u &= (I_{n_u} + KG)^{-1}KG \\ S_y &= (I_{n_y} + GK)^{-1}, & T_y &= (I_{n_y} + GK)^{-1}GK \end{aligned} \quad (4.4)$$

Les matrices sont liées par les relations $S_u + T_u = I_{n_u}$ et $S_y + T_y = I_{n_y}$. Dans le cas monovarié (SISO), les fonctions de sensibilités coïncident selon $S = S_u = S_y$ et $T = T_u = T_y$.

TABLE 4.4 – Différents types d'incertitudes pour l'analyse de la stabilité

Type d'incertitude	Matrice d'incertitude	Système incertain	Transfert associé
Multiplicatif inverse en entrée	$\Lambda_u = \frac{1}{I + \Delta_u}$		S_u
Multiplicatif inverse en sortie	$\Lambda_y = \frac{1}{I + \Delta_y}$		S_y
Multiplicatif direct en entrée	$\Lambda_u = I + \Delta_u$		T_u
Multiplicatif direct en sortie	$\Lambda_y = I + \Delta_y$		T_y
Multiplicatif inverse et direct en entrée	$\Lambda_u = \frac{I - \frac{1}{2}\Delta_u}{I + \frac{1}{2}\Delta_u}$		$\frac{1}{2}(S_u - T_u)$
Multiplicatif inverse et direct en sortie	$\Lambda_y = \frac{I - \frac{1}{2}\Delta_y}{I + \frac{1}{2}\Delta_y}$		$\frac{1}{2}(S_y - T_y)$

Nous définissons maintenant les marges de stabilité envisagés dans ces travaux. Chacune est associée à un type d'incertitude considéré pour lequel il est possible de déduire des marges de gain et phase généralisés par l'application du théorème du petit gain (voir théorème 2.1) comme il est détaillé dans [Feyel, 2013]. Une fois encore, l'indice est omis, mais les marges déduites doivent être considérées en multivariable selon l'entrée u ou la sortie y .

Définition 4.1 : Marges de module

Soient le système G et le contrôleur K , les marges de module en entrée et en sortie sont respectivement définies par :

$$\text{MM}_u = \frac{1}{\|S_u\|_\infty}, \quad \text{MM}_y = \frac{1}{\|S_y\|_\infty} \quad (4.5)$$

La modélisation par une incertitude multiplicative inverse considère que le facteur multiplicatif est de la forme $\Lambda = (I + \Delta)^{-1}$. L'analyse implique alors la fonction de sensibilité directe S qui est associé à la marge de module (MM). Les marges de gain et de phase généralisés se déduisent selon :

$$\underline{\text{GM}} = \frac{1}{1 + \text{MM}}, \quad \overline{\text{GM}} = \frac{1}{1 - \text{MM}}, \quad \text{PM} = \pm 2 \arcsin\left(\frac{\text{MM}}{2}\right) \quad (4.6)$$

De par ses propriétés mathématiques, la norme $\|S\|_\infty$ ne peut être inférieure à 1, ainsi le critère de la marge de module permet de garantir au maximum $\text{GM} = [-6 \text{ dB}, \infty]$ et $\text{PM} = \pm 60^\circ$.

Définition 4.2 : Marges de module complémentaire

Soient le système G et le contrôleur K , les marges de module complémentaires en entrée et en sortie sont respectivement définies par :

$$\text{TM}_u = \frac{1}{\|T_u\|_\infty}, \quad \text{TM}_y = \frac{1}{\|T_y\|_\infty} \quad (4.7)$$

La modélisation par une incertitude multiplicative directe considère que le facteur multiplicatif est de la forme $\Lambda = I - \Delta$. L'analyse implique alors la fonction de sensibilité complémentaire T qui est associé à la marge de module complémentaire (TM). Les marges de gain et de phase généralisés se déduisent selon :

$$\underline{\text{GM}} = 1 - \text{TM}, \quad \overline{\text{GM}} = 1 + \text{TM}, \quad \text{PM} = \pm 2 \arcsin\left(\frac{\text{TM}}{2}\right) \quad (4.8)$$

De même, la norme $\|T\|_\infty$ ne peut être inférieure à 1 et donc le critère de la marge de module complémentaire permet de garantir au maximum $\text{GM} = [-\infty, 6 \text{ dB}]$ et $\text{PM} = \pm 60^\circ$.

Définition 4.3 : Marges de disque

Soient le système G et le contrôleur K , les marges de disque en entrée et en sortie sont respectivement définies par :

$$\text{DM}_u = \frac{1}{\left\| \frac{1}{2}(S_u - T_u) \right\|_\infty}, \quad \text{DM}_y = \frac{1}{\left\| \frac{1}{2}(S_y - T_y) \right\|_\infty} \quad (4.9)$$

La modélisation par une incertitude multiplicative directe et inverse qui est une combinaison des deux premières, considère que le facteur multiplicatif est de la forme $\Lambda = (I - \frac{1}{2}\Delta)(I + \frac{1}{2}\Delta)^{-1}$. La fonction alors impliquée est $\frac{1}{2}(S - T)$ qui est associée à la marge de disque (DM) pour l'anglicisme *disk margin* [Blight *et al.*, 1994, Seiler *et al.*, 2020]. L'étude de cette marge trouve ses motivations dans les marges de gain et de phase généralisées qui peuvent s'en déduire :

$$\underline{\text{GM}} = \frac{1 - \frac{1}{2}\text{DM}}{1 + \frac{1}{2}\text{DM}}, \quad \overline{\text{GM}} = \frac{1 + \frac{1}{2}\text{DM}}{1 - \frac{1}{2}\text{DM}}, \quad \text{PM} = \pm 2 \arctan\left(\frac{1}{2}\text{DM}\right) \quad (4.10)$$

La norme $\|\frac{1}{2}(S_u - T_u)\|_\infty$ ne peut quant à elle être inférieure à $\frac{1}{2}$, ainsi le critère de la marge de disque permet de garantir au maximum $\text{GM} = \pm\infty$ et $\text{PM} = \pm 90^\circ$.

Par une représentation dans le plan complexe, les marges de stabilité traduisent qualitativement la distance du tracé du lieu de Nyquist de la boucle ouverte avec le point critique (-1) . La marge de module qui est celle couramment utilisée dans la littérature correspond à la distance euclidienne entre le point critique et le lieu de Nyquist de la boucle ouverte. Elle s'interprète comme une région d'exclusion de Nyquist à respecter afin de vérifier la condition de stabilité. Les autres marges présentées ont également l'interprétation d'un cercle d'exclusion dont le centre n'est cependant pas au point critique, mais décalé par rapport au point critique. La figure 4.11 présente les trois régions d'exclusions dans le plan complexe correspondant aux trois marges de stabilité en entrée obtenues en reprenant l'exemple d'illustration de la section 4.2.3 du chariot mobile avec pendule. Le lieu de Nyquist est tel que $L(z) = K(z)G(z)$ où $G(z)$ correspond au modèle linéarisé du système (4.2) puis discrétisé et $K(z)$ au contrôleur synthétisé par la méthode \mathcal{H}_∞ puis également discrétisé.

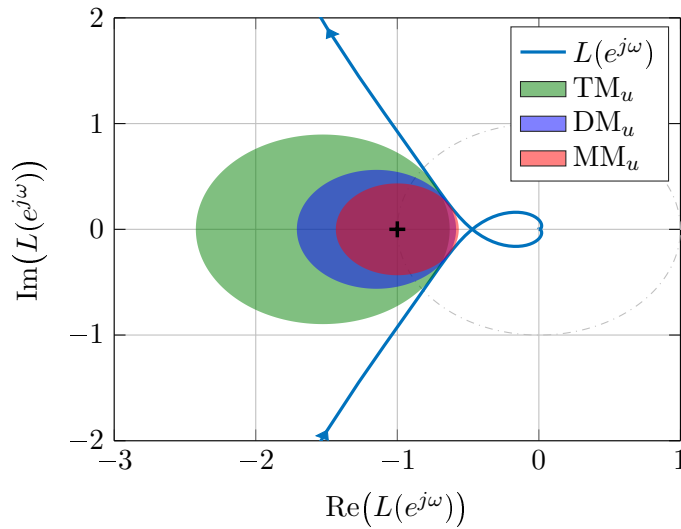


FIGURE 4.11 – Régions d'exclusion dans le plan de Nyquist obtenues pour les différentes marges de stabilité

Les marges de gain et de phase généralisées déduites des marges de stabilité, s'interprètent également comme une distance du point critique avec le lieu de Nyquist. Dans le cas de la marge de gain, cette distance est le long de l'axe réel tandis que la marge de phase mesure cette distance le long du cercle unité. Les variations maximales de gain et de phase peuvent se déduire des cercles

d'exclusion potentiellement asymétriques par rapport au point critique comme l'illustre la figure 4.12 pour le même exemple du chariot mobile avec pendule. Par cette méthode, les deux axes de variations sont envisagés en même temps ce qui engendre des marges nécessairement plus faibles que lorsqu'elles sont étudiées indépendantes l'une de l'autre. Néanmoins, cette démarche permet de considérer des combinaisons de variations de gain et de phase qui seraient indépendamment sans risque, mais éventuellement déstabilisantes si elles sont envisagées de façon simultanée. Le tableau 4.5 présente la valeur des différentes marges en entrée pour le même cas d'étude du chariot mobile ainsi que les marges généralisées déduites pour chaque approche.

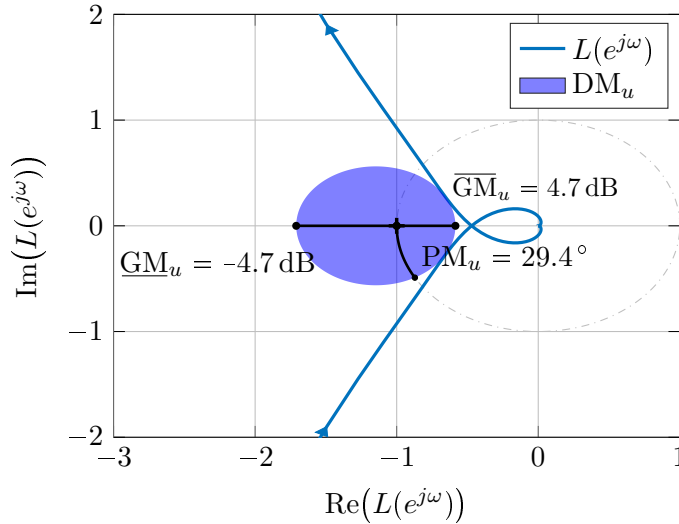


FIGURE 4.12 – Marges de gains et de phases généralisées déduites de la marge de disque

TABLE 4.5 – Marges de stabilité et leurs marges généralisées

marge étudiée		marges généralisées		
type	valeur	GM_u	$[\underline{\text{GM}}_u, \overline{\text{GM}}_u]$	PM_u
MM_u	0.43	4.9 dB	$[0.70, 1.77]$	25.1°
TM_u	0.59	4.0 dB	$[0.41, 1.59]$	34.2°
DM_u	0.52	4.7 dB	$[0.58, 1.71]$	29.4°

Remarque 4.1. Les trois marges de stabilité présentées ci-dessus peuvent être rassemblées dans le formalisme de la marge de disque symétrique des travaux de [Seiler et al., 2020]. La marge de disque symétrique α_{\max} est définie comme $\alpha_{\max} = \|S + \frac{\sigma-1}{2}\|_{\infty}^{-1}$ où σ est un paramètre d'asymétrie donné. La marge de disque du cas équilibré, c'est-à-dire $\sigma = 0$ exprime la marge de disque $\alpha_{\max} = \text{DM} = \|\frac{1}{2}(S - T)\|_{\infty}^{-1}$. Lorsque les informations disponibles sur le système permettent de conjecturer le type de variations, il peut être intéressant de favoriser une direction de variation de gain plutôt que l'autre à l'aide du paramètre σ . Quand l'augmentation relative du gain est favorisée par $\sigma = -1$ ou la diminution $\sigma = +1$, alors la condition de marge de disque se simplifie pour correspondre respectivement à $\alpha_{\max} = \text{MM} = \|S\|_{\infty}^{-1}$ et $\alpha_{\max} = \text{TM} = \|T\|_{\infty}^{-1}$.

4.3.2 Marges de stabilité de SSNN

Nous développons dans cette partie, les marges de stabilité pouvant être considérées pour les systèmes non-linéaires et plus particulièrement ici, ceux modélisés par des réseaux de neurones de type SSNN. Ces marges se fondent directement sur l'approche linéaire dont les concepts sont étendus au cadre des systèmes non-linéaires que sont les SSNN grâce aux liens établis avec les systèmes LPV dans le chapitre 3. Ainsi, les structures des systèmes étudiés demeurent identiques à celles présentées dans la section précédente, cependant la norme à l'étude est la norme induite \mathcal{L}_2 (voir la définition 2.7).

Nous définissons donc les marges de stabilité considérées dans ces travaux afin de quantifier la robustesse de la boucle fermée comportant un contrôleur K^{NN} et un système G^{NN} tout deux de type SSNN. Les interconnexions de ces réseaux liées aux marges de stabilité peuvent être aisément définies à l'aide des travaux de combinaison de SSNN du chapitre 3 tels que les équations d'interconnexion de deux SSNN (3.52) associées à la figure 3.6.

Définition 4.4 : Marges de stabilité de SSNN

Pour deux réseaux K^{NN} et G^{NN} de type SSNN, les marges de stabilités suivantes sont définies telles que

- les marges de module en entrée et en sortie sont respectivement données par :

$$\text{MM}_u^{\mathcal{L}_2} = \frac{1}{\|S_u^{\text{NN}}\|_{\mathcal{L}_2}}, \quad \text{MM}_y^{\mathcal{L}_2} = \frac{1}{\|S_y^{\text{NN}}\|_{\mathcal{L}_2}} \quad (4.11)$$

- les marges de module complémentaire en entrée et en sortie sont respectivement données par :

$$\text{TM}_u^{\mathcal{L}_2} = \frac{1}{\|T_u^{\text{NN}}\|_{\mathcal{L}_2}}, \quad \text{TM}_y^{\mathcal{L}_2} = \frac{1}{\|T_y^{\text{NN}}\|_{\mathcal{L}_2}} \quad (4.12)$$

- les marges de disque en entrée et en sortie sont respectivement données par :

$$\text{DM}_u^{\mathcal{L}_2} = \frac{1}{\|D_u^{\text{NN}}\|_{\mathcal{L}_2}}, \quad \text{DM}_y^{\mathcal{L}_2} = \frac{1}{\|D_y^{\text{NN}}\|_{\mathcal{L}_2}} \quad (4.13)$$

où les réseaux S_u^{NN} , S_y^{NN} , T_u^{NN} , T_y^{NN} , D_u^{NN} , D_y^{NN} sont issus des interconnexions définies dans le tableau 4.6.

4.3.3 Méthodologie d'apprentissage et d'évaluation des marges

Les différentes marges de stabilité étant définies, nous présentons à présent une méthode afin d'évaluer ces marges pour une boucle d'asservissement donnée.

4.3.3.1 Identification neuronale du contrôleur et du système

La méthode proposée afin d'apprécier les marges de stabilité que confère le contrôleur neuronale n'échappe pas à la mise en place préalable d'un modèle du système dynamique à contrôler. Ainsi, il est nécessaire d'effectuer indépendamment l'apprentissage du contrôleur et du système

TABLE 4.6 – Schémas d'interconnexion associés aux marges de stabilité de SSNN

Réseau	Schéma d'interconnexion	Matrices définissant l'interconnexion (3.52)
S_u^{NN}		$ \begin{aligned} W_1^p &= W_1^u, & B_p &= B_1, \\ C_p &= 0, & D_{pp} &= I, \\ D_{pu} &= I, & b_p &= 0, \\ D_p &= 0 \end{aligned} $
S_y^{NN}		$ \begin{aligned} W_1^p &= 0, & B_p &= 0, \\ C_p &= C_1, & D_{pp} &= I, \\ D_{pu} &= D_1, & b_p &= b_{y1}, \\ D_p &= I \end{aligned} $
T_u^{NN}		$ \begin{aligned} W_1^p &= W_1^u, & B_p &= B_1, \\ C_p &= 0, & D_{pp} &= 0, \\ D_{pu} &= I, & b_p &= 0, \\ D_p &= 0 \end{aligned} $
T_y^{NN}		$ \begin{aligned} W_1^p &= 0, & B_p &= 0, \\ C_p &= C_1, & D_{pp} &= 0, \\ D_{pu} &= D_1, & b_p &= b_{y1}, \\ D_p &= I \end{aligned} $
D_u^{NN}		$ \begin{aligned} W_1^p &= W_1^u, & B_p &= B_1, \\ C_p &= 0, & D_{pp} &= I, \\ D_{pu} &= \frac{1}{2}I, & b_p &= 0, \\ D_p &= 0 \end{aligned} $
D_y^{NN}		$ \begin{aligned} W_1^p &= 0, & B_p &= 0, \\ C_p &= C_1, & D_{pp} &= \frac{1}{2}I, \\ D_{pu} &= D_1, & b_p &= b_{y1}, \\ D_p &= I \end{aligned} $

comme l'illustre la figure 4.13. D'une part, l'identification du contrôleur aboutit à un modèle neuronale K^{NN} dont il convient d'étudier les propriétés de stabilité. D'autre part, l'identification du système mène à un modèle G^{NN} indispensable à l'étude.

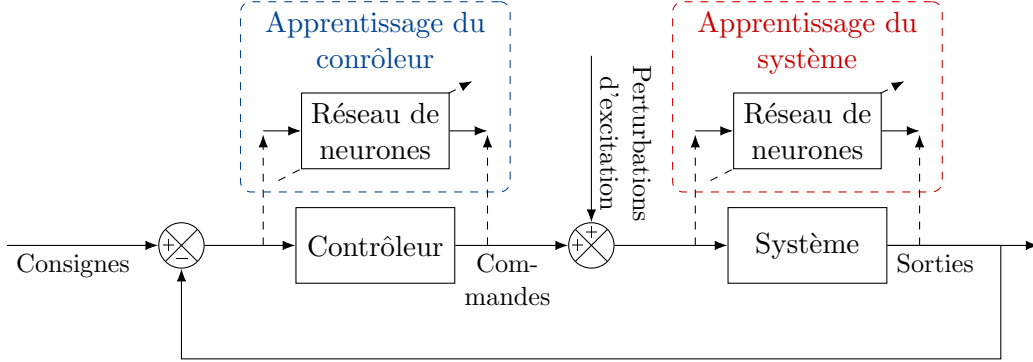


FIGURE 4.13 – Schéma d'apprentissage de contrôleur et du système à asservir

À partir des modèles neuronaux des deux acteurs de la boucle d'asservissement, il est possible de définir le modèle nécessaire à l'étude de la stabilité. Ce modèle d'analyse est un réseau M^{NN} qui correspond à l'une des structures d'interconnexion du contrôleur K^{NN} avec le système G^{NN} définies dans le tableau 4.6. L'analyse de la robustesse de la boucle fermée coïncide désormais à l'analyse des marges de stabilité du réseau M^{NN} qui est de la forme SSNN, c'est-à-dire :

$$M^{\text{NN}} : \begin{cases} h_1(k) = \sigma_1(W_1^x x(k) + W_1^u u(k) + b_1) \\ h_\ell(k) = \sigma_\ell(W_\ell h_{\ell-1}(k) + b_\ell) \quad \text{pour } \ell = 2, \dots, L \\ x(k+1) = Ax(k) + Bu(k) + W_x h_L(k) + b_x \\ y(k) = Cx(k) + Du(k) + b_y \end{cases} \quad (4.14)$$

4.3.3.2 Évaluation des marges de stabilité via le modèle LPV-LFT associé

L'évaluation des marges de stabilité du réseau d'analyse M^{NN} passe par la détermination du gain \mathcal{L}_2 du réseau $\|M^{\text{NN}}\|_{\mathcal{L}_2}$ (définition 4.4). Les travaux développés dans le chapitre 3, peuvent alors être appliqués afin d'évaluer cette norme via la représentation LPV-LFT équivalente du réseau.

Pour les besoins de l'analyse de performance, la structure SSNN est examinée par une représentation LPV-LFT équivalente. La transformation de SSNN en forme LPV-LFT a été développée dans la section 3.2. Ainsi, supposons qu'il existe $F_u(\tilde{M}(z), \tilde{\Delta})$ le modèle LPV-LFT associé au réseau d'interconnexion M^{NN} . De par sa construction, les paramètres variants isolés sont de la forme $\tilde{\Delta} = \text{diag}(\tilde{\Delta}_G, \tilde{\Delta}_K)$, c'est-à-dire que les paramètres sont respectivement relatifs au système et au contrôleur. À travers l'analyse de la forme LPV-LFT, une borne γ du gain \mathcal{L}_2 du réseau peut se déduire du théorème 3.4 reposant sur le lemme borné réel :

$$\|M^{\text{NN}}\|_{\mathcal{L}_2} < \gamma \iff \|F_u(\tilde{M}(z), \tilde{\Delta})\|_{\mathcal{L}_2} < \gamma \quad (4.15)$$

La marge de stabilité associée au réseau d'étude se déduit alors directement selon :

$$\frac{1}{\|M^{\text{NN}}\|_{\mathcal{L}_2}} = \frac{1}{\gamma} \quad (4.16)$$

L'analyse peut également être menée sur un système augmenté comme il est présenté par l'équation (3.60) où des filtres sur les paramètres variants sont incorporés afin d'exclure de l'analyse les vitesses de variations arbitrairement rapides.

De plus, lorsque le réseau d'analyse M^{NN} présente une complexité telle que les outils de résolution rencontrent des difficultés numériques, alors nous proposons d'étendre la conception des marges de stabilité aux résultats obtenus par μ -analyse comme il a été présenté dans la section 3.4.3 et avec toutes les réserves déjà énoncées dans cette section. À proprement parler, une estimation des marges de stabilité peut être obtenue par cette méthode au regard de certaines limitations théoriques. L'estimation de la performance du réseau $\|M^{\text{NN}}\|_{\mu}$ est effectué par le calcul des valeurs singulières structurés de la représentation LPV-LFT :

$$\|M^{\text{NN}}\|_{\mu} = \mu_{\underline{\Delta}}(\tilde{M}(z)) \quad (4.17)$$

Cette estimation conduit à une marge de stabilité associée par μ -analyse selon :

$$\frac{1}{\|M^{\text{NN}}\|_{\mu}} \quad (4.18)$$

4.3.4 Application

Afin d'illustrer la méthode d'évaluation des marges, nous proposons d'appliquer les développements proposés dans cette partie au problème d'introduction du chapitre correspondant à l'exemple du chariot avec pendule (section 4.2.3). L'objectif est d'identifier le contrôleur à trois degrés de liberté déjà existant ainsi que ses propriétés de robustesse. Pour les besoins de l'expérimentation, un modèle du système est disponible et le contrôleur existant est connu. Ainsi, les marges pour le contrôleur initial peuvent être estimées à l'aide des méthodes d'évaluation linéaires. Les SSNN étant par définition discrets, les analyses impliquant le contrôleur d'origine ou le modèle linéarisé du système sont réalisés après discrétisation. En raisonnant par exemple sur la marge de disque, le contrôleur à identifier présente une fois discrétisé, les marges de stabilité énoncées par le tableau 4.7.

TABLE 4.7 – Marges de stabilité du contrôleur à identifier évaluées pour le système linéarisé

Marges d'entrée			Marges de sortie		
DM _u	GM _u	PM _u	DM _y	GM _y	PM _y
0.52	4.7 dB	29.4°	0.23	2.0 dB	13.0°

L'identification du système non-linéaire est réalisée selon des paramètres identiques à ceux présentés dans le tableau 4.2 pour la collecte des données et l'algorithme d'apprentissage. Après une procédure d'essai-erreur de détermination des hyper-paramètres, les résultats du réseau final G^{NN} sont exposés dans le tableau 4.8. L'apprentissage est réalisé en deux temps selon la méthode décrite dans la section 1.5.3, un premier apprentissage en SSNNARX est utilisé afin d'initialiser le second apprentissage en SSNNNOE. La base de données utilisée est en l'occurrence Z_2^{dt} , obtenue par l'excitation simultanée de la consigne et de la perturbation d'entrée.

TABLE 4.8 – Résultats d'apprentissage du système non-linéaire du chariot mobile avec pendule

	Structure						Résultats				
	topologie	n_{in}	n_{out}	σ_1	n_1	$n_{\mathcal{W}}$	iter.	durée	err. app.	err. val.	err. test
G^{NN}	SSNNARX	2	2	tanh	5	66	2761	2m	3.9e-11	4.0e-11	2.4e-11
	SSNNOE	2	2	tanh	5	66	3448	47m	1.4e-08	6.0e-08	6.4e-08

Le réseau d'analyse de stabilité est par la suite déterminé pour chacun des contrôleurs associé au modèle neuronal G^{NN} du système. Les structures sont celles associées aux marges de disque d'entrée D_u^{NN} et de sortie D_y^{NN} . Le calcul du gain \mathcal{L}_2 est réalisé après transformation en système LPV-LFT équivalent de la boucle fermée globale, les valeurs obtenues ainsi que les marges généralisées sont présentées dans le tableau 4.9. Les marges obtenues pour le contrôleur d'origine K_{syn} permettent de renforcer la confiance accordée au modèle du système puisqu'elles se rapprochent des valeurs obtenues avec le modèle linéaire. Comme l'ont démontré les simulations de la section 4.2.3, le contrôleur neuronal K_1^{NN} semble disposer de très faibles marges de robustesse. L'analyse de K_2^{NN} confirme également les résultats de simulations puisque le contrôleur expose une marge d'entrée similaire à celle du modèle imité, mais une robustesse en sortie plus fragile.

TABLE 4.9 – Marges de stabilité des contrôleurs évaluées pour le système neuronal

	Marges d'entrée			Marges de sortie		
	$DM_u^{\mathcal{L}_2}$	GM_u	PM_u	$DM_y^{\mathcal{L}_2}$	GM_y	PM_y
K_{syn}	0.52	4.6 dB	29.0°	0.18	1.6 dB	10.5°
K_1^{NN}	0.02	0.2 dB	1.2°	0.01	0.1 dB	0.5°
K_2^{NN}	0.51	4.5 dB	28.6°	0.03	0.3 dB	1.8°

SECTION 4.4

Méthodes d'apprentissage avec prise en compte de stabilité

Nous abordons dans cette section une méthode permettant d'inclure la prise en compte a priori du critère de robustesse lors de la phase d'apprentissage. En effet, le problème de synthèse à l'étude consiste à calculer un contrôleur neuronal qui satisfait de bonnes performances d'imitation et qui garantit un certain degré de robustesse à la boucle fermée neuronale. Les méthodes proposées dans la partie 4.3.2 ci-dessus permettent de quantifier les marges de stabilité conférées par un contrôleur neuronal. Comme l'a montrée l'exemple introductif de la section 4.2.3, ce critère ne se retrouve pas nécessairement optimisé en parallèle lorsque que l'entraînement du contrôleur est basé sur l'erreur de simulation temporelle. Nous proposons ici d'explicitement prendre en considération l'indicateur de robustesse et d'examiner l'apprentissage de contrôleur en tant que problème d'optimisation multi-objectifs.

4.4.1 Notions de problème d'optimisation multi-objectifs

De façon générale, un problème d'optimisation multi-objectifs à n_{var} paramètres (ou variables de décision) et n_{obj} objectifs peut se formuler selon :

$$\min_{x \in X} F(x) = (f_1(x), \dots, f_{n_{obj}}(x)) \quad (4.19)$$

où $x = [x_1 \ \dots \ x_{n_{var}}]^T \in X \subset \mathbb{R}^{n_{var}}$ est le vecteur de décision et $f_i(x) \in \mathbb{R}$, $i \in 1, \dots, n_{obj}$ sont les objectifs à minimiser. L'optimisation multi-objectifs consiste donc à optimiser simultanément ces différents critères qui peuvent être contradictoires afin de proposer les solutions représentant les meilleurs compromis entre toutes ces fonctions. Ainsi la notion de solution optimale unique à retenir de l'optimisation mono-objectif disparaît au profit de l'optimalité au sens de Pareto [Van Veldhuizen, 1999]. Cette dernière fournit une surface constituée d'un ensemble de solutions correspondant aux meilleurs compromis possibles pour résoudre le problème.

Les problèmes multi-objectifs ont pour particularité de ne pas présenter une relation d'ordre totale entre les solutions c'est-à-dire qu'il n'est pas possible de trier simplement en absolu les solutions. Puisque plusieurs critères sont retenus, une solution peut présenter de meilleures performances sur certains objectifs, mais non sur l'ensemble des objectifs. De plus, la nature potentiellement conflictuelle des critères ne permet pas d'aboutir à une solution unique qui procure simultanément la solution optimale pour l'ensemble des fonctions à optimiser. Ainsi, le concept de solution optimale reste à formaliser dans ce contexte multi-objectifs. Pour ce faire, il est tout d'abord nécessaire d'établir une relation d'ordre entre les éléments afin d'identifier les meilleures solutions. La notion d'optimalité au sens de Pareto est basée sur la relation de dominance dont les concepts sont présentés par les définitions suivantes.

Définition 4.5 : Dominance de Pareto

Soient deux vecteurs $x \in X$ et $y \in X$ solutions candidates d'un problème multi-objectifs (4.19) donné, alors la solution x domine au sens de Pareto la solution y si :

$$x < y := \begin{cases} \forall i \in 1, \dots, n_{obj}, f_i(x) \leq f_i(y) \\ \exists i \in 1, \dots, n_{obj}, f_i(x) < f_i(y) \end{cases} \quad (4.20)$$

Définition 4.6 : Optimalité de Pareto

Soit un vecteur $x \in X$ solution d'un problème multi-objectifs (4.19) donné, alors x est une solution optimale au sens de Pareto si elle n'est dominée par aucune autre c'est-à-dire :

$$\nexists y \in X, y < x \quad (4.21)$$

Une solution optimale au sens de Pareto constitue en d'autres termes un compromis, c'est-à-dire qu'aucune amélioration ne peut être faite sur un objectif sans la dégradation d'au moins un autre objectif. Dans le cadre multi-objectifs, l'optimisation conduit à un ensemble de solutions optimales qui contient donc toutes les solutions non dominées du problème. Ces éléments, constituant les meilleurs compromis, sont référés dans l'espace de recherche comme l'ensemble optimal de Pareto ou selon leur image par la fonction multi-objectifs comme front de Pareto ou surface des compromis. La figure 4.14 illustre ces concepts pour une fonction bi-objectifs.

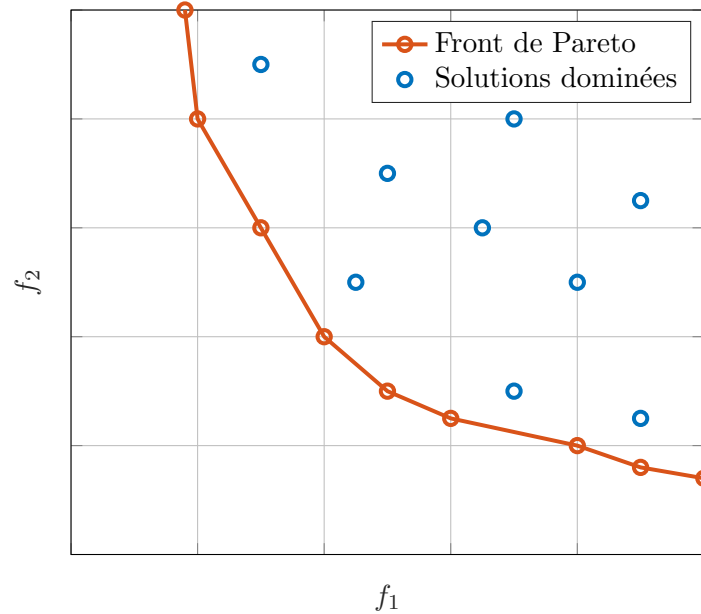


FIGURE 4.14 – Front de Pareto

Définition 4.7 : Ensemble optimal et front de Pareto

Soit un problème d'optimisation multi-objectifs (4.19) donné, l'ensemble optimal de Pareto regroupe les solutions optimales selon la définition :

$$X^* := \{x \in X \mid \nexists y \in X, y < x\} \quad (4.22)$$

Cet ensemble est associé au front de Pareto qui est son image dans l'espace des critères défini tel que :

$$F(X^*) := \{F(x) \in \mathbb{R}^{n_{obj}} \mid x \in X^*\} \quad (4.23)$$

4.4.2 Formulation multi-objectifs de l'apprentissage de contrôleurs robustes

Le problème d'apprentissage d'un contrôleur robuste, vu comme un problème d'optimisation multi-objectifs, consiste tout comme l'optimisation mono-objectif en un problème de recherche des coefficients optimaux du contrôleur, qui sont les poids (matrices de poids et biais) du réseau de neurones. La structure du contrôleur est imposée comme étant un SSNN et l'ensemble de ses poids (matrice de poids et biais) sont regroupés dans le vecteur \mathcal{W} qui constitue donc le vecteur de décision du problème.

D'après les concepts d'optimisation multi-objectifs introduits ci-dessus, nous proposons maintenant de formuler l'apprentissage du contrôleur robuste selon le problème de minimisation des deux critères suivants :

$$\min_{\mathcal{W}} F(\mathcal{W}) = (f_\epsilon(\mathcal{W}), f_\gamma(\mathcal{W})) \quad (4.24)$$

où

- f_ϵ est un critère de performance d'imitation, c'est-à-dire mesurant l'écart de régression temporelle entre les données simulées et celles du jeu de données Z^{dt} . Cette fonction s'assimile à l'indice de performance utilisé dans l'apprentissage mono-objectif par les méthodes de *backpropagation* (voir section 1.3.1). Ainsi, typiquement, le critère retenu correspond à l'erreur quadratique moyenne $J_{mse}(\mathcal{W}, Z^{dt})$ définie par l'équation (1.15) du chapitre 1.
- f_γ est un critère de robustesse de stabilité qui caractérise la boucle d'asservissement. La fonction peut alors exploiter les marges de stabilité de SSNN définies dans la section 4.3.2. Pour ce faire, l'étude est menée sur un modèle G^{NN} du système qui est préalablement établi par exemple via une identification neuronale.

Ce problème d'optimisation s'apparente au problème de compromis entre les performances et la robustesse, inhérent à tout problème d'automatique.

Finalement, en respectant cette expression d'optimisation bi-objectifs, le problème d'identification de contrôleur robuste consiste à déterminer le contrôleur K^{NN} de type SSNN dont la recherche des poids \mathcal{W} peut être formulée à l'aide du problème d'optimisation suivant :

$$\begin{aligned} \min_{\mathcal{W}} F(\mathcal{W}) &= (f_\epsilon(\mathcal{W}), f_\gamma(\mathcal{W})) \\ f_\epsilon(\mathcal{W}) &= J_{mse}(\mathcal{W}, Z^{dt}) \\ f_\gamma(\mathcal{W}) &= \gamma, \gamma^{-1} = \min(\text{DM}_u^{\mathcal{L}_2}, \text{DM}_y^{\mathcal{L}_2}) \end{aligned} \quad (4.25)$$

En explicitant les critères retenus, d'une part le critère de régression temporelle est l'erreur quadratique moyenne de la sortie simulée du contrôleur y_{nn} évaluée pour le jeu de données Z^{dt} telle que :

$$J_{mse}(\mathcal{W}, Z^{dt}) = \frac{1}{2N_{dt}n_y} \sum_{k=1}^N [y(k) - y_{nn}(k)]^T [y(k) - y_{nn}(k)] \quad (4.26)$$

D'autre part, le critère de stabilité de l'asservissement du système G^{NN} , considère la marge de disque en entrée et en sortie par le calcul du gain \mathcal{L}_2 des réseaux associés D_u^{NN} et D_y^{NN} conformément à :

$$\gamma = \max\left(\|D_u^{\text{NN}}\|_{\mathcal{L}_2}, \|D_y^{\text{NN}}\|_{\mathcal{L}_2}\right) = \max\left(\left(\text{DM}_u^{\mathcal{L}_2}\right)^{-1}, \left(\text{DM}_y^{\mathcal{L}_2}\right)^{-1}\right) \quad (4.27)$$

Notons que le choix des marges n'est pas restreint à celles retenues ici et que d'autres marges de stabilité pourraient être prise en considération.

4.4.3 Optimisation multi-objectifs sans gradient

L'apprentissage de contrôleur robuste est un problème d'optimisation difficile, sa résolution implique alors l'emploi d'algorithmes d'optimisation adaptés. Tout d'abord, contrairement à l'erreur de régression temporelle, la considération de la stabilité implique des critères non-dérivables auxquels vont se heurter les techniques d'apprentissage comme celles fondées sur le gradient présentées dans la section 1.3.2. De plus, l'implication de plusieurs objectifs dans l'optimisation, induit l'utilisation d'algorithmes compatibles avec cette formulation. Ainsi, nous nous intéressons ici à une technique appartenant à la classe de méthodes des métaheuristiques qui est l'algorithme génétique.

4.4.3.1 Apprentissage par algorithmes génétiques

L'optimisation stochastique par les algorithmes génétiques vise tout comme les autres métaheuristiques à résoudre les problèmes d'optimisation plus difficiles. Ce type d'algorithme évolutionnaire aborde la résolution du problème d'optimisation à l'aide d'une population de solutions potentielles. Chacune d'elle constitue un individu correspondant à un vecteur de l'espace de recherche. La population souvent définie initialement de façon aléatoire, évolue par la suite progressivement au fur et à mesure des générations. Le renouvellement de la population s'effectue à chaque itération principalement par l'application des opérations de sélection, croisement et mutation. Le processus est arrêté lorsqu'une condition d'arrêt est satisfaite, cette dernière peut se rapporter par exemple au nombre d'itérations, au nombre d'appels à la fonction de coût, à la durée d'exécution, ou encore à un critère de convergence ou de stagnation de la population.

Les algorithmes génétiques se différencient des autres méthodes d'optimisation selon plusieurs particularités :

- Lors du processus d'optimisation, aucun calcul de gradient n'est nécessaire, ainsi la fonction à minimiser peut être éventuellement non-différentiable ou de gradient inconnu. La fonction considérée peut donc être de forme quelconque tant qu'il est possible de l'évaluer. Dans le contexte de contrôleur robuste, ces algorithmes permettent donc de considérer les marges de robustesse qui sont des critères non-dérivables.
- Le caractère stochastique de la méthode permet d'envisager les problèmes pour lesquels il est difficile de déterminer un optimum global de par la présence de nombreux minimums locaux. Par opposition aux procédures déterministes qui exploitent une règle de transition fixe prédéterminée, les procédures stochastiques permettent à l'algorithme de faire face aux fonctions de coût de paysage modal plus complexe en lui fournissant une perspective globale de recherche.
- Le problème d'optimisation est abordé par une approche par population, ainsi plusieurs solutions sont utilisées à l'inverse de la mise à jour d'une solution à chaque itération. L'utilisation d'une population présente notamment pour avantage de trouver plusieurs solutions ce qui facilite la résolution de problème multi-objectifs comme c'est le cas dans nos travaux. Néanmoins, le temps de calcul peut être conséquent, puisqu'il faut évaluer la fonction de coût un nombre de fois égal à la taille de la population à chaque itération.
- Les principaux inconvénients bien connus de la méthode portent sur l'absence de garantie de convergence vers l'optimum global, en conséquence la pertinence des solutions trouvées doit être analysée. De plus, les algorithmes peuvent présenter une vitesse de convergence lente et un temps de calcul important.

L'optimisation par algorithmes génétiques s'avère intéressante pour les problèmes multi-objectifs et non-différentiables, de plus, leur emploi est généralement efficace pour les problèmes de commande comme cela est exposé dans [Feyel, 2017]. Les algorithmes génétiques apparaissent donc ici adaptés à l'apprentissage de contrôleurs avec prise en compte de la stabilité sous une formulation multi-objectifs.

4.4.3.2 Formulation multi-objectifs

La complexité de l'entraînement de contrôleurs robustes par imitation peut être mieux gérée en cherchant sa formulation en tant que problème d'optimisation multi-objectifs. En effet, une

difficulté rencontrée en optimisation mono-objectif est de formuler le problème sous forme d'une équation unique à l'aide d'éventuelles fonctions de pénalité. Si plusieurs critères interviennent, alors il convient de leur assigner des pondérations adéquates qui auront une influence sur la solution obtenue et la convergence de l'algorithme. L'optimisation multi-objectifs ne nécessite donc pas de réglage au niveau de la fonction de coût et octroie ces degrés de liberté qui peuvent parfois manquer en mono-objectif.

La formulation retenue dans nos travaux constitue un problème d'optimisation bi-objectif. Un nombre limité à deux objectifs permet d'envisager pleinement cette méthode qui à l'inverse peut trouver ses limites de résolution lorsque de nombreux objectifs sont impliqués. L'interprétation des résultats se voit également facilitée quand le nombre d'objectifs reste raisonnable.

L'implémentation de l'algorithme génétique dans une version multi-objectifs est réalisée à l'aide de la fonction *gamultiobj* disponible dans la *Global Optimization Toolbox* de MATLAB. Cet outil est fondé sur un algorithme génétique multi-objectifs élitiste contrôlé résultant des travaux de [Deb *et al.*, 2002, Deb, 2011]. Lors de la mise à jour de la population, la génération suivante est établie selon deux critères : un rang de dominance qui est attribué aux individus en utilisant les fonctions d'objectifs et une mesure de distance des individus entre eux de la génération actuelle. La seconde contribution permet de favoriser et de contrôler la diversité de la population pour la convergence vers le front de Pareto.

4.4.3.3 Méthodologie d'implémentation

Nous cherchons à résoudre le problème d'optimisation multi-objectifs défini par l'équation (4.25). La structure du contrôleur K^{NN} est imposée comme étant un SSNN à L couches cachées, ainsi les paramètres du problème d'optimisation sont regroupés dans le vecteur de décision donné par :

$$\mathcal{W} = \text{vec}\left(A, B, C, D, W_x, b_x, b_y, W_1^x, W_1^u, b_1, \dots, W_L, b_L\right) \quad (4.28)$$

où vec est la fonction de vectorisation qui à une ou plusieurs matrices associe le vecteur de leurs éléments terme à terme concaténés. La dimension du vecteur des poids est conditionnée par les hyper-paramètres du réseau, mais également par sa topologie qui peut imposer des matrices creuses par définition (voir par exemple celles du SSNNOE (1.33)).

Comme il a déjà été évoqué précédemment, la résolution du problème multi-objectifs est effectuée à l'aide de la fonction *gamultiobj* de MATLAB. Cet algorithme, comme certains autres, exploite une mesure de distance entre les individus, en plus de la dominance, afin de contrôler la diversité de la population. Cette mesure de distance s'effectue au choix, dans l'espace des objectifs ou dans l'espace de recherche. Cette première possibilité est apparue plus efficace pour nos travaux, néanmoins, elle implique une certaine sensibilité dans la formulation des critères. Le problème d'optimisation (4.25) est formulée selon deux objectifs d'ordre de grandeur sensiblement différents. Afin d'améliorer les propriétés numériques, le problème est résolu par la formulation alternative suivante :

$$\begin{aligned} \min_{\mathcal{W}} F'(\mathcal{W}) &= (f'_\epsilon(\mathcal{W}), f'_\gamma(\mathcal{W})) \\ f'_\epsilon(\mathcal{W}) &= \log_{10}(J_{mse}(\mathcal{W}, Z^{dt})) \\ f'_\gamma(\mathcal{W}) &= \frac{-1}{\gamma}, \gamma^{-1} = \min(DM_u^{\mathcal{L}_2}, DM_y^{\mathcal{L}_2}) \end{aligned} \quad (4.29)$$

La formulation de $f'_\gamma(\mathcal{W})$ à l'aide d'une valeur négative laisse la possibilité d'ajouter un critère de stabilisation défini pour les valeurs positives comme c'est le cas dans [Feyel, 2017] et dont le but est d'aider à la convergence de l'algorithme. Nous pouvons imaginer par exemple qu'une première analyse peu coûteuse en temps est effectuée et conditionne une seconde analyse plus rigoureuse, mais qui nécessite plus de temps de calcul. Ainsi, dans le cas de nos travaux, une première analyse des valeurs propres de la partie linéaire du système pourrait permettre de s'abstenir, pour les cas les moins stables, du calcul plus coûteux du gain \mathcal{L}_2 du système, même si cette approche n'est pas rigoureuse.

Finalement, nous cherchons à résoudre le problème d'optimisation (4.29) où la variable de décision \mathcal{W} est définie en (4.28) et où l'évaluation de la fonction $F'(\mathcal{W})$ consiste à réaliser les étapes suivantes :

1. Construction du réseau neuronal K^{NN} du contrôleur à partir du vecteur de paramètres \mathcal{W} (le réseau comporte également le système de retards appliqué à l'entrée via (3.49)).
2. Calcul du critère $f_\epsilon(\mathcal{W}) = J_{mse}(\mathcal{W}, Z^{dt})$:
 - Simulation temporelle du réseau K^{NN} selon le signal d'entrée $u(k)$ du jeu de données d'apprentissage Z^{dt} afin de générer la sortie estimée y_{nn} .
 - Évaluation du critère de performance d'imitation $J_{mse}(\mathcal{W}, Z^{dt})$ à l'aide de (4.26) appliqué aux données normalisées.
3. Calcul du critère $f_\gamma(\mathcal{W}) = \gamma$, $\gamma^{-1} = \min(DM_u^{\mathcal{L}_2}, DM_y^{\mathcal{L}_2})$:
 - Construction des réseaux neuronaux associés à l'analyse des marges de stabilité à partir de K^{NN} et du modèle du système G^{NN} (ici D_u^{NN} et D_y^{NN} obtenus selon le tableau 4.6).
 - Évaluation du gain \mathcal{L}_2 de chaque réseau par une transformation en LPV-LFT (voir section 4.3.3.2).
 - Définition du critère de robustesse γ selon l'équation (4.27)
4. Évaluation de $F'(\mathcal{W}) = \left(\log_{10}(f_\epsilon(\mathcal{W})), \frac{-1}{f_\gamma(\mathcal{W})} \right)$

4.4.3.4 Techniques d'amélioration de la convergence pour l'apprentissage neuronal

Les inconvénients observés pour les algorithmes génétiques mono-objectifs restent vrais en multi-objectifs. La méthode d'optimisation ne garantit pas de trouver un optimum, même local et la convergence est souvent plus lente. De plus, le temps de calcul peut être important du fait de l'évaluation de la fonction de coût pour chaque individu à chaque itération, même si la parallélisation de la charge de calcul permet de le réduire déjà fortement. Pour surmonter les points faibles de l'approche, nous choisissons de mettre en place les axes d'amélioration présentés ci-dessous ce qui permet d'accroître sensiblement la rapidité de convergence du processus d'optimisation et la performance des résultats obtenus.

Transformation de l'espace de recherche. L'objectif de cette transformation est d'améliorer la sensibilité de l'algorithme en imposant à toutes les valeurs de l'espace de recherche d'avoir un même ordre de grandeur ce qui confère aux valeurs de l'espace initial la même importance. La transformation s'effectue par un changement de variable par l'application d'une fonction comparable à la fonction logarithme. Afin d'également traiter les variables négatives, nous utilisons

tout comme dans [Feyel, 2017], la fonction sinus hyperbolique sh_{10} ainsi que sa réciproque ash_{10} définies comme suit :

$$\text{sh}_{10}(x) = \frac{10^x - 10^{-x}}{2} \quad (4.30)$$

$$\text{ash}_{10}(x) = \log_{10} \left(x + \sqrt{1 + x^2} \right) \quad (4.31)$$

La transformation de l'espace X de recherche initial en l'espace X' utilisé par l'algorithme d'optimisation est effectuée de la façon suivante :

$$\begin{aligned} x \in X & \xrightarrow{x' = \text{ash}_{10}(Mx)} x' \in X' \\ x' \in X' & \xrightarrow{x = \text{sh}_{10}(x')/M} x \in X \end{aligned} \quad (4.32)$$

où M est un paramètre de réglage, choisi dans nos travaux tel que $M = 5000$.

Initialisation de la population. La phase de création de la population initiale de l'algorithme génétique consiste généralement en une génération aléatoire de solutions dans les limites inférieures et supérieures de l'espace de recherche. Néanmoins, comme il est indiqué dans [Deb, 2011], si pour un problème donné, la connaissance de certaines bonnes solutions est disponible, il est préférable d'utiliser cette information pour l'initialisation de la population. Cette pratique peut se révéler déterminante lors de résolutions de problème d'optimisation complexe, et permet souvent au minimum d'accélérer la convergence de la recherche.

Dans le contexte d'identification de systèmes dynamiques par un modèle neuronal, l'optimisation des poids du réseau par des algorithmes génétiques est particulièrement lente et la convergence vers de bonnes performances de régression est difficile. D'un autre côté, les techniques de descente de gradient offrent de bonnes performances concernant l'erreur de régression temporelle, mais ne permettent pas de prendre en considération d'autres critères d'optimisation. Nous choisissons dans ces travaux, d'initialiser une partie de la population de l'algorithme à l'aide de plusieurs pré-entraînements réalisés avec des méthodes de rétropropagation du gradient comme par exemple celle de Levenberg-Marquardt (LM). Afin que cette population personnalisée d'initialisation soit suffisamment diversifiée, les critères d'arrêts de l'algorithme d'apprentissage peuvent être ajustés d'un pré-entraînement à l'autre, c'est-à-dire d'un individu à l'autre. Nous pouvons par exemple, sélectionner des valeurs de coûts minimales d'arrêt différents ou un nombre d'itérations maximales plus ou moins important dans le but de générer un ensemble d'individus de différents niveaux innés de spécialisation.

4.4.4 Application

Nous reprenons dans cette partie de mise en pratique, le cas d'étude du chariot mobile avec pendule déjà évoqué dans les sections 4.2.3 et 4.3.4. L'objectif est maintenant de réaliser l'apprentissage du contrôleur par imitation en prenant explicitement en considération les marges de stabilité par la formulation multi-objectifs.

Le contrôleur à imiter est K_{syn} le contrôleur à trois degrés de liberté issu d'une synthèse \mathcal{H}_∞ (figure 4.5), les données d'apprentissage restent donc inchangées : la première base de données est Z_1^{dt} générée par l'excitation de la consigne (schéma 4.2) et la seconde Z_2^{dt} par l'excitation complémentaire de la perturbation d'entrée (schéma 4.3). Ainsi, deux apprentissages indépendants

sont réalisés, ils aboutissent respectivement dans le premier et dans le second cas à un ensemble de contrôleurs optimal de Pareto K_1^* et K_2^* . Le tableau 4.10 récapitule les structures neuronales utilisées ainsi que le nombre d'itération et la durée de l'algorithme avant son arrêt. L'algorithme génétique exploite une population de 32 individus dont les limites de l'espace de recherche sont fixées aux valeurs ± 1000 . De plus, afin d'accélérer la convergence de l'algorithme vers de faibles erreurs de régression, une part d'individus, ici au nombre de 10, n'a pas été initialisée aléatoirement, mais à l'aide d'un pré-entraînement réalisée par l'algorithme LM. Les coûts minimums d'arrêt utilisés ont été choisis afin d'arrêter l'apprentissage plus ou moins prématurément pour garantir la diversité de la population.

TABLE 4.10 – Résultats de l'apprentissage multi-objectifs des contrôleurs neuronaux pour les deux bases de données

	Structure						Résultats	
	topologie	n_{in}	n_{out}	σ_1	n_1	$n_{\gamma\gamma}$	iter.	durée
K_1^*	SSNNOE	5	6	tanh	5	155	28720	3j23h43m
K_2^*	SSNNOE	5	6	tanh	5	155	12140	2j8h56m

Les fronts de Pareto associés aux ensembles optimaux de contrôleurs sont illustrés par la figure 4.15. Ces courbes, qui ont pour ce cas d'étude une apparence concave selon l'échelle logarithmique, exposent parfaitement la dualité entre les deux critères d'optimisation. Du côté du front où le critère de robustesse γ est privilégié, les deux entraînements permettent d'avoisiner la valeur minimale admissible puisque par définition γ représente la valeur maximale entre $\|D_u^{NN}\|_{\mathcal{L}_2}$ et $\|D_y^{NN}\|_{\mathcal{L}_2}$ (voir l'équation (4.27)), qui sont toutes deux des grandeurs positives supérieures à $\frac{1}{2}$. La différence se fait néanmoins remarquer en termes de performance, l'erreur de régression est à l'avantage du cas du jeu Z_1^{dt} comme ce fut le cas lors de l'apprentissage en mono-objectif (voir le tableau 4.3).

Un sous-ensemble représentatif des contrôleurs optimaux de Pareto est extrait pour les deux entraînements. Le tableau 4.11 présente les marges de disque et les marges généralisées déduites des contrôleurs avec l'erreur de régression temporelle obtenue. L'examen des marges montre que pour ce cas d'étude, la marge de sortie est pour chacun des points le critère limitant.

Les propriétés de robustesses issues des différents contrôleurs sont maintenant mises à l'épreuve par une série de simulations temporelles regroupée dans les figures 4.16 et 4.17. Les simulations sont réalisées avec le chariot avec pendule décrit par ses équations non-linéaires (4.1). Pour chaque sous-ensemble de contrôleurs, une première simulation est réalisée en régime nominal avec rejet d'une perturbation d'entrée. Une seconde expérience est menée suivant le schéma 4.8 où une variation importante de gain de sortie du système est présente. À partir des résultats d'expérience, nous pouvons énoncer les remarques suivantes :

- Concernant la robustesse des contrôleurs, les comportements observés sont stables pour l'ensemble des contrôleurs et l'ensemble des réponses simulées. Ainsi, la stabilité de l'asservissement est obtenue que ce soit en simulation nominale, lors de la présence d'une perturbation ou d'une variation paramétrique.
- Concernant les performances des contrôleurs, il est important de garder à l'esprit qu'il s'agit d'une performance d'imitation et non de la performance intrinsèque de l'asservissement.

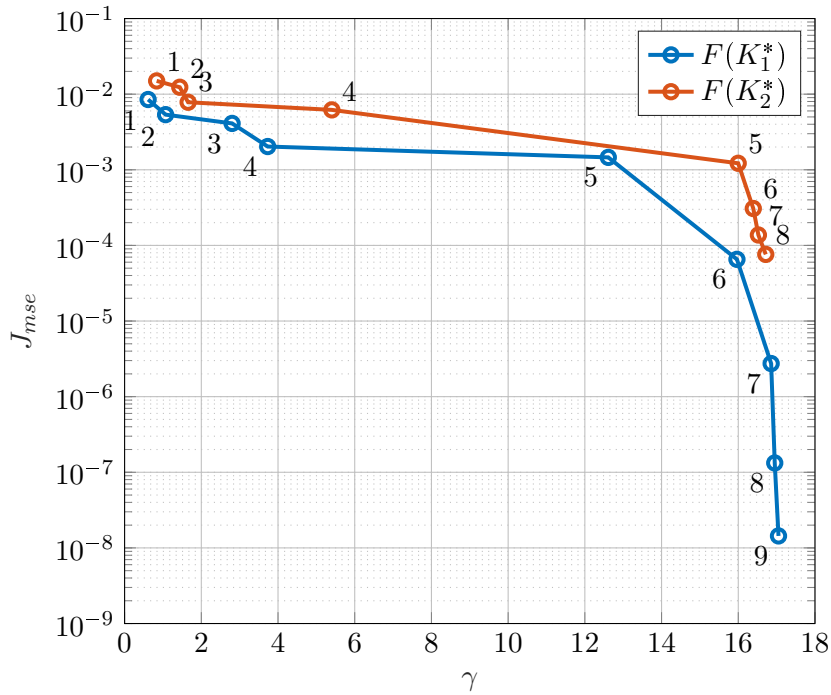
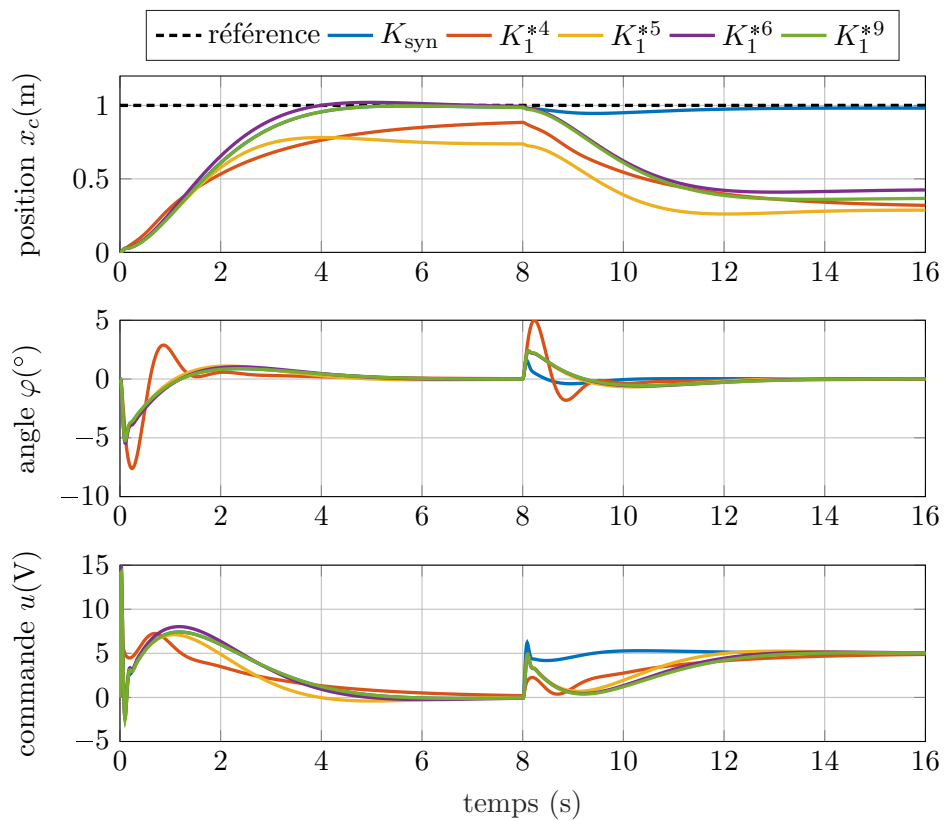


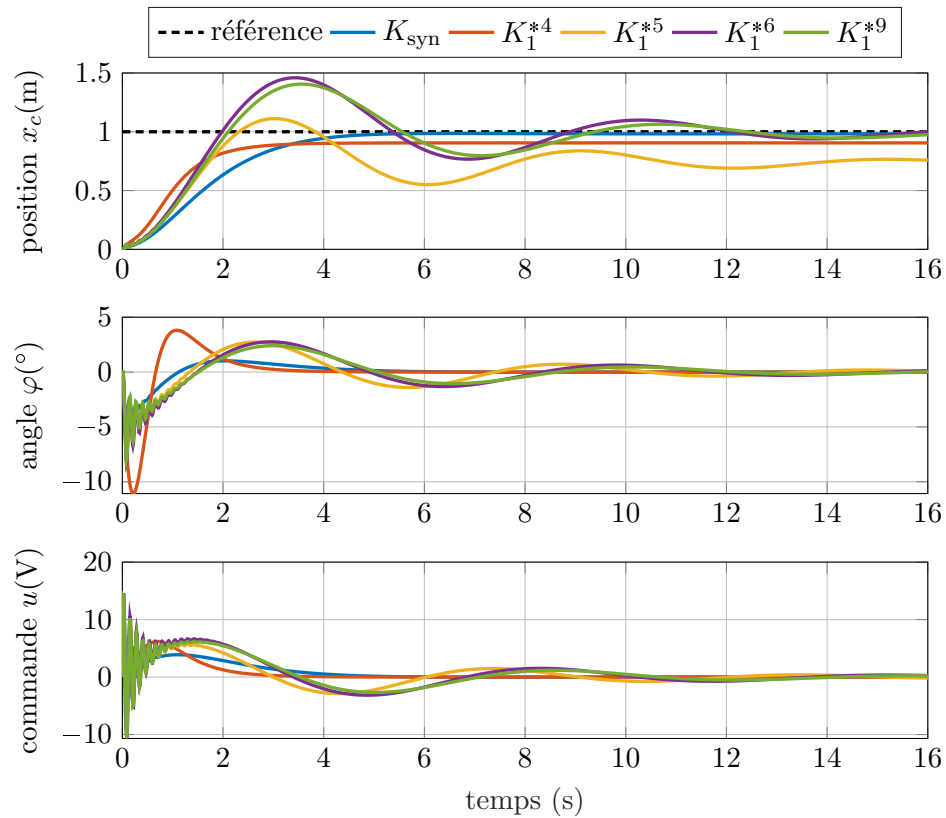
FIGURE 4.15 – Fronts de Pareto des ensembles de contrôleurs neuronaux optimaux

TABLE 4.11 – Marges de stabilité des sous-ensembles de contrôleur optimaux de Pareto

	Marges d'entrée				Marges de sortie				Erreur J_{mse}
	$\gamma^{DM_u^{\mathcal{L}^2}}$	$DM_u^{\mathcal{L}^2}$	GM_u	PM_u	$\gamma^{DM_y^{\mathcal{L}^2}}$	$DM_y^{\mathcal{L}^2}$	GM_y	PM_y	
K_{syn}	1.92	0.52	4.6 dB	29.0°	5.55	0.18	1.6 dB	10.5°	
K_1^{*4}	0.65	1.50	17.0 dB	73.9°	3.73	0.27	2.3 dB	15.3°	2.0e-3
K_1^{*5}	1.43	0.70	6.3 dB	38.4°	12.6	0.08	0.7 dB	4.5°	1.5e-3
K_1^{*6}	1.45	0.69	6.3 dB	38.1°	16.0	0.06	0.5 dB	3.6°	6.5e-5
K_1^{*9}	1.43	0.70	6.4 dB	38.7°	17.0	0.06	0.5 dB	3.4°	1.4e-8
K_2^{*3}	0.55	1.82	26.6 dB	84.7°	1.66	0.60	5.4 dB	33.6°	7.8e-3
K_2^{*4}	0.72	1.38	14.7 dB	69.2°	5.40	0.19	1.6 dB	10.6°	6.2e-3
K_2^{*5}	1.93	0.52	4.6 dB	29.0°	16.0	0.06	0.5 dB	3.6°	1.2e-3
K_2^{*8}	1.93	0.52	4.6 dB	29.0°	16.7	0.06	0.5 dB	3.4°	7.6e-5

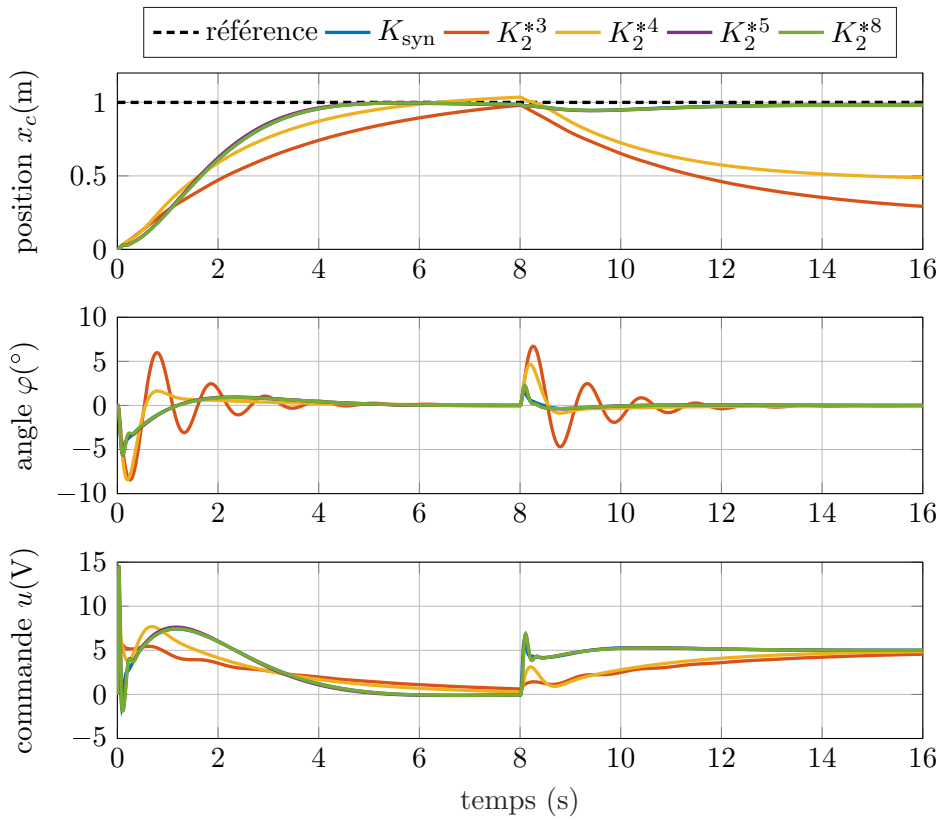


(a) Simulation dans le cas nominal avec rejet d'une perturbation d'entrée de -5V à partir de 8s

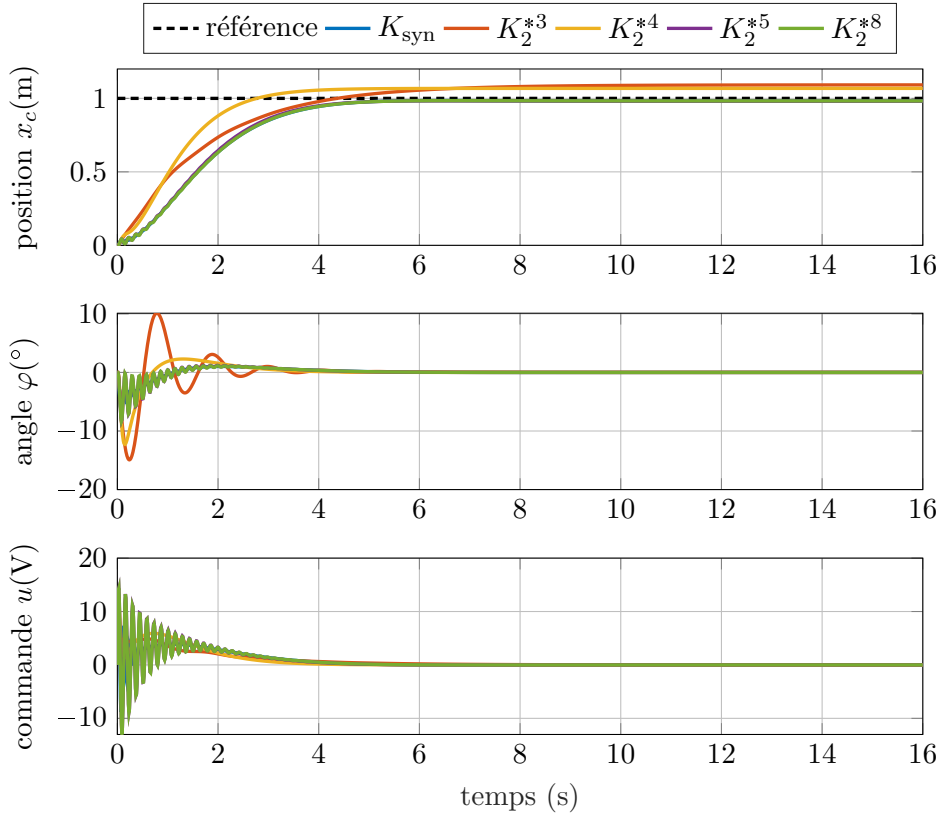


(b) Simulation dans le cas d'une variation de gain $\Lambda_y = 2$ et sans perturbation

FIGURE 4.16 – Simulation d'un sous-ensemble des contrôleurs optimaux de Pareto (K_1^*)



(a) Simulation dans le cas nominal avec rejet d'une perturbation d'entrée de $-5V$ à partir de 8s



(b) Simulation dans le cas d'une variation de gain $\Lambda_y = 2$ et sans perturbation

FIGURE 4.17 – Simulation d'un sous-ensemble des contrôleurs optimaux de Pareto (K_2^*)

L'analyse des réponses temporelles doit donc s'effectuer en comparaison avec la réponse du contrôleur d'origine K_{syn} . De ce point de vue, les contrôleurs issus de l'apprentissage sur les données Z_2^{dt} présentent, comme dans les simulations de la section 4.2.3, un comportement se rapprochant le plus de celui d'origine. Ceci se vérifie y compris lorsque des perturbations ou variations interviennent, notamment pour les contrôleurs K_2^{*5} et K_2^{*8} .

- Le cas de rejet de perturbation offre un cas d'étude intéressant relatif à une propriété propre au contrôleur d'origine. En réponse à la perturbation, les contrôleurs de l'ensemble K_1^* occasionnent une erreur statique importante de suivi de consigne de position. Ce phénomène témoigne de l'incapacité à reproduire le comportement intégrateur du contrôleur d'origine qui n'a pas été suffisamment représenté dans les données Z_1^{dt} . De part l'excitation supplémentaire présente dans la base de données Z_2^{dt} , une partie des contrôleurs de l'ensemble K_2^* manifeste une erreur de suivi nulle. L'intégrateur est donc correctement assimilé dans ce cas, mais logiquement réservé aux contrôleurs les plus performants et donc les moins robustes.

Pour conclure, les comportements observés des différents asservissements sont en accord avec les marges de stabilité et les erreurs d'apprentissage du tableau 4.11. Ces différentes simulations le long du front de Pareto valident alors temporellement l'indicateur de robustesse. De plus, ceci confirme la démarche mise en place et l'intérêt de cette méthode d'apprentissage avec prise en compte de la stabilité.

SECTION 4.5

Conclusions et perspectives

4.5.1 Conclusions

Nous avons développé dans ce chapitre des stratégies d'apprentissage de systèmes de contrôle robustes par imitation de comportements contenus dans une base de données. Nous avons dans un premier temps mis en avant la problématique de la stabilité qui survient lors de l'apprentissage de contrôleurs intervenant dans une boucle de rétroaction. En effet, la loi de commande obtenue peut alors rapidement exposer des comportements instables du fait des erreurs résiduelles d'apprentissages et des écarts inévitables de l'évolution du système entre la réalité et le scénario d'entraînement. Par un exemple académique, nous avons pu démontrer l'importance des marges de robustesse, ainsi que le besoin de disposer de méthodes permettant de les quantifier et de les ajuster.

Une première contribution de ce chapitre a été de proposer une méthode pour évaluer les marges de stabilité d'un asservissement neuronal. Nous avons tout d'abord défini le concept de marge de robustesse dans le contexte des SSNN selon différentes approches. Par la suite, nous avons présenté une technique afin d'évaluer numériquement ces marges. Une étape préliminaire consiste à concevoir le contrôleur neuronal ainsi qu'à identifier un modèle neuronal du système à partir des données. L'étape d'estimation des marges de stabilité est par la suite effectuée sur le SSNN global de la boucle fermée à l'aide des méthodes d'analyse du modèle LPV-LFT équivalent.

En second lieu, nous avons également contribué à la mise en place d'une méthode d'apprentissage qui tient directement compte des marges de stabilité lors de l'optimisation du contrôleur neuronal. Du fait de sa conception même, cette stratégie peut par exemple être mise en œuvre dans le but de renforcer les propriétés de robustesse d'un contrôleur existant. Le problème

d'apprentissage est envisagé selon une formulation multi-objectifs afin de concilier les performances d'imitation du contrôleur et la robustesse de stabilité de la boucle d'asservissement. De par la nature du problème et du caractère non-différentiable du calcul des marges de stabilité, l'apprentissage est réalisé à l'aide d'algorithmes génétiques multi-objectifs. Par cette méthode, l'apprentissage abouti à un ensemble de contrôleurs réparti sur le front de Pareto pour lequel l'automaticien peut déterminer d'un point de vue plus haut-niveau ses exigences de performance et de robustesse. Un avantage de cette formulation est notamment de pouvoir se prémunir des erreurs d'identifications et donc de la représentativité de l'indicateur de robustesse en choisissant des marges de stabilités plus importantes. Ce chapitre, nous a également permis par la mise en application de souligner l'importance des données d'apprentissage et des informations qu'elles doivent contenir afin de caractériser l'intégralité des propriétés du contrôleur qui doivent être assimilées notamment d'un point de vue fréquentiel.

4.5.2 Perspectives

Comme nous l'avons évoqué, les marges de stabilité sont évaluées sur la boucle d'asservissement identifiée, c'est-à-dire à partir du contrôleur établis, mais également du modèle du système. Néanmoins, une perspective de travail, serait de tenir compte de l'erreur d'apprentissage du système dans l'exigence des marges de stabilité. En effet, si cette erreur d'identification est importante et que les marges sont tout de même évaluées avec le modèle associé, il est nécessaire d'inclure l'erreur dans le raisonnement puisqu'elle influencera directement le niveau de confiance qui peut être accordé aux indicateurs de robustesse. En complément, ces indicateurs pourraient être étudiés davantage comme par exemple en établissant leurs rapports avec la généralisation. En effet, dans le cadre d'un apprentissage de contrôleurs par un réseau de neurones récurrent, ces indicateurs pourraient éventuellement témoigner des capacités de généralisation du modèle une fois introduit dans la boucle d'asservissement.

Une autre piste de travail concernerait l'amélioration des méthodes de résolutions numériques. D'une part, les méthodes d'analyses développées dans ce chapitre présentent des limitations théoriques comme nous l'avons déjà évoqué dans les chapitres précédents, néanmoins, dans la pratique la capacité de résolution des problèmes LMI est également un aspect fondamental. En effet, de nombreuses difficultés numériques sont rencontrées lors de l'analyse de réseaux de neurones, ainsi, il serait important de développer des techniques de résolution plus robustes numériquement parlant. D'autre part, d'autres méthodes d'optimisation multi-objectifs et sans gradient pourraient être employées et comparées aux algorithmes utilisés dans nos travaux.

Approche multi-modèle pour l'implémentation du contrôleur

Sommaire

5.1	Introduction	120
5.2	Introduction au MMAC	120
5.2.1	Principe	120
5.2.2	Architecture du MMAC	121
5.3	Extension neuronale du MMAC	124
5.3.1	Intérêts de l'approche	124
5.3.1.1	Consolidation de l'apprentissage	124
5.3.1.2	Considération des situations difficiles	124
5.3.2	Architecture du MMAC neuronal (NMMAC)	125
5.3.2.1	Banque de contrôleurs	125
5.3.2.2	Banques modèles-estimateurs et logique de sélection	126
5.4	Étude de la stabilité et réglage dans le cas linéaire	127
5.4.1	État de l'art	127
5.4.2	Analyse du processus de décision incertain	128
5.4.2.1	Structure d'analyse du MMAC	129
5.4.2.2	Domaine de stabilité du MMAC fondé sur la μ -analyse	131
5.4.2.3	Application	132
5.4.3	Réglage de la logique de décision dans le cas linéaire	135
5.4.3.1	Approche <i>Estimator-based MMAC</i>	135
5.4.3.2	Critère caractéristique du réglage du MMAC	136
5.4.3.3	Optimisation du réglage du MMAC	137
5.4.3.4	Application	137
5.4.4	Limitations de l'étude	140
5.5	Conclusions et perspectives	142
5.5.1	Conclusions	142
5.5.2	Perspectives	142

SECTION 5.1

Introduction

Nous présentons dans ce chapitre une méthode d'extension du contrôleur neuronal par une approche multi-modèle. En effet, dans le chapitre 4, nous avons pu définir une mécanique d'apprentissage d'un contrôleur et du modèle du système sujet à l'asservissement. Néanmoins, la mise en place d'un contrôleur unique peut se révéler très complexe lorsque les données d'apprentissages sont par exemple fortement diversifiées et quantitatives ou lorsque de nouvelles données et comportement doivent être assimilés. Dans cette perspective, le contrôle adaptatif multi-modèle (MMAC) permet de représenter un système et sa stratégie de contrôle en décomposant l'espace de fonctionnement en plusieurs sous-espaces. Plusieurs modèles locaux décrivent le système global plus complexe et des contrôleurs locaux leur sont associés pour chaque zone de fonctionnement. Nous développons ainsi dans ce chapitre, une méthode multi-modèle neuronale pour laquelle nous dressons des pistes d'études.

Nous introduisons dans la section 5.2, la méthode de contrôle adaptative multi-modèles (MMAC). Dans la section 5.3, nous développons plus en détail l'intérêt d'une telle approche dans nos travaux et nous présentons le déploiement des modèles et contrôleurs neuronaux conformément à cette stratégie. Nous proposons finalement dans la section 5.4, une méthode d'étude et de réglage dans le cas linéaire.

SECTION 5.2

Introduction au MMAC**5.2.1 Principe**

L'approche multi-modèle offre une solution mature à de nombreux problèmes confrontés à des processus réels qui peuvent être utilisés dans une large gamme de conditions d'opération. La modélisation, l'identification ou le contrôle de tels processus ne sont pas des tâches faciles de par leur complexité intrinsèque. Une solution souvent judicieuse est de décomposer le problème initial en sous-problèmes se rapprochant des problèmes connus dans une stratégie de *diviser pour mieux régner*.

Basée sur ce principe de décomposition, la méthode de contrôle adaptative multi-modèle notée MMAC (*Multiple Model Adaptive Control*) considère que le système à contrôler peut être représenté par un nombre fini de sous-modèles valables dans certaines conditions d'utilisation. Des contrôleurs sont respectivement conçus pour chaque modèle et enfin, une logique de sélection décide à chaque instant quels sont le ou les contrôleurs devant être impliqués pour le système actuel.

L'approche MMAC est utilisée pour asservir les systèmes avec de grandes incertitudes paramétriques ou avec un fort comportement non-linéaire. Le problème à l'étude est qu'un seul contrôleur fixe ne peut pas stabiliser toutes les configurations possibles et répondre à certaines exigences de performances. Le système est décrit par une combinaison de modèles locaux où chaque modèle est valable dans une région d'opération particulière. L'architecture du MMAC est typiquement séparée en deux sous-systèmes autonomes :

- le processus de contrôle (banque de contrôleurs non-adaptatifs)
- le processus d'identification et de décision (banque d'estimateurs et logique de sélection)

Premièrement, l'entité de contrôle consiste en une banque de contrôleurs non-adaptatifs conçus pour chaque modèle. La seconde entité constitue le mécanisme d'adaptation, elle est composée d'une banque d'observateurs dérivant de chaque modèle et d'une logique de sélection qui se base sur les erreurs d'estimation des sorties. Fréquemment, ce second sous-système, emploie le MMAE (pour *Multiple Model Adaptive Estimation*) qui consiste par exemple en une banque de filtres de Kalman (voir [Ducard, 2009, Fekri *et al.*, 2006, Sims *et al.*, 1969]). Néanmoins d'autres types d'observateurs peuvent être employés comme des observateurs de Luenberger comme nous le présenterons plus loin dans la section 5.4.3. Pour ces approches, les modèles sont supposés pour l'analyse linéaires ou linéarisés autour de points de fonctionnement.

Dans le cadre de nos travaux, une version non-linéaire du MMAC est utilisée, ainsi une déclinaison non-linéaire des deux processus est présentée dans ce chapitre. Le MMAE original est remanié en une variante étendue (*Extended MMAE*) par l'utilisation de filtres de Kalman étendus (EKF) dans le rôle d'estimateurs non-linéaires. Cette extension non-linéaire est par exemple présente dans les travaux de [Ducard, 2009] ou suggéré dans ceux de [Fekri, 2005].

5.2.2 Architecture du MMAC

Un schéma global de l'architecture du MMAC est présenté par la figure 5.1. Sur ce schéma, l'asservissement du système G est assuré par la banque de contrôleurs $K_i, i \in \{1, \dots, N\}$ générant les commandes notés $U = [u_1 \dots u_N]^T$ et la banque des estimateurs $E_i, i \in \{1, \dots, N\}$ qui fournit les sorties estimées $\hat{Y} = [\hat{y}_1 \dots \hat{y}_N]^T$. Les différents intervenants de cette stratégie de contrôle sont détaillés ci-dessous.

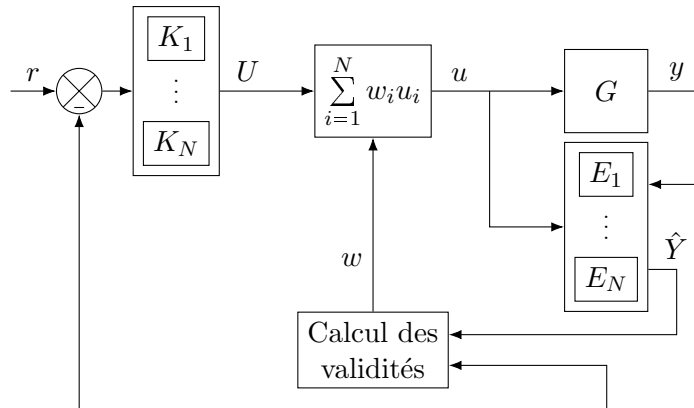


FIGURE 5.1 – Architecture du MMAC

Considérons le système G sujet à des variations paramétriques θ non mesurées, prenant des valeurs dans l'ensemble compact $\Omega \subset \mathbb{R}^{n_\theta}$. Le système G est supposé de façon générale multivariable (MIMO) et de la forme :

$$G(\theta) := \begin{cases} x(k+1) = f(x(k), u(k), \theta(k)) \\ y(k) = g(x(k), \theta(k)) \end{cases} \quad (5.1)$$

où $x(k) \in \mathbb{R}^{n_x}$ dénote l'état du système, $u(k) \in \mathbb{R}^{n_u}$ est le vecteur des commandes et $y(k) \in \mathbb{R}^{n_y}$

le vecteur des sorties mesurées.

Premièrement, considérons que N modèles soient nécessaires pour couvrir l'ensemble des incertitudes du système réel. Soit un ensemble fini de valeurs de paramètres candidates $\Theta := \{\Theta_1, \dots, \Theta_N\}$. À chaque configuration nominale $\Theta_i \in \Theta \subset \mathbb{R}^{n_\Theta}$, $i = \{1, \dots, N\}$, un modèle nominal correspondant est obtenu $M_i := G(\Theta_i)$. Le terme Θ_i peut par exemple représenter une variation brutale de la valeur d'un paramètre physique due à un défaut ou à une valeur spécifique d'un paramètre non mesuré, etc.

Une fois la banque de modèles définie, une banque de contrôleurs doit lui être associée. Pour chaque $i^{\text{ème}}$ modèle local, un $i^{\text{ème}}$ contrôleur local est conçu afin de garantir des exigences de robustesse de stabilité et de performances. La banque de contrôleurs est composée de N éléments décrits par les représentations d'état suivantes :

$$K_i := \begin{cases} x_i^K(k+1) = f^K(x_i^K(k), r(k) - y(k)) \\ u_i(k) = g^K(x_i^K(k), r(k) - y(k)) \end{cases} \quad (5.2)$$

où $r(k) \in \mathbb{R}^{n_y}$ est la référence à suivre. Afin d'obtenir la loi de contrôle globale, les contrôleurs sont partiellement combinés. La commande finale pour le système réel est une somme pondérée des sorties de chaque contrôleur local :

$$u(k) = \sum_{i=1}^N w_i(k) u_i(k) \quad (5.3)$$

où $w_i(t)$ est la validité ou le poids reflétant l'importance relative de chaque $i^{\text{ème}}$ modèle par rapport au système réel. Ces validités sont attribuées selon des probabilités de façon à ce que les modèles moins probables (c'est-à-dire ressemblant moins au système réel actuel) soient moins pondérés. Cela garantit que les contrôleurs conçus pour des modèles moins probables ont moins d'influence sur la commande résultante. Le vecteur des validités $w \in \mathbb{R}^N$ possède la propriété de convexité suivante :

$$\forall i \in \{1, \dots, N\} w_i(k) \geq 0, \quad \sum_{i=1}^N w_i(k) = 1 \quad (5.4)$$

L'étape suivante de l'approche consiste à déterminer une banque d'estimateurs souvent référée comme MMAE qui est composée de N estimateurs E_i pour chaque modèle nominal M_i . Ces estimateurs génèrent des sorties estimées du système $\hat{y}_i(k)$ en se basant sur les commandes $u(k)$ et les sorties mesurées $y(k)$ du système réel. Ces estimations constitueront la base du calcul des validités.

Le dernier aspect et l'un des plus essentiels du MMAC est la méthodologie qui permet de déterminer la validité ou le poids associé à chaque contrôleur pour former la commande globale. Cette logique doit identifier les contrôleurs qui doivent être impliqués dans la boucle d'asservissement et leurs contributions en fonction de la valeur réelle du paramètre θ . Le calcul de la validité est fondé sur la distance entre la sortie du système et celles des différents estimateurs définis dans la banque. Cette distance appelée erreur résiduelle ou erreur d'estimation est donc définie pour le $i^{\text{ème}}$ modèle comme :

$$\varepsilon_i(k) = y(k) - \hat{y}_i(k) \quad (5.5)$$

La validité normalisée est calculée en temps réel à chaque instant $k \in \mathbb{N}$ par l'estimateur ultérieur de probabilité ou PPE (pour *Posterior Probability Evaluator*) selon l'équation récursive suivante :

$$w_i(k+1) = \frac{\beta_i \exp\left(-\frac{1}{2}\varepsilon_i^T(k)\Gamma_i\varepsilon_i(k)\right) w_i(k)}{\sum_{j=1}^N \beta_j \exp\left(-\frac{1}{2}\varepsilon_j^T(k)\Gamma_j\varepsilon_j(k)\right) w_j(k)} \quad (5.6)$$

Les validités sont ainsi calculées dynamiquement par une formule de validité bayésienne (voir [Ferik et Adeniran, 2017, Sims *et al.*, 1969]) où Γ_i est une matrice de pondération définie-positive à choisir tout comme le coefficient réel positif β_i . Ces paramètres jouent un rôle important dans le réglage du taux de convergence des probabilités $w_i(k)$ et doivent être choisis minutieusement. Leur choix qui pilote la logique bayésienne est souvent associé au réglage des estimateurs, ainsi lors de l'emploi de filtres de Kalman, les paramètres sont désignés à l'aide des matrices de covariance de l'innovation $S_i(k)$ selon :

$$\beta_i(k) = \left[(2\pi)^{n_y} \det(S_i(k)) \right]^{-\frac{1}{2}} \quad (5.7)$$

$$\Gamma_i(k) = S_i^{-1}(k)$$

Remarque 5.1. Dans la pratique, pour éviter que la validité associée à un modèle tende vers zéro et que ce modèle ne soit plus pris en compte dans les évolutions ultérieures de part le calcul récursif, un seuil $\delta > 0$ est utilisé, les probabilités sont réinitialisées à cette valeur et ne peuvent atteindre zéro. Enfin, les poids sont normalisés en excluant les modèles ayant atteint le seuil dans le calcul effectif de la commande. De façon détaillée, les validités sont déterminées selon trois étapes :

1. Mise à jour récursive :

$$w'_i(k+1) = \frac{\beta_i \exp\left(-\frac{1}{2}\varepsilon_i^T(k)\Gamma_i\varepsilon_i(k)\right) w'_i(k)}{\sum_{j=1}^M \beta_j \exp\left(-\frac{1}{2}\varepsilon_j^T(k)\Gamma_j\varepsilon_j(k)\right) w'_j(k)} \quad (5.8)$$

2. Minoration excluant zéro :

$$w'_i(k) = \begin{cases} w'_i(k) & \text{si } w'_i(k) > \delta \\ \delta & \text{sinon} \end{cases} \quad (5.9)$$

3. Normalisation :

$$w_i(k) = \begin{cases} \frac{w'_i(k)}{\sum_{j \in \mathcal{J}} w'_j(k)} & \text{si } w'_i(k) > \delta \\ 0 & \text{sinon} \end{cases} \quad (5.10)$$

où $\mathcal{J} := \{j \in \{1, \dots, N\} \mid w'_j(k) > \delta\}$

SECTION 5.3

Extension neuronale du MMAC

Dans le cadre de nos travaux, les différents éléments du MMAC sont élaborés à partir de modèles neuronaux issus de plusieurs phases d'apprentissage hors-ligne. Nous présentons dans cette section les intérêts d'une telle approche ainsi qu'une présentation plus détaillée des différentes banques de cette version neuronale de l'approche multi-modèle notée NMMAC (*Neural MMAC*).

5.3.1 Intérêts de l'approche**5.3.1.1 Consolidation de l'apprentissage**

L'un des principaux intérêts, est que l'approche multi-modèle offre une méthode ordonnée permettant d'accumuler des connaissances. En effet, une idée simple consiste à penser que l'intelligence artificielle entraînée s'avérera de plus en plus fiable et pertinente si elle consolide son apprentissage. Lorsque de nouvelles données sont mises à disposition, il s'agit de les apprendre sans pour autant oublier celles déjà connues. Il fait également non-sens de tout réapprendre comme c'est le cas usuel en apprentissage supervisé. Le MMAC propose une architecture relativement flexible où il est possible de rajouter des connaissances (une paire modèle-contrôleur) tout en gardant en mémoire les acquis. Ainsi, il peut être facilement envisagé d'accroître les capacités du contrôleur en considérant une variété de systèmes, de tâches à réaliser ou de stratégies de contrôle. Par exemple, pour une application aéronautique, plusieurs avions pourraient être pris en charge par un même autopilote ou différentes lois de contrôle pourraient être établies à partir de pilotes différents et pour différentes tâches.

De plus, le multi-modèle fournit un cadre d'étude des connaissances du contrôleur global. Cette démarche, donne en effet, la possibilité d'étudier l'importance des structures apprises afin de cibler les plus importantes et d'oublier celles qui ne le sont pas ou qui sont redondantes. Ainsi, les connaissances acquises par le contrôleur intelligent peuvent être maîtrisées au moyen du MMAC via l'ajout ou le retrait de modèle-contrôleur.

Une piste également intéressante est le *transfer learning* qui consiste à transférer les connaissances préalablement acquises pour traiter au mieux de nouvelles données. Cependant, cette méthode seule n'assure en rien la conservation des compétences déjà connues. Dans le cas d'apprentissage de contrôleur, cela est un facteur limitant quant à la garantie de stabilité de la boucle d'asservissement résultante. Le MMAC s'impose donc comme une alternative pour perpétuer l'apprentissage. Néanmoins, le *transfer learning* peut-être envisagé localement pour fournir par exemple, un point de départ dans l'élaboration d'une nouvelle paire modèle-contrôleur.

5.3.1.2 Considération des situations difficiles

Le raisonnement multi-modèle présente également un intérêt pour faire face à des bases de données potentiellement complexes et de grandes dimensions. L'apprentissage d'un jeu de données de trop grande taille peut rencontrer des difficultés de convergence lors de l'apprentissage ou nécessiter un temps de calcul important. De plus, il peut être difficile d'apprendre sur des données relativement complexes et constituées de nombreux éléments distincts. Ainsi, le multi-modèle permet d'aborder la problématique selon une multitude d'apprentissages plus accessibles. L'apprentissage est alors réalisé de façon incrémentale, sur

des jeux de données compartimentés par exemple à l'aide de méthode de *clustering*. De cette manière, divers modèles et contrôleurs peuvent être dédiés à des tâches foncièrement différentes ou pour envisager des variations importantes comme des situations difficiles ou de pannes.

Dans un souci de commande tolérante aux pannes souvent référé FTC (*Fault Tolerant Control*), le MMAC apparaît comme une méthode intuitive et efficace afin de considérer les situations dégradées. Chaque scénario de défaillance ou situation difficile envisagé est décrit par un modèle spécifique puis appairé avec un correcteur *a priori* conçu. Le MMAC est alors le principal élément d'une méthode de détection, isolation de défaut et reconfiguration ou FDIR (*Fault Detection, Isolation and Recovery*), qui permet au système dans un cas de panne de continuer à fonctionner, mais dans un mode dégradé.

Dans un contexte aéronautique, qui sera le cadre d'étude du chapitre 6, une reconfiguration rapide et précise des commandes de vol a une importance primordiale afin d'augmenter la capacité de réaction des avions en présence de graves défaillances de sous-systèmes et de dommages structurels. Le contrôle adaptatif notamment par multi-modèles présente de nombreux avantages pour atteindre les performances souhaitées en présence de grandes incertitudes, incluant des variations brusques inconnues dues à des défaillances. Toutefois, si plusieurs commandes de vols adaptatives et reconfigurables ont été proposées, par exemple [Athans *et al.*, 1977, Fekri *et al.*, 2008] ou [Ducard, 2009] parmi beaucoup d'autres, la littérature manque encore d'analyse détaillée de la robustesse en stabilité et en performances de l'ensemble de la boucle fermée comportant le contrôleur global.

5.3.2 Architecture du MMAC neuronal (NMMAC)

5.3.2.1 Banque de contrôleurs

Les contrôleurs qui composent la banque sont pertinents pour certains des différents cas d'utilisation, ceci au même titre que les différentes modélisations de la banque de modèles qui sont valides dans certaines régions d'opérations distinctes. En se focalisant sur la banque des contrôleurs, un pré-requis nécessaire à la stabilité du MMAC, est qu'il existe toujours un contrôleur stabilisant pour l'ensemble des valeurs (fixes) de variations possibles du système. Il s'agit donc pour le paramétrage incertain du système $\theta \in \Omega$ de s'assurer que les contrôleurs et leurs espaces de stabilité robuste recouvrent l'ensemble incertain Ω .

Dans le cadre linéaire, plusieurs approches d'analyses sont envisageables dont la plupart sont fondées sur les outils proposés par la commande robuste. De nombreux travaux utilisent la ν -gap métrique [Vinnicombe, 2000] associée à la marge de stabilité proposée par l'approche loop-shaping (voir [Mahdianfar *et al.*, 2011, Tao *et al.*, 2016, Kersting et Buss, 2017]). Suite à l'étude, si la condition de recouvrement n'est pas satisfaite, il convient alors à l'utilisateur d'ajouter judicieusement une paire modèle-contrôleur ou de relâcher des exigences de performances.

Pour le NMMAC, les contrôleurs constituant la banque sont des contrôleurs neuronaux, directement issus des différentes phases d'identification. L'un des enjeux des travaux de thèse est qu'aucune présupposition sur les contrôleurs et leur cahier des charges n'est émise puisque tout est contenu dans les données d'entraînement. Ainsi, la banque de contrôleurs découle des différentes bases de données et ils sont donc imposés lors de l'apprentissage. Néanmoins, un degré

de liberté supplémentaire peut être obtenu à l'aide des méthodes d'identification de contrôleurs robustes développées dans le chapitre 4. La considération de la stabilité lors de l'entraînement permet alors de contraindre les marges de stabilité des contrôleurs qui composent la banque. Dans chaque cas local, la formulation multi-objectifs du problème aboutit à un ensemble de contrôleurs de compromis optimaux selon les deux critères de performance et de robustesse. La sélection du contrôleur local dans l'ensemble optimal de Pareto peut alors s'effectuer au regard des autres contrôleurs locaux retenus et de la stabilité globale du MMAC. Un avantage apporté par la résolution multi-objectifs, est qu'elle permet de s'abstenir d'éventuelles présuppositions sur les contraintes de marges de robustesse à imposer à chacun des contrôleurs locaux dès la phase d'apprentissage. Chaque identification de contrôleur local peut alors s'effectuer indépendamment, tandis que l'analyse globale intervient lors de la sélection des contrôleurs locaux dans leurs ensembles optimaux de Pareto respectifs.

5.3.2.2 Banques modèles-estimateurs et logique de sélection

La banque de modèles est composée d'un ensemble de modélisations valides localement pour certaines régions de l'espace de variation dans lequel le système réel est supposé évoluer. La répartition des modèles doit permettre de couvrir l'ensemble des incertitudes possibles du système réel. Même si la banque de modèles n'est pas directement impliquée dans l'architecture du MMAC, les modèles peuvent être utilisés pour l'analyse ou la synthèse des estimateurs et des contrôleurs.

Les estimateurs sont directement fondés sur les modèles préalablement établis afin de prendre part à la logique de sélection. L'entité d'identification et de décision a pour objectif de déterminer dans quelles régions d'opération se situe le système actuel par rapport aux modèles de la banque. Les correcteurs impliqués dans l'asservissement découleront alors de façon directe de cette sélection. Bien qu'une banque d'observateurs soit employée, les états reconstruits ne sont pas utilisés pour la commande. Seules les sorties reconstruites sont employées pour l'estimation des validités. Une façon intuitive de procéder est d'utiliser une banque de modèles et non d'estimateurs afin de reconstruites les sorties estimées. Cependant, les observateurs présentent de nombreux avantages comme la robustesse aux bruits ou aux perturbations par le biais d'états augmentés par exemple. Leur utilisation a également un intérêt pour la maîtrise de la vitesse de convergence de l'erreur d'estimation qui est un point important pour le réglage de la stratégie de commutation. De plus, si des modèles sont employés, leurs sorties ne sont pas directement rebouclées avec les contrôleurs et ils se retrouveront donc en boucle ouverte.

Dans le contexte de nos travaux, nous ne supposons pas d'autres connaissances du système que celles contenues dans les données. Les modèles sont issus de l'apprentissage sur les jeux de données successifs et les modèles se verront donc imposés. La banque de modèles s'agrandira à chaque itération ainsi, la pertinence du MMAC devrait donc augmenter avec le nombre de modèles même si pour des raisons d'implémentation, il est toutefois possible d'étudier après coup l'importance et l'utilité de certains d'entre eux.

Pour le NMMAC, la représentation du système s'effectue à l'aide des modèles neuronaux acquis par les phases successives d'identification du système. L'élaboration du modèle neuronal qui avait déjà été nécessaire pour l'évaluation des marges de stabilité du chapitre 4, se révèle également utile dans le cadre multi-modèle. Pour former la banque d'estimateurs, la méthode de synthèse peut tirer profit de la forme des modèles neuronaux qui sont de type SSNN. Ainsi,

tout type d'observateur non-linéaire peut être implémenté à partir de la représentation d'état non-linéaire afin de construire la banque. Des estimateurs tels que des EKF peuvent être directement obtenus à partir des modèles neuronaux. Une technique commune de robustification des observateurs qui peut être employée ici, est d'envisager un modèle augmenté tant que le système reste détectable. L'état augmenté contient alors l'état du système et les perturbations possibles (stochastiques ou non), supposées continues par morceaux. Cette technique bien connue ([Mayne *et al.*, 2006] à titre d'exemple) permet d'améliorer la fiabilité des estimateurs et donc du MMAC.

La logique de sélection assurée par le PPE doit être réglée en fonction de la dynamique des estimateurs. Lors de l'emploi d'EKF, les matrices de pondérations de la loi bayésienne peuvent être désignées à l'aide des matrices de covariance de l'innovation comme évoqué par l'équation (5.7). Ainsi, le mécanisme de sélection favorise les modèles se rapprochant le plus du système réel actuel en termes de la distance mathématique référée comme distance de Mahalanobis [Mahalanobis, 1936]. Dans le cadre non-linéaire, les matrices de covariance sont calculés à chaque pas d'échantillonnage par les EKF et elles sont donc fournies au PPE pour le calcul des validités.

SECTION 5.4

Étude de la stabilité et réglage dans le cas linéaire

Nous abordons dans cette section, le cas linéaire afin d'examiner la faisabilité et l'intérêt de l'approche multi-modèle. Les modèles et les contrôleurs ainsi que le système sont supposés être LTI issus de processus linéaires ou linéarisés. L'élaboration d'une loi de commande MMAC peut être, de façon générale, décomposée d'une part, en la recherche d'un ensemble de contrôleurs qui couvre l'ensemble des incertitudes du système et d'autre part en la conception d'une stratégie de commutation.

Selon la philosophie de nos travaux, les modèles et les contrôleurs sont imposés avec une certaine limitation par les données. Nous adoptons alors dans cette partie une approche générale en considérant un ensemble de paires modèle-contrôleur nominales supposées connues. La méthodologie est présentée dans cette section en temps continu, mais des résultats analogues peuvent se déduire dans le domaine discret. Nous présentons dans un premier temps, un cadre d'étude de la stabilité de la loi de contrôle globale fondé sur la μ -analyse. Cette technique d'analyse fournit, sous certaines hypothèses, un domaine de combinaisons de contrôleurs candidats où la robustesse de la stabilité est garantie pour des valeurs fixes des paramètres. Nous nous concentrons dans un second temps, sur le réglage des unités d'identification et de décision. Nous proposons alors d'employer des observateurs de Luenberger dont les réglages sont directement reliés au calcul de validités en ligne suivant la loi bayésienne. Nous décrivons alors une stratégie systématique de réglage des observateurs selon les résultats de l'analyse de stabilité précédente. La viabilité de la méthodologie est illustrée pour le chariot mobile avec pendule inversée soumis à d'importantes variations paramétriques.

5.4.1 État de l'art

Malgré la maturité de l'approche multi-modèle et ses nombreuses applications fructueuses, les principaux enjeux du MMAC résident dans le développement d'un cadre d'analyse de la stabilité et d'une méthode de synthèse. Des recherches actives qui abordent à la fois la robustesse en stabilité et en performances sont regroupées sous le terme RMMAC (*Robust*

Multiple Model Adaptive Control) comme par exemple [Fekri *et al.*, 2006, Mahdianfar *et al.*, 2011, Rosa, 2011]. Ces techniques emploient le plus souvent une banque de filtres de Kalman en tant qu'estimateurs, pour lesquels il est absolument essentiel d'avoir un réglage convenable. Le RMMAC utilise alors comme logique de commutation le calcul récursif fondé sur la formule bayésienne (5.6), où l'heuristique probabiliste est pilotée par les matrices de covariance du bruit de mesures des filtres de Kalman selon (5.7). Les filtres sont donc des éléments critiques de l'approche cependant leurs réglages se révèlent en pratique complexes et spécifiques aux cas d'utilisation.

Une conception appropriée de chaque filtre de Kalman est essentielle dans l'architecture RMMAC pour permettre une identification correcte par la logique de sélection. Les travaux de [Baram et Sandell, 1978] se sont focalisés sur l'analyse de la convergence de la logique bayésienne, les conclusions sont que cette dernière converge vers le modèle le plus proche du système réel au sens d'une norme stochastique référée comme la distance de Baram ou BPM (*Baram Proximity Measure*). Ainsi, la logique de commutation du multi-modèle sélectionne le filtre de Kalman possédant la plus petite valeur de BPM. Les travaux offrent des résultats théoriques de stabilité essentiels via la BPM, cependant le calcul de cette mesure comporte quelques points moins accessibles dans la pratique. Par exemple, une condition suffisante pour le calcul de la BPM est que le système ait dans le cas discret ses valeurs propres à l'intérieur du cercle unité, de ce fait, son utilisation est impossible pour le cas des systèmes instables.

Les travaux [Fekri *et al.*, 2006, Fekri *et al.*, 2007] utilisent la BPM afin de concevoir la banque de filtres de Kalman dans le RMMAC. L'idée clé exploitée, est d'utiliser un algorithme itératif pour faire coïncider les BPM des différents modèles avec les frontières des sous-régions où les contrôleurs sont stabilisants. En d'autres termes, le modèle dont le contrôleur assure la stabilité robuste et des performances robustes dans une sous-région, aura la plus petite BPM dans cette même région. De cette manière, la logique de sélection converge presque sûrement vers ce modèle. Cette méthode devient plus compliquée pour deux ou plusieurs paramètres incertains (voir [Fekri *et al.*, 2006]). L'article [Hassani *et al.*, 2011] fournit une analyse de stabilité intéressante pour le RMMAC en supposant un tel réglage soigneusement choisi qui est essentiel pour satisfaire les hypothèses théoriques. Les travaux de [Rosa *et al.*, 2009] abordent le cas de mauvais réglages des filtres par rapport aux bruits ou perturbations réels. La méthode a pour but de réduire l'impact d'un contrôleur mal assorti au système en limitant la période de transition nécessaire pour le disqualifier.

Le RMMAC offre donc un cadre d'analyse pour la stabilité avec une littérature active pour faire face aux paramètres variants et incertains. Bien que dans une majorité des cas, la méthode est développée pour des paramètres incertains constants, elle est par la suite testée pour des paramètres variants dans le temps (comme [Rosa *et al.*, 2007]). Le RMMAC peut notamment se rapprocher des nombreux travaux sur les systèmes linéaires à paramètres variants (LPV) puisqu'ils s'adressent à des problèmes similaires (voir [Kuipers et Ioannou, 2010]). De plus, une extension du RMMAC au cadre non-linéaire est facilement envisageable par le biais de filtres de Kalman étendus.

5.4.2 Analyse du processus de décision incertain

Selon les conditions d'étude de la section, les différents éléments du MMAC (système, modèles, contrôleurs, estimateurs) abordent un comportement linéaire. Sous cette hypothèse, le

calcul des poids en ligne par le PPE est le seul élément présentant une dynamique non-linéaire. Dans le cadre de l'analyse de stabilité du MMAC, la dynamique des poids de validité n'est pas prise en compte. Les poids sont supposés être constants par morceaux afin de pouvoir utiliser les méthodes classiques d'analyse de stabilité telles que la μ -analyse avec une certaine limitation des résultats. L'analyse de stabilité se concentre donc sur les poids qui peuvent ne pas avoir une correspondance parfaite avec le système réel en raison de leur estimation par le PPE.

Il est important de signaler que la μ -analyse offre une méthode générale et flexible. Bien que seule la stabilité soit étudiée par la suite, il est facilement envisageable de considérer des incertitudes complexes supplémentaires afin de prendre en compte des robustesses de marges de stabilité, des robustesses de performances ou d'éventuelles dynamiques négligées. Une façon de réduire les limitations de la méthode pourrait être de considérer la dynamique des validités comme négligée.

5.4.2.1 Structure d'analyse du MMAC

Dans les conditions d'étude, les différents agents du MMAC sont supposés linéaires, le système G est alors décrit selon la représentation d'état multivariable de la forme :

$$G(\theta) := \begin{cases} \dot{x}(t) = A_\theta x(t) + B_\theta u(t) \\ y(t) = C_\theta x(t) \end{cases} \quad (5.11)$$

De plus, les contrôleurs de la banque ont également la dynamique linéaire suivante :

$$K_i := \begin{cases} \dot{x}_i^K(t) = A_i^K x_i^K(t) + B_i^K (r(t) - y(t)) \\ u_i(t) = C_i^K x_i^K(t) + D_i^K (r(t) - y(t)) \end{cases} \quad (5.12)$$

Dans cette approche d'analyse de la stabilité du MMAC, l'hypothèse suivante est émise.

Hypothèse 5.1 : Correspondance du système aux modèles

Le système réel $G(\theta)$ décrit par (5.11) est soumis à des variations de paramètres θ qui correspondent exactement à l'une des valeurs de paramètres candidates $\Theta = \{\Theta_1, \dots, \Theta_N\}$.

Sous l'hypothèse 5.1, l'objectif est d'analyser le comportement en boucle fermée pour le cas où le système G correspond à l'un des modèle $M_i, i \in \{1, \dots, N\}$. L'analyse est effectuée pour chaque paramètre $\theta \in \Theta$ et peut être facilement automatisée. L'objectif est de fournir une estimation de l'ensemble de combinaisons de validités où le système reste stable.

Remarque 5.2. *Par souci de simplicité des notations, seules les correspondances avec les systèmes présents dans la banque de modèles sont prises en compte, toutefois une généralisation à \tilde{N} configurations est envisageable. Une analyse est effectuée pour chaque système $G(\theta)$ correspondant à une valeur de paramètre candidat dans l'ensemble $\tilde{\Theta} := \{\tilde{\Theta}_1, \dots, \tilde{\Theta}_{\tilde{N}}\}$ qui peut comporter des cas d'inter-modèles entre ceux nominaux.*

Pour un système donné dépendant d'un paramètre fixe, les seules incertitudes à prendre en compte sont les poids fournis par le PPE. L'objectif de cette partie est de représenter le système à étudier et ses incertitudes sous forme LFT avec une matrice d'incertitudes structurée (la matrice d'incertitude est diagonale) pour permettre l'étude par la μ -analyse [Doyle, 1985, Duc

et Font, 1999].

L'objet d'étude est le système considéré $G(\theta)$, $\theta \in \Theta$ bouclé par la banque de contrôleurs dont la combinaison est incertaine. Pour analyser la robustesse de la stabilité, une façon courante de procéder consiste à choisir une paramétrisation $\rho_w = (w_1^0, \dots, w_{N-1}^0, w_1^1, \dots, w_{N-1}^1)$ qui permet d'examiner chaque $i^{\text{ème}}$ validité, autour de la valeur centrale w_i^0 et une gamme de variations possibles allant de $w_i^0 - w_i^1$ à $w_i^0 + w_i^1$. En respectant la propriété de convexité des combinaisons (5.4), la paramétrisation ρ_w est définie par :

$$\begin{cases} w_i = w_i^0 + w_i^1 \delta_{w_i} & \forall i \in \{1, \dots, N-1\} \\ w_N^0 = 1 - \sum_{i=1}^{N-1} w_i^0 \end{cases} \quad (5.13)$$

Imposer $w_i^1 = \min(w_i^0, 1 - w_i^0)$ assure que les validités soient bornées $0 < w_i < 1$ et que les incertitudes soient normalisées $-1 < \delta_{w_i} < 1$. Le choix des valeurs nominales pour la paramétrisation sera abordé dans la section suivante.

Une représentation LFT générale d'un système soumis à des incertitudes est donnée dans la figure 5.2. Toutes les incertitudes du modèle sont rassemblées dans la matrice Δ , la matrice de transfert $M_\theta(\rho_w)$ qui, dans le cas d'un système asservi, dépend évidemment du contrôleur mais dépend aussi des coefficients choisis pour la paramétrisation ρ_w . Le système étudié $M_\theta(\rho_w)$ est supposé stable et modélise les interconnexions internes avec le bloc d'incertitude par le biais des signaux v et z .

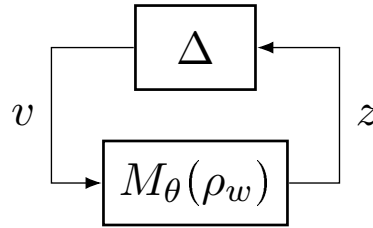


FIGURE 5.2 – Forme LFT pour l'analyse du MMAC

La dynamique du système avec ses incertitudes est régie par :

$$\dot{x} = A_\theta x + B_\theta \sum_{i=1}^N w_i u_i = A_\theta x + B_\theta \sum_{i=1}^N w_i^0 u_i + B_\theta \sum_{i=1}^N w_i^1 \delta_{w_i} u_i \quad (5.14)$$

Une forme LFT du MMAC pour la paramétrisation choisie $\rho_w = (w_i^0, w_i^1), i \in \{1, \dots, N-1\}$ est alors proposée, en considérant la commande fournie par la banque de contrôleurs (5.12), une référence nulle et avec $v(t) = \Delta z(t)$:

$$M_\theta(\rho_w) := \begin{cases} \dot{x}^M(t) = A_\theta^M x^M(t) + B_\theta^M v(t) \\ z(t) = C_\theta^M x^M(t) \end{cases} \quad (5.15)$$

avec $x^M = [x^T \ x_1^{K^T} \ \dots \ x_N^{K^T}]^T$, $v = [v_1^T \ \dots \ v_{N-1}^T]^T$, $z = [z_1^T \ \dots \ z_{N-1}^T]^T$, , les matrices données par

$$A_\theta^M = \begin{bmatrix} A_\theta - B_\theta \left(\sum_{i=1}^N w_i^0 D_i^K \right) C_\theta & B_\theta w_1^0 C_1^K & \cdots & B_\theta w_N^0 C_N^K \\ -B_1^K C_\theta & A_1^K & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ -B_N^K C_\theta & 0 & 0 & A_N^K \end{bmatrix}, \quad B_\theta^M = \begin{bmatrix} B_\theta w_1^1 & \cdots & B_\theta w_{N-1}^1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$C_\theta^M = \begin{bmatrix} -(D_1^K - D_N^K) C_\theta & C_1^K & 0 & 0 & -C_N^K \\ \vdots & 0 & \ddots & 0 & \vdots \\ -(D_{N-1}^K - D_N^K) C_\theta & 0 & 0 & C_{N-1}^K & -C_N^K \end{bmatrix}$$

et

$$\Delta(s) = \begin{bmatrix} \delta_{w_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \delta_{w_{N-1}} \end{bmatrix}$$

5.4.2.2 Domaine de stabilité du MMAC fondé sur la μ -analyse

Comme nous l'avons présenté dans la section 3.4.3, la μ -analyse est une méthode d'étude de robustesse qui permet de prendre en considération la structure des incertitudes du modèle via la valeur singulière structurée. Des résultats moins pessimistes sont alors obtenus par le théorème du petit gain généralisé (voir le théorème 3.5). Sur la base de ce théorème, il s'agit maintenant de calculer la valeur singulière structurée de $M_\theta(\rho_w)$ puis de déduire les plages d'incertitudes pour lesquelles le système en boucle fermée de la figure 5.2 reste stable. La μ -analyse donne l'hypercube le plus grand contenu dans le domaine de stabilité et centré sur le point nominal de la paramétrisation.

Pour un système fixe donné $G(\theta)$, $\theta \in \Theta$ et une paramétrisation ρ_w respectant (5.13), une borne supérieure $\bar{\mu}^{\rho_w} = \sup_w (M_\theta(j\omega, \rho_w))$ est calculée et aboutit à un domaine de stabilité garanti $\mathcal{D}_w^{\rho_w}(\theta)$ pour $|\delta_{w_i}| < \frac{1}{\bar{\mu}^{\rho_w}}$, $i \in \{1, \dots, N-1\}$, c'est-à-dire :

$$\mathcal{D}_w^{\rho_w}(\theta) := \left\{ w_i \in \left[w_i^0 - \frac{w_i^1}{\bar{\mu}^{\rho_w}}; w_i^0 + \frac{w_i^1}{\bar{\mu}^{\rho_w}} \right], i \in \{1, \dots, N-1\} \right\} \quad (5.16)$$

En raison de la propriété de symétrie des incertitudes, les résultats d'une seule évaluation de l'analyse est très pessimiste par rapport au domaine réel de stabilité. Pour résoudre ce problème, la μ -analyse est évaluée sur un ensemble fini de paramétrisations de combinaison \mathcal{P}_w . Le domaine total de stabilité garanti pour chaque système donné $G(\theta)$, $\theta \in \Theta$ est obtenu par l'union des domaines de chaque paramétrisation :

$$\mathcal{D}_w(\theta) = \bigcup_{\rho_w \in \mathcal{P}_w} \mathcal{D}_w^{\rho_w}(\theta) \quad (5.17)$$

Le choix de l'ensemble de paramétrisation \mathcal{P}_w doit couvrir toute la gamme des validités possibles. Les nœuds d'évaluation de l'analyse peuvent être une grille de l'espace convexe des poids incertains avec un pas approprié. Pour maximiser le domaine $\mathcal{D}_w(\theta)$, un quadrillage suffisamment fin se révèle efficace pour maximiser l'estimation du domaine mais nécessite plus

de ressources de calcul. En effet plus la paramétrisation sera dense, plus le domaine de stabilité garanti s'approchera du domaine de stabilité réel dans lequel il est inclus.

La méthode proposée permet alors d'intuiter la robustesse du MMAC en procurant pour un système $G(\theta)$ donné, un ensemble de validités pour lequel la stabilité est assurée si $w \in \mathcal{D}_w(\theta)$.

5.4.2.3 Application

Afin d'illustrer la méthode d'analyse de stabilité de l'approche multi-modèle, nous reprenons le cas d'étude du chariot mobile avec pendule présenté dans le chapitre 4. L'objectif est d'asservir la position du chariot tout en régulant cette fois-ci le pendule en position inversée (c'est-à-dire autour de π) comme l'illustre la figure 5.3.

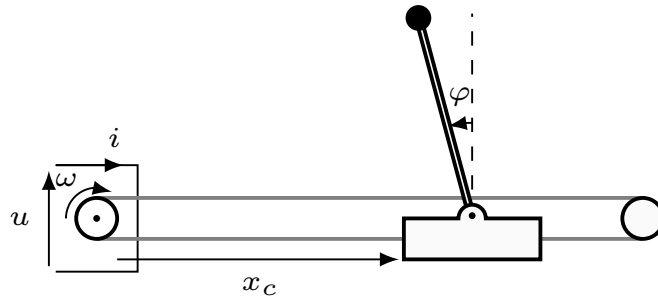


FIGURE 5.3 – Chariot mobile avec pendule inversé

Le comportement du système est régi par les équations non-linéaires définies en (4.1) où les valeurs nominales des différents paramètres sont résumés dans le tableau 4.1. La linéarisation du système en position inversée autour de $\varphi \approx \pi$ est donnée, en négligeant la constante de temps électrique, par la représentation d'état suivante :

$$\frac{d}{dt} \begin{bmatrix} x_c(t) \\ \omega(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} 0 & r/n & 0 & 0 \\ 0 & -a & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -ad & g/l & -f_\alpha/l \end{bmatrix} \begin{bmatrix} x_c(t) \\ \omega(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ b \\ 0 \\ bd \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ -c \\ 0 \\ -cd \end{bmatrix} \gamma(t) \quad (5.18)$$

$$a = \frac{\phi^2}{RJ} + \frac{f}{J}, \quad b = \frac{\phi}{RJ}, \quad c = \frac{1}{J}, \quad d = \frac{r}{ln}$$

Dans le but d'illustrer la méthode multi-modèle, nous supposons que le système est soumis à d'importantes variations paramétriques pour lesquelles il est nécessaire de concevoir un contrôle adaptatif par MMAC. Pour la clarté de l'exemple, trois cas sont considérés correspondant à des variations de l'inertie ramenée sur l'axe moteur. La banque de modèles se compose des modèles $M_i, i \in \{1, 2, 3\}$, obtenus pour l'ensemble de valeurs pour le paramètre incertain $\Theta = \{\Theta_1, \Theta_2, \Theta_3\}$ qui sont définies par :

$$\begin{aligned} \text{Modèle 1 :} & \quad \Theta_1 = \{J = 5 e^{-6}\} \\ \text{Modèle 2 :} & \quad \Theta_2 = \{J = 50 e^{-6}\} \\ \text{Modèle 3 :} & \quad \Theta_3 = \{J = 125 e^{-6}\} \end{aligned} \quad (5.19)$$

Le premier cas correspond à un fonctionnement nominal du système. Les deux autres cas représentent un fonctionnement dégradé avec une modification de l'inertie du moteur qui peut

correspond à une variation de la charge à l'extrémité du pendule.

La banque de contrôleurs est également composée de trois éléments $K_i, i \in \{1, 2, 3\}$, où un contrôleur approprié K_i est conçu pour chacun des trois systèmes M_i . La synthèse de chaque contrôleur reprend l'architecture à trois degrés de liberté de la figure 4.5 où le correcteur a pour entrées l'écart entre la consigne et la position, la position du chariot et l'angle du pendule par rapport à la position inversée. L'élaboration de contrôleurs multivariables optimaux \mathcal{H}_∞ est réalisé à l'aide de la méthode *hinfsyn* de MATLAB en considérant des filtres de pondération adaptés. Pour les besoins de la démonstration, chaque contrôleur stabilise son modèle associé en respectant une marge de stabilité appropriée, mais les contrôleurs répondent à des spécifications élevées sur leurs performances et ne stabilisent pas, individuellement, l'ensemble des systèmes obtenus pour les valeurs du paramètre considérées.

Il convient à présent d'étudier la stabilité par μ -analyse de la loi de commande du MMAC. Le MMAC est défini selon la banque de modèles M_i et celle des contrôleurs K_i établies précédemment. La banque des estimateurs n'est quand à elle pas considérée ici puisque les validités qu'elle génère, sont abordées uniquement en tant que paramètres incertains. Selon les postulats de l'étude (hypothèse 5.1), le paramètre incertain J du système asservi coïncide à l'une des configurations incluses dans $\Theta = \{\Theta_1, \Theta_2, \Theta_3\}$. Pour chaque configuration, la forme LFT d'étude définie par (5.15), est établie pour différentes combinaisons de paramétrisation $\rho_w \in \mathcal{P}_w$. Dans chacun des sous-cas, un domaine de stabilité $\mathcal{D}_w^{\rho_w}(\Theta_i)$ est obtenu par μ -analyse selon (5.16) et enfin le domaine total de stabilité garanti $\mathcal{D}_w(\Theta_i)$ est déterminé par union.

La figure 5.4 présente les résultats de la μ -analyse obtenus dans le cas nominal Fig. 5.4(a) et dans les cas dégradés Fig. 5.4(b), Fig. 5.4(c). Les résultats sont représentés en deux dimensions puisque la valeur de la troisième validité peut être déduite des deux autres (propriété de convexité (5.4)). Sur les figures, les domaines des poids admissibles obtenus par μ -analyse sont tracés sous forme d'un rectangle rempli et les croix représentent les valeurs nominales de poids pour lesquelles le système en boucle fermée $M_\theta(\rho_w)$ n'est pas stable et donc pour lesquelles l'analyse de stabilité robuste ne peut et ne doit pas être effectuée. Pour le système donné $G(\Theta_i)$, le domaine dans lequel la stabilité est garantie $\mathcal{D}_w(\Theta_i)$ avec des combinaisons fixes de contrôleurs est obtenu par l'union des hyper-cubes donnés par la μ -analyse. Lors de cette étude, un quadrillage de paramétrisation des poids nominaux avec un pas de grille de 0,02 est utilisé pour couvrir tous les poids tout en respectant leur convexité (5.4).

Dans les configurations Θ_1 et Θ_3 , une contribution élevée des contrôleurs non associés ne permet pas de conclure sur la stabilité. Il est intéressant de noter que dans le cas de Θ_2 , qui est un cas central, les contrôleurs ont des marges suffisantes pour qu'un grand nombre de combinaisons de contrôleurs puisse stabiliser le système. Il peut être intéressant d'utiliser cette analyse *a posteriori* pour la conception des contrôleurs afin d'assouplir certaines spécifications de performance pour élargir le domaine où la robustesse de la stabilité est assurée.

Cette analyse de stabilité nous permet d'évaluer la robustesse de notre approche multi-modèle. En effet, un large domaine de stabilité montre qu'une estimation plus approximative ou moins rapide peut être obtenue par le PPE. En revanche, un domaine très étroit concentré sur un sommet indique que le modèle diffère fortement des autres, de sorte que son contrôleur associé est très spécifique. Dans ce cas, sa validité estimée par le PPE devra être très précise.

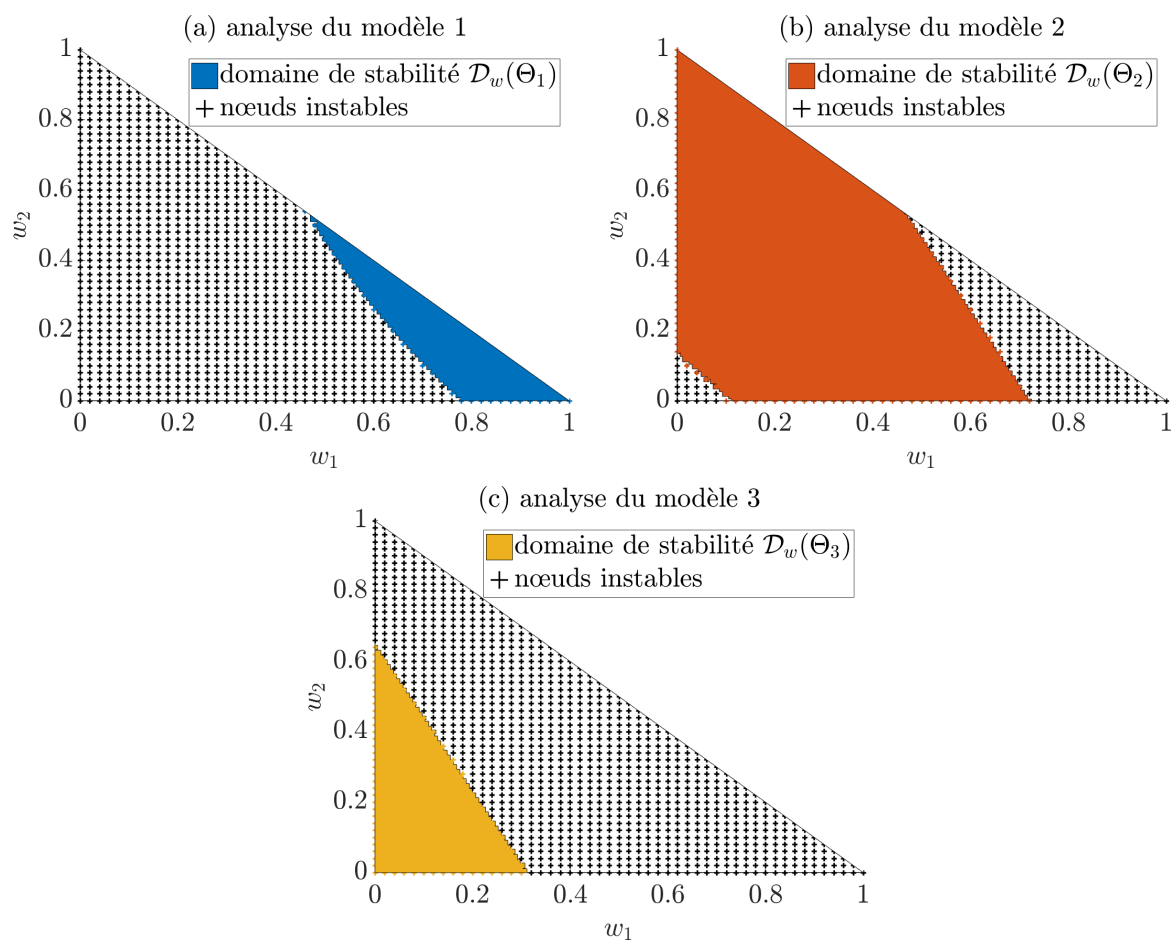


FIGURE 5.4 – Analyse de la stabilité pour les 3 configurations du système

5.4.3 Réglage de la logique de décision dans le cas linéaire

Nous développons dans cette partie une méthode de réglage du processus décisionnel du MMAC. L'architecture multi-modèle proposée est dénommée *Estimator-based MMAC*, elle emploie des observateurs de Luenberger dont le réglage est relié au calcul bayésien des validités. La conception des observateurs s'effectue par LMI, selon une stratégie s'appuyant sur l'analyse de stabilité de la section 5.4.2 et un critère caractéristique à minimiser.

5.4.3.1 Approche *Estimator-based MMAC*

Pour chaque modèle nominal $M_i, i \in \{1, \dots, N\}$, les sorties estimées \hat{y}_i sont obtenues via des observateurs de Luenberger E_i de la forme :

$$E_i := \begin{cases} \dot{\hat{x}}_i(t) &= A_i \hat{x}_i(t) + B_i u(t) + L_i (y(t) - \hat{y}_i(t)) \\ \hat{y}_i(t) &= C_i \hat{x}_i(t) \end{cases} \quad (5.20)$$

où $\hat{x}_i(t) \in \mathbb{R}^{n_x}$ est l'état estimé du $i^{\text{ème}}$ modèle local, $u(t) \in \mathbb{R}^{n_u}$ est le vecteur de commande, $\hat{y}_i(t) \in \mathbb{R}^{n_y}$ est le vecteur des sorties estimées et L_i sont les gains d'observation à déterminer. Lorsque le système coïncide avec le modèle nominal M_i , l'erreur d'observation associée $\varepsilon_i = x - \hat{x}_i$ suit la dynamique :

$$\dot{\varepsilon}_i(t) = (A_i - L_i C_i) \varepsilon_i(t) \quad (5.21)$$

Les gains d'observation L_i sont choisis de manière à assurer que l'erreur d'observation converge vers zéro pour n'importe quelle condition initiale.

Une façon de procéder est d'assurer une convergence exponentielle de l'erreur d'estimation via la α -stabilité de l'observateur. Les résultats bien connus issus de [Boyd *et al.*, 1994] fournissent des conditions suffisantes pour garantir un taux de convergence de l'erreur d'estimation selon un taux de décroissance donnée.

Théorème 5.1 : Estimateur exponentiellement stable [Boyd *et al.*, 1994]

Soit le système correspondant au modèle nominal M_i , et son observateur associé E_i , si il existe une matrice $P_i = P_i^T > 0$, une matrice R_i et un scalaire $\alpha_i > 0$ tels que

$$A_i^T P_i + P_i A_i - C_i^T R_i^T - R_i C_i + \alpha_i P_i < 0 \quad (5.22)$$

alors le gain d'observation $L_i = P_i^{-1} R_i$ garantit la décroissance globale et exponentielle de l'erreur d'estimation ε_i avec un taux de décroissance caractéristique minimal α_i .

Contrairement à la méthode de placement des pôles, le taux de convergence de la fonction de Lyapunov α_i , est un paramètre de conception caractérisant à lui seul les performances dynamiques de l'observateur.

Au même titre que le RMMAC qui utilise des filtres de Kalman [Fekri *et al.*, 2006, Hassani *et al.*, 2011], il est proposé dans cette thèse d'utiliser le réglage de l'observateur pour piloter la logique bayésienne du PPE. En utilisant les termes d'innovation des estimateurs, les validités sont récursivement calculées avec la matrice de convergence $\Gamma_i = L_i^T L_i$ et $\beta_i = 1$, par conséquent

l'équation (5.6) devient :

$$w_i(k+1) = \frac{\exp\left(-\frac{1}{2}\varepsilon_i^T(k)L_j^T L_j \varepsilon_i(k)\right) w_i(k)}{\sum_{j=1}^M \exp\left(-\frac{1}{2}\varepsilon_j^T(k)L_j^T L_j \varepsilon_j(k)\right) w_j(k)} \quad (5.23)$$

Ainsi, les estimateurs et le PPE sont regroupés sous une seule entité dont les seuls paramètres de réglage sont les dynamiques imposées aux erreurs d'observation directement dépendantes des $\alpha_i, i \in \{1, \dots, N\}$. Le paramétrage des α_i doit être soigneusement choisi et il est déterminant pour la stabilité de la boucle fermée globale.

5.4.3.2 Critère caractéristique du réglage du MMAC

Cette section part du principe qu'une banque de contrôleurs avec une conception appropriée est déjà disponible et qu'il n'est pas nécessaire de les modifier pour obtenir un réglage correct du MMAC. Ce travail se concentre donc sur les observateurs. En effet, la construction des estimateurs a un impact important sur la convergence du PPE, d'une part avec l'utilisation de l'erreur d'observation et d'autre part via la matrice de gain de l'observateur comme matrice de convergence de la mise à jour récursive des validités bayésiennes (5.23). Afin de fournir une approche systématique pour le réglage de l'observateur, une méthode fondée sur les domaines de stabilité déterminés dans la section précédente est proposée.

Pour le système LTI $G(\theta), \theta \in \Theta$, la stabilité en boucle fermée est assurée avec le domaine de stabilité correspondant pour toute validité $w \in \mathcal{D}_w(\theta)$. Selon l'hypothèse 5.1, les trajectoires du paramètre incertain θ sont variantes par morceaux. Une trajectoire de $\Theta_i \in \Omega$ à $\Theta_j \in \Omega$ est appelée commutation et définie par :

$$G(\Theta_j \rightarrow \Theta_i) := \left\{ G(\theta) \mid \exists t_s, \theta(t = t_s) = \Theta_j \text{ et } \theta(t > t_s) = \Theta_i \right\} \quad (5.24)$$

Une condition nécessaire mais non suffisante pour conclure sur la stabilité lors d'une commutation est que les validités appartiennent aux domaines de stabilité correspondants. Cependant, la dynamique des observateurs et du PPE n'étant pas négligeable, un calcul de poids en dehors du domaine stable semble inévitable. Il semble raisonnable de limiter la durée pendant laquelle les poids n'appartiennent pas au domaine de stabilité afin de ne pas compromettre la stabilité du système en boucle fermée. Il s'agit en effet de limiter et minimiser le temps passé en dehors du domaine de stabilité. Sous l'hypothèse d'exacte correspondance du système avec ceux envisagés (5.1), un nombre fini de commutation de $\Theta_j \in \Theta$ vers $\Theta_i \in \Theta$ peut être considéré : $G(\Theta_j \rightarrow \Theta_i), (i, j) \in \{1, \dots, N\}^2$ et $i \neq j$.

Par ailleurs, un ensemble de scénarios \mathcal{S} correspondant à l'utilisation typique du système est également envisagé. Cette série de scénarios temporels contient des profils de variations de paramètres en fonction des conditions opérationnelles. Elle peut également inclure des situations caractéristiques de signaux externes telles que des variations de référence importantes ou des perturbations potentielles. Afin de limiter la dépendance de la conception des observateurs aux scénarios envisagés, des signaux stochastiques comme des bruits de mesure significatifs peuvent être pris en considération.

Pour un scénario donné et une commutation $G(\Theta_j \rightarrow \Theta_i)$, il est possible de calculer le temps nécessaire $\tau_{\Theta_i \rightarrow \Theta_j}$ pour que les validités calculées par le PPE atteignent le domaine de stabilité et restent à l'intérieur :

$$\tau_{\Theta_i \rightarrow \Theta_j} := \inf_{\tau} \{ \forall t > \tau, w(t) \in \mathcal{D}_w(\Theta_i) \} \quad (5.25)$$

Ensuite, pour chaque scénario de commutation possible, un temps de parcours $\tau_{\Theta_i \rightarrow \Theta_j}$ à l'extérieur du domaine de stabilité peut être calculé et un temps global est obtenu par sommation comme suit :

$$\tau := \sum_{\sigma \in \mathcal{S}} \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \tau_{\Theta_j \rightarrow \Theta_i} \quad (5.26)$$

Le temps obtenu est proposé pour caractériser le réglage des observateurs et plus généralement la conception du MMAC, en effet pour rappel la dynamique du PPE dépend aussi principalement des gains des observateurs (5.23). Ce temps se présente comme une mesure qui doit être minimisée par rapport à la dynamique de l'observateur.

5.4.3.3 Optimisation du réglage du MMAC

Par la méthode présentée dans le théorème 5.1, la conception d'un observateur E_i est caractérisée en un seul paramètre α_i qui est le taux de décroissance exponentielle imposé à la dynamique de l'erreur de l'observateur. Pour une banque de contrôleurs donnée, l'objectif est de régler de manière optimale le MMAC en fonction de la banque d'observateurs de la forme (5.20), selon la méthode (5.22). Ceci est formulé par le problème d'optimisation suivant :

$$\min_{\alpha_i, i \in \{1, \dots, N\}} \tau(\alpha_i, i \in \{1, \dots, N\}) \quad (5.27)$$

Ce problème d'optimisation est fortement non-linéaire, il peut être résolu à l'aide d'algorithmes génétiques ou d'autres algorithmes d'optimisation sans gradient comme l'algorithme de Nelder-Mead [Lagarias *et al.*, 1998].

La fonction de coût est donnée par l'évaluation de l'organigramme suivant :

- Concevoir chaque gain d'observation L_i d'après (5.22) pour $i = \{1, \dots, N\}$,
- Calculer chaque $\tau_{\Theta_j \rightarrow \Theta_i}$ selon (5.25),
- Évaluer $\tau(\alpha_i, i \in \{1, \dots, N\})$ suivant (5.26).

Il est également possible de réduire la taille du problème d'optimisation en supposant une dynamique identique pour tous les observateurs. Les convergences des erreurs d'observation satisferront alors les équations de Lyapunov avec le même taux de décroissance imposé α . Le problème d'optimisation (5.27) devient alors limité à un seul paramètre α , il est plus conservatif mais plus facile à résoudre.

5.4.3.4 Application

Nous continuons dans cette partie l'illustration de la méthode à l'aide du cas d'étude du chariot mobile avec pendule de la section 5.4.2.3. Le système gouverné par les équations (4.1) est sujet à d'importantes variations paramétriques incertaines $\theta \in \Theta = \{\Theta_1, \Theta_2, \Theta_3\}$, une loi de contrôle de type MMAC est déployée dont les banques de modèle et de contrôleurs utilisées sont respectivement $M_i, i \in \{1, 2, 3\}$ et $K_i, i \in \{1, 2, 3\}$ qui ont été précédemment établies à partir de

chacune des configurations envisagées de (5.19).

Pour les trois cas considérés, les observateurs E_i sont basés sur des modèles augmentés avec une perturbation additive supposée constante sur les sorties. Cette technique, modélisant un bruit de mesure par exemple, va permettre de rendre plus robustes les estimateurs et indirectement la convergence du calcul des validités par le PPE. Les estimateurs sont de type Luenberger (5.20) et définis par la méthode présentée dans le théorème 5.1. Un taux de décroissance minimal α_i est imposé à la décroissance exponentielle de la fonction quadratique de Lyapunov en fonction de l'erreur d'estimation.

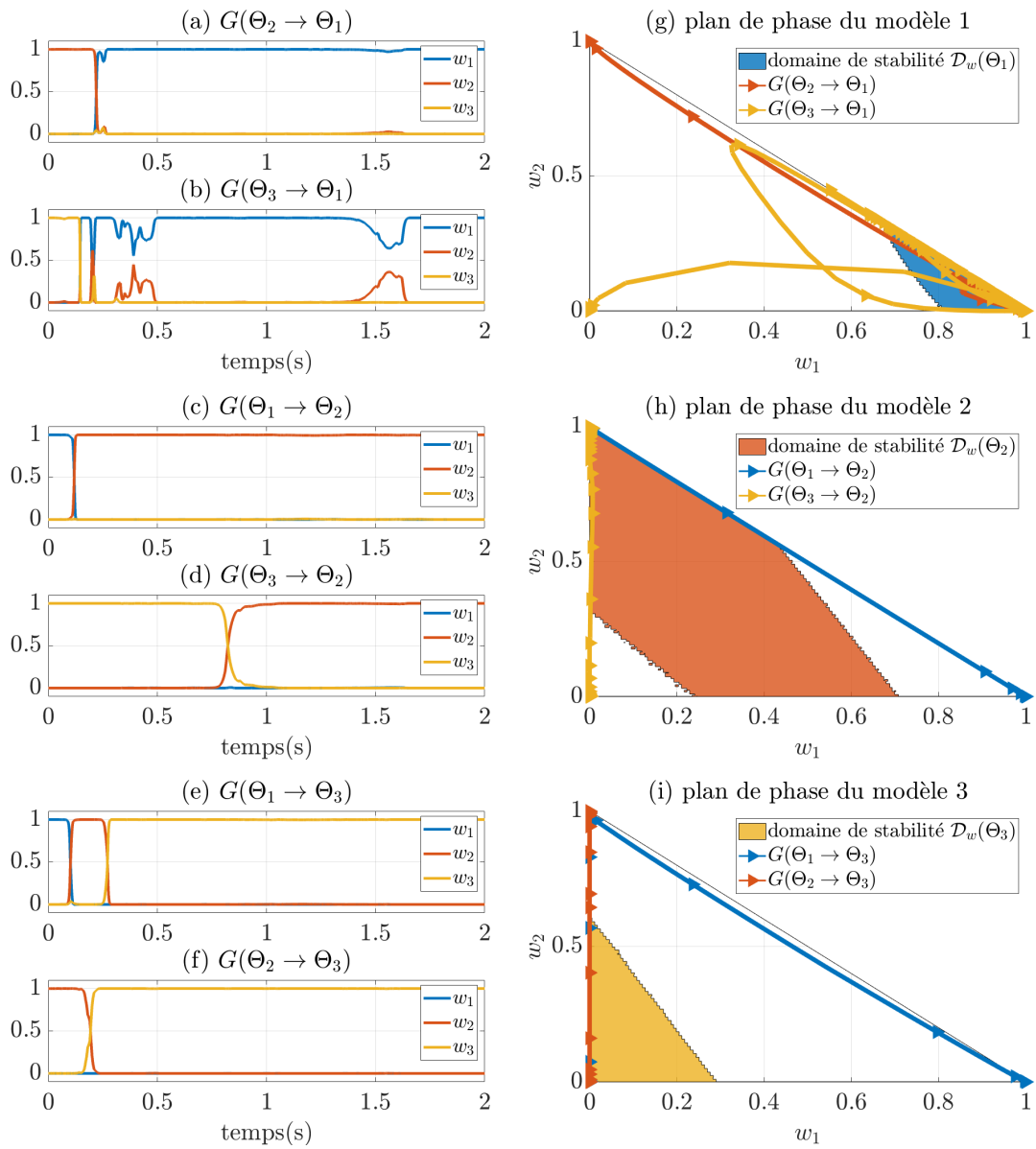
Il convient maintenant de prêter attention au réglage des observateurs et indirectement à la convergence du calcul des validités par le PPE (5.23). Pour chaque configuration du système considérée, il est possible d'étudier la robustesse en stabilité par μ -analyse comme cela a été effectué dans la section 5.4.2.3. Le réglage des taux de décroissance minimaux α_i est donc réalisé à l'aide des régions de stabilité garantie préalablement déterminées.

Pour ce faire, un scénario type d'utilisation du système est envisagé. Ce scénario comprend des variations de référence par morceaux entre $[-1; 1]$, un bruit blanc gaussien additif sur les sorties compris à 95% dans $[-0,02; 0,02]$ et un temps d'échantillonnage de $0,01s$. Le PPE fonctionne avec un temps d'échantillonnage de $10^{-3}s$ et un seuil $\delta = 10^{-3}$. Tous les changements possibles $G(\Theta_j \rightarrow \Theta_i), (i, j) \in \{1, 2, 3\}^2, i \neq j$ sont également pris en considération. Par la simulation du scénario choisi et pour chaque situation de commutation, il est possible d'associer une métrique caractéristique $\tau_{\Theta_i \rightarrow \Theta_j}$ (5.25) correspondant au temps nécessaire pour atteindre le domaine de stabilité $\mathcal{D}_w(\Theta_i)$ et y rester. Une caractéristique globale τ est alors obtenue en additionnant les différents temps générés (5.26).

La figure 5.5 illustre les résultats obtenus pour chaque configuration donnée. Les figures 5.5(a)-(f) présentent les évolutions temporelles des poids pour chaque scénario étudié et les figures 5.5(g)-(i) représentent les trajectoires des poids dans le plan de phase. Pour cet exemple de réglage, les temps caractéristiques calculés sont $\tau_{\Theta_2 \rightarrow \Theta_1} = 0.345s$, $\tau_{\Theta_3 \rightarrow \Theta_1} = 0.358s$, $\tau_{\Theta_1 \rightarrow \Theta_2} = 0.121s$, $\tau_{\Theta_3 \rightarrow \Theta_2} = 0.773s$, $\tau_{\Theta_1 \rightarrow \Theta_3} = 0.259s$, $\tau_{\Theta_2 \rightarrow \Theta_3} = 0.179s$ et leur somme $\tau = 2.04s$.

L'étape suivante consiste à minimiser ce critère selon le problème d'optimisation (5.27). Pour illustrer le plus simplement possible la méthode présentée, une dynamique similaire est imposée à tous les observateurs par $\forall i \in \{1, 2, 3\}, \alpha_i = \alpha$. Le problème de réglage du MMAC devient ainsi un problème d'optimisation à paramètre unique α . Il est proposé d'évaluer la fonction de coût pour un quadrillage approprié de valeurs de réglage possibles selon l'organigramme introduit dans la section 5.4.3.3. La figure 5.6 expose avec une échelle logarithmique, les valeurs de fonction de coût pour α dans l'intervalle $[10^{-2}; 10^2]$. Cette figure expose le comportement fortement non-linéaire de la fonction de coût du problème d'optimisation considéré. Le réglage optimal du MMAC pour la méthode proposée et le scénario de simulation considéré est obtenu pour $\alpha = 10^{0.479} = 3.013$.

Les simulations sont réalisées à partir du système décrit par les équations non-linéaires (4.1). Le système est sujet à des variations paramétriques $\theta \in \Theta$ et un bruit de mesure gaussien compris à 95% dans $[-0,02, 0,02]$. La figure 5.7 témoigne du comportement satisfaisant du MMAC en présentant un exemple de réponses de chariot et du pendule avec le réglage retenu. Un point



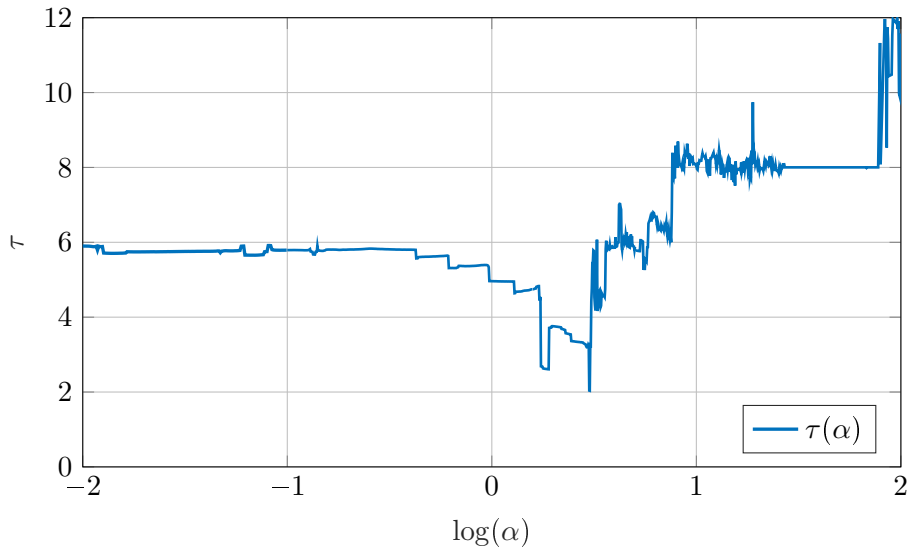


FIGURE 5.6 – Évaluation de la fonction de coût

important à noter est que la majorité des réglages possibles du MMAC ne garantissent pas la stabilité du système. Et qu'il semble très difficile et fastidieux de trouver manuellement un réglage suffisamment robuste.

5.4.4 Limitations de l'étude

L'étude de stabilité du MMAC menée au cours de cette section comporte deux hypothèses restrictives qui imposent certaines limites aux résultats obtenus.

Premièrement, supposer les validités constantes par morceaux pour la μ -analyse n'apportent pas d'assurance sur la stabilité. En effet, commuter entre deux systèmes stables peut engendrer une instabilité. L'étude menée par μ -analyse a permis d'intuiter la stabilité, mais ce n'est pas une garantie pour des poids qui vont varier dans le temps. L'approche pourrait être étendue à l'aide de dynamiques négligées judicieusement introduites ou dans un cadre plus général par l'utilisation des IQC (*Integral Quadratic Constraints*) [Megretski et Rantzer, 1997, Veenman et al., 2016].

De plus, l'hypothèse 5.1 restreint les variations possibles du paramètre inconnu du système à un ensemble fini de valeurs candidates. Une fois encore, la dynamique de variations n'est pas prise en compte. Les résultats obtenus sont donc à modérer en fonction de la vitesse de variation envisagée pour le paramètre inconnu. Le nombre de cas fini considéré peut également être source de limitation, néanmoins la méthode présentée est facilement automatisable, il est tout à fait possible de relâcher la restriction en considérant un nombre important de valeurs.

Bien que la stabilité n'ait pu être garantie, l'étude proposée offre une structure d'analyse prometteuse, de plus la méthode de réglage du MMAC résultante présente de très bons résultats dans les nombreux cas d'applications simulés.

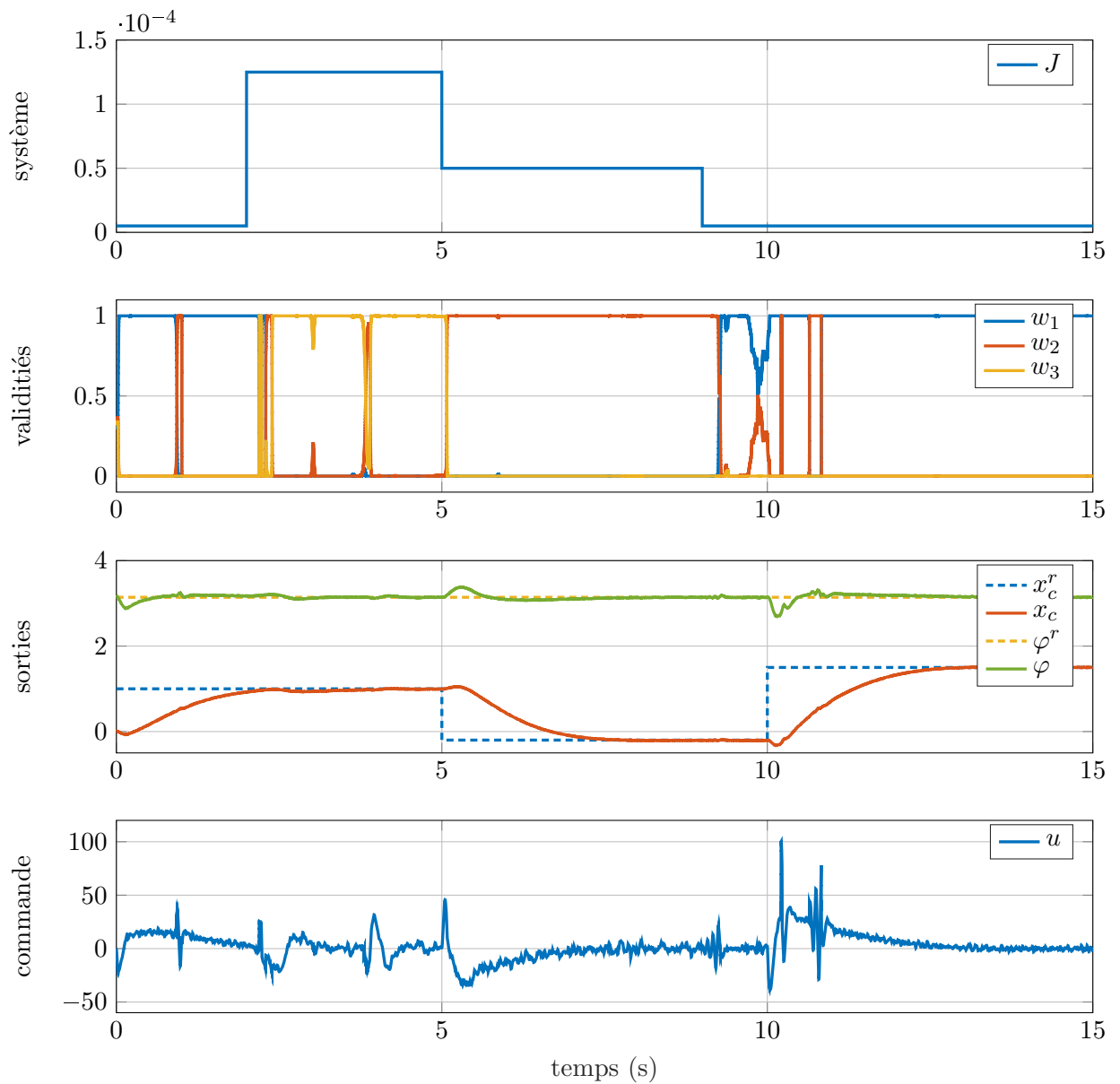


FIGURE 5.7 – Réponses du MMAC pour le réglage optimal retenu lors de variations paramétriques

Conclusions et perspectives

5.5.1 Conclusions

Ce chapitre présente la mise en place d'une technique de commande fondée sur les multi-modèles qui a pu être complétée par l'intégration de modèles et contrôleurs neuronaux. L'approche permet d'aborder des systèmes présentant de grandes incertitudes ou variations paramétriques, en particulier le système peut par exemple être sujet à des pannes pour lesquelles il est nécessaire de les détecter et de se reconfigurer. L'approche offre donc une architecture ordonnée et flexible afin d'accumuler des connaissances sous forme de MMAC.

Ce chapitre fournit également une méthode pour l'analyse de stabilité du MMAC dans le domaine linéaire ainsi qu'un processus de réglage. L'approche est fondée sur la représentation LFT du problème et la μ -analyse est utilisée sur un ensemble de paires modèle-contrôleur nominaux afin de fournir un domaine de stabilité des combinaisons de contrôleurs. Bien que le cadre d'étude impose certaines limitations aux résultats, il pose les bases de l'analyse de stabilité et offre de nombreuses perspectives. Une autre contribution des travaux réside dans la proposition d'une méthode qui utilise des observateurs de Luenberger dont le réglage permet de guider le calcul bayésien des validités. Les observateurs sont synthétisés au regard des résultats de l'analyse de stabilité effectuée au préalable. Un avantage certain de la méthode est de fournir une solution pour concevoir simultanément la logique de commutation.

5.5.2 Perspectives

Les travaux entrepris dans ce chapitre abordent un sujet bien complexe pour lequel nous avons pu apporter des premiers résultats d'étude et d'implémentation, néanmoins de nombreuses perspectives sont envisageables.

Concernant l'étude de stabilité menée dans le contexte linéaire, la μ -analyse qui est employée ici pourrait être substituée par des outils d'analyse de systèmes LPV afin de prendre en considération le caractère variant des paramètres. Pour aller plus loin, l'analyse pourrait spécifiquement tenir compte du calcul des validités en introduisant cette connaissance dans les incertitudes. Comme nous l'avons déjà évoqué ci-dessus, ces considérations pourraient être faites en introduisant des incertitudes pour les dynamiques négligées ou en utilisant des contraintes quadratiques intégrales (IQC). La prise en compte de variations paramétriques dans le système peut également être étudiée en ajoutant par exemple de nouvelles incertitudes.

De plus, l'extension de la méthode multi-modèle avec l'utilisation de réseaux de neurones pourrait être étudiée davantage. L'approche entreprise dans le domaine linéaire constitue un point de départ pour cette étude où les différents réseaux pourraient être appréhendés selon des modèles obtenus par une linéarisation locale. De plus, par la méthode d'apprentissage multi-objectifs présentée dans le chapitre 4, un ensemble de contrôleurs est obtenue le long du front de Pareto pour chacun des modèles. Il est alors envisageable de relier le choix des contrôleurs sur leur front de Pareto respectif au regard de la stabilité globale du MMAC, c'est-à-dire de déterminer le degré de robustesse exigé pour chaque contrôleur.

Troisième partie

**Applications à un pilote automatique
d'avion**

Plateforme d'apprentissage d'autopilotes neuronaux

Sommaire

6.1	Introduction	146
6.2	Module de co-simulation	147
6.2.1	Le simulateur de vol : X-Plane	147
6.2.1.1	Présentation	147
6.2.1.2	Interface de données	148
6.2.2	Mise en pratique de la co-simulation	149
6.2.2.1	Configuration de X-Plane	149
6.2.2.2	Configuration de Matlab et Simulink	150
6.3	Module d'apprentissage de contrôleurs robustes	151
6.3.1	Synthèse d'un autopilote de guidage en vol libre	151
6.3.1.1	Description du problème	151
6.3.1.2	Génération des données	152
6.3.1.3	Imitation de l'autopilote de X-Plane	154
6.3.1.4	Imitation d'un pilote	156
6.3.2	Synthèse d'un autopilote de guidage pour le suivi du chemin de vol	164
6.3.2.1	Description du problème	164
6.3.2.2	Génération des données	167
6.3.2.3	Apprentissage de l'autopilote neuronal	167
6.3.2.4	Expérimentations	170
6.4	Module de déploiement de l'autopilote par multi-modèle	176
6.4.1	Déploiement d'un NMMAC de guidage en vol libre	176
6.4.1.1	Description et intérêt du problème	176
6.4.1.2	Apprentissage multi-modèle de l'autopilote de X-Plane	176
6.4.1.3	Implémentation du NMMAC pour le contrôle autonome	178
6.4.2	Déploiement d'un NMMAC de guidage pour le suivi de chemin de vol	179
6.4.2.1	Description du problème	179
6.4.2.2	Apprentissage multi-modèle d'un pilote	182
6.4.2.3	Implémentation du NMMAC pour le suivi de chemin de vol	184
6.5	Conclusions et perspectives	186
6.5.1	Conclusions	186
6.5.2	Perspectives	188

SECTION 6.1

Introduction

L'objectif de ce chapitre est de tirer profit des méthodes et outils développés au cours des travaux de thèse pour apprendre par imitation un pilote automatique d'avion. Cette partie décrit en ce sens, une plateforme pour développer des autopilotes neuronaux uniquement fondés sur des bases de données de vol. Les simulateurs de vol tels que X-Plane s'avèrent être des outils puissants et efficaces pour créer une telle base de données et expérimenter les méthodes de contrôle. La méthodologie est répartie en trois modules qui correspondent aux trois principales étapes du développement et de mise en œuvre de l'autopilote :

- Collecte des données : Au cours de différentes sessions de simulation, les informations du vol sont enregistrées et regroupées dans une banque de données. Cette dernière comprend les principales données de vol couplées aux actions d'un pilote ou d'un autopilote déjà implémenté qui constituera le sujet de l'imitation.
- Apprentissage hors-ligne : Différents contrôleurs neuronaux sont développés à l'aide des bases de données de vol. Les apprentissages sont effectués conformément aux méthodes présentées dans le chapitre 4, afin d'identifier des contrôleurs robustes neuronaux ainsi que des modèles neuronaux du système.
- Contrôle autonome : L'implémentation en ligne de l'autopilote est établie conjointement à la simulation de l'avion. Le simulateur fournit les données aéronautiques pertinentes (position, attitude, etc.) afin qu'il puisse calculer et envoyer les signaux de commande (position des actionneurs, angle de guidage de référence, etc.).

La principale contribution du chapitre est de formaliser une méthode complète allant de l'acquisition des données à l'implémentation du pilote automatique en trois étapes qui sont souvent traitées séparément dans la littérature. La plateforme est donc organisée en trois modules comme le montre l'organigramme de la figure 6.1.

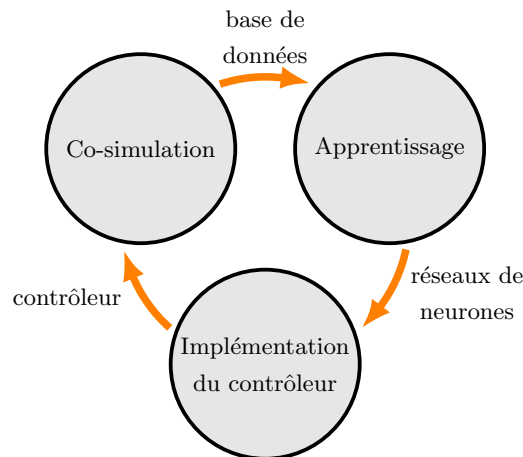


FIGURE 6.1 – Les trois modules de la plateforme d'apprentissage d'autopilotes neuronaux

Nous détaillons dans la section 6.2, le développement du module de co-simulation entre MATLAB et X-Plane en utilisant le protocole UDP (*User Datagram Protocol*). Dans la section 6.3, nous présentons le module d'apprentissage dont les fondements sont directement issus des méthodes

d'identification neuronale de contrôleurs robustes du chapitre 4. Le module d'implémentation du contrôleur qui est décrit dans la section 6.4 est quant à lui établi selon l'approche multi-modèle présentée dans le chapitre 5. Les modules peuvent intéresser le lecteur indépendamment les uns des autres et les sections peuvent être lues séparément.

SECTION 6.2

Module de co-simulation

Dans l'approche de co-simulation, les calculs de dynamique de vol sont effectués par le simulateur (figure 6.2). Ce processus fonctionne comme une boîte noire pour MATLAB, comme ce serait le cas pour un avion réel. Cette méthode bénéficie du haut degré de réalisme fourni par le simulateur, qui utilise des modèles de dynamique de vol réalistes. La plate-forme d'expérimentation est basée sur X-Plane 11, un simulateur de vol commercial développé par Laminar Research. Ce logiciel a été utilisé dans ce travail en raison de sa grande fiabilité de simulation et de sa souplesse d'utilisation.

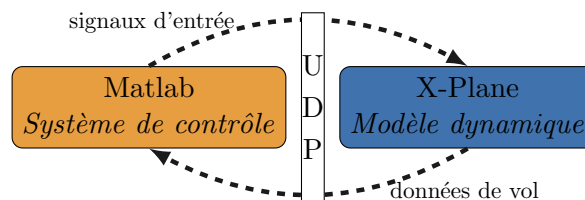


FIGURE 6.2 – Co-simulation entre X-Plane et MATLAB

Cette section met en évidence les étapes clés du développement du module de co-simulation entre X-Plane et MATLAB via l'environnement Simulink. En raison des ressources de calcul requises et de la contrainte de simulation en temps réel, les deux logiciels sont exécutés sur deux ordinateurs distincts.

6.2.1 Le simulateur de vol : X-Plane

6.2.1.1 Présentation

X-Plane est un logiciel de simulation de vol développé par Laminar Reserch, la version 11 du simulateur est initialement parue en 2017 et a connu de nombreuses mises à jour par la suite. X-Plane permet de simuler des sessions de vols d'aéronefs comme des avions, hélicoptères, planeurs ou fusées. Le logiciel intègre également de nombreux outils de création ou modification d'aéronefs et d'environnements. Le simulateur est développé pour un usage amateur ou professionnel à destination des pilotes ou en tant qu'outil de développement pour l'ingénierie.

X-Plane se caractérise par une grande précision de modélisation, ce qui permet une simulation avec un haut degré de similitude par rapport à un vol réel. Ce simulateur se distingue par son modèle de vol fondé sur la forme géométrique de l'avion. La simulation modélise en temps réel les forces agissant sur les différentes parties de l'avion pour déterminer la portance et la traînée, puis calcule l'accélération, la vitesse et la position de l'avion. Ce processus, appelé

théorie des éléments de pale (*blade element theory*), consiste à décomposer une surface en plusieurs petites sections et sous-sections, puis à déterminer les forces sur chacun de ces petits éléments. Cette approche diffère de la méthode empirique conventionnelle consistant à agréger des mesures puis à déterminer les forces aérodynamiques à l'aide de tables de correspondance prédéfinies. Le simulateur X-Plane est certifié par la Federal Aviation Administration (FAA) pour être utilisé par les pilotes comme outil de formation, lorsqu'il est connecté à du matériel certifié (cockpit et commandes de vol).

Un autre élément clé de X-Plane est sa large gamme d'options qui permet la création d'un environnement de simulation riche. Il est possible d'utiliser facilement différents aéronefs, y compris des modèles prototypes conçus avec le *Plane-Maker*. De nombreux paramètres de l'avion permettent de simuler des scénarios de défaillance ou d'autres événements spécifiques, très utiles pour le développement de pilotes automatiques intelligents. Certains des avions mis à disposition proposent également un autopilote interne déjà implémenté, celui-ci peut se révéler utile pour la création de données par exemple ou pour permettre d'aborder différents étages de contrôle comme le pilotage, le guidage ou la navigation. D'autres fonctionnalités sont également intégrées à X-Plane pour la gestion des conditions de vol, tels que les conditions atmosphériques, la vitesse et la direction du vent ou les turbulences.

Enfin, l'interaction avec le simulateur est grandement facilitée et flexible pour l'utilisateur ou un logiciel tiers. L'interface de jeu du simulateur permet l'intervention d'un pilote pour enregistrer sa réaction dans diverses situations ou analyser son comportement. De nombreuses données de vol sont également accessibles pendant la simulation, il est alors possible d'interagir avec le simulateur via un logiciel extérieur.

6.2.1.2 Interface de données

La communication entre X-Plane et d'autres logiciels extérieurs comme MATLAB est assurée par le protocole de datagramme utilisateur ou UDP (*User Datagram Protocol*). Ce protocole de télécommunication fait partie de la couche transport du modèle OSI (*Open Systems Interconnection*). Il permet la transmission de données sous forme de paquets de données ou datagrammes de manière simple entre deux entités définies par leur adresse IP et un numéro de port. Cette approche permet de faire fonctionner le simulateur et les autres logiciels avec lequel il interagit, sur des plateformes différentes, à condition qu'ils soient sur le même réseau local. Ce protocole ne dispose pas de mécanisme d'établissement de liaison (ou *handshaking*), il est alors exposé aux éventuels problèmes de fiabilité du réseau. Par conséquent, la transmission n'est pas garantie en termes de livraison et d'exactitude des données telles que l'ordre d'arrivée ou la duplication éventuelle des datagrammes. Ce protocole est donc adapté à une utilisation sans détection et correction d'erreurs. Dans un contexte de simulation en temps réel, la nature du protocole UDP le rend utile pour transmettre rapidement de petites quantités de données, puisqu'il est préférable de perdre éventuellement un datagramme plutôt que d'attendre qu'il soit retransmis.

Les datagrammes de X-Plane respectent un format standard consistant en un flux de paquets de données. Le paquet de données est une chaîne d'octets illustrée par le tableau 6.1 et respectant le schéma suivant :

- Une en-tête composé de 5 octets : les quatre premiers octets sont des codes ASCII pour

les mots-clés "DATA" indiquant qu'il s'agit d'un paquet de données. Le cinquième octet est un code interne défini comme "0".

- Un ensemble de données regroupées par groupe de 36 octets répétés par le nombre total de groupes : les quatre octets suivant l'en-tête constituent l'indice du premier groupe de données en décimale de 0 à 133. Les 32 octets suivants représentent les données de 8 paramètres à raison de 4 octets par paramètre. Un paramètre correspond à la valeur de la donnée en virgule flottante de simple précision (32-bit). Les autres groupes de données se succèdent alors dans le datagramme par des paquets de 36 octets en respectant le même schéma. Pour illustrer le format, le tableau 6.2 donne un exemple de jeu de données.

D'autres clarifications peuvent être ajoutées pour faciliter la mise en œuvre :

- La longueur du paquet de données varie en fonction du nombre total de groupes de données sélectionnés. Les groupes de données sont rangés les uns après les autres dans le message en respectant un ordre croissant en fonction de l'indice de chaque groupe. Le groupe de données ayant l'indice le plus bas sera inséré juste après l'en-tête et ainsi de suite.
- La majorité des groupes de données contiennent moins de 8 paramètres, les informations non utilisées sont donc remplies par le nombre "-999". Ce nombre, lorsqu'il est utilisé pour l'envoi, signifie également que la valeur du paramètre ne sera pas modifiée dans X-Plane.

TABLE 6.1 – Format des datagrammes de X-Plane

En-tête	Groupe n°1		Groupe n°2		...
5 bytes	9 × 4 bytes		9 × 4 bytes		
'DATA0'	Indice	8 Paramètres	Indice	8 Paramètres	...
5 bytes	4 bytes	8 × 4 bytes	4 bytes	8 × 4 bytes	

TABLE 6.2 – Exemple de format d'un groupe de données

Groupe "Pitch, roll & headings"					
	Indice	Paramètre 1	Paramètre 2	...	Paramètre 8
Terme	17	<i>pitch</i> , deg	<i>roll</i> , deg		<i>none</i>
Byte	17-0-0-0	156-162-9-192	240-92-19-65	...	0-192-121-196
Valeur	2.3822e-44	-2.15055	9.21016		-999

6.2.2 Mise en pratique de la co-simulation

6.2.2.1 Configuration de X-Plane

Les particularités de X-plane conféré par sa conception en tant qu'outil de développement, rendent son utilisation très souple pour l'expérimentation. De nombreuses données sont accessibles et peuvent être affichées pendant la simulation, envoyées sur un réseau ou écrites

dans un fichier. La sélection et l'utilisation des données se font par le biais du tableau de sortie des données. Le tableau indique les indices des groupes de données afin de correctement les identifier par la suite, néanmoins les contenus de ces groupes, c'est-à-dire les valeurs des paramètres, peuvent être obtenus à partir de leurs affichages dans le cockpit ou dans le fichier d'enregistrement. L'utilisateur peut choisir les groupes de données à envoyer ou à sauvegarder et le débit jusqu'à 99,9 paquets de données par seconde. Pour le fonctionnement correct de la communication par UDP, il est nécessaire de spécifier dans les options de X-Plane l'adresse IP de la machine où est exécuté MATLAB ainsi que le port de réception à relier.

De nombreuses autres options sont également disponibles, notamment sur le plan des performances graphiques. Le niveau d'affichage peut être ajusté en fonction de la machine utilisée afin de garantir une exécution en temps réel du simulateur. Un paramètre qui peut être important est le modèle de vol par image. Ce paramètre représente le nombre de fois que le simulateur calcule les forces aérodynamiques pour chaque image. Il doit être ajusté si la fréquence d'images est très faible et d'autant plus si l'avion est petit, rapide ou léger.

6.2.2.2 Configuration de Matlab et Simulink

Du côté de MATLAB, la réception et l'envoi de données à X-Plane s'effectuent dans l'environnement Simulink à l'aide de l'*Instrument Control Toolbox* qui fournit des blocs pour communiquer sur le réseau via UDP. Il est possible de contourner l'utilisation de cette *toolbox* en utilisant une méthode similaire à celle proposée dans [Aschauer *et al.*, 2015]. Néanmoins, une attention doit être portée à l'optimisation du code et au temps de calcul nécessaire pour ne pas ralentir l'exécution de MATLAB et assurer son bon fonctionnement en temps réel.

La réception et le décodage des données dans l'environnement Simulink s'effectuent selon les fonctions et les étapes suivantes :

- *UDP Receive* : Réception du message UDP depuis l'adresse IP où s'exécute le logiciel X-Plane et spécification du port à lier. Le message reçu est au format variable d'entier non signé de 8 bits (uint8) et sa taille sera de 5 octets pour l'en-tête plus 36 octets pour chaque ensemble de données de sortie sélectionné dans la table de sortie de données X-Plane.
- *Byte Unpack* : Décomposition des données en un tableau d'octets. L'en-tête est un *uint8* égal à "68-65-84-65-48" pour "DATA0" tandis que le reste du message doit être séparé en 9 groupes de valeurs flottant de simple précision (32-bit) pour chaque groupe de données.
- *Matlab Function* : Récupération de chaque paramètre individuellement en sélectionnant le groupe de données dans lequel le paramètre est contenu et sa position dans ce groupe.
- *Data Type Conversion* : Conversion des données en valeurs à double précision (64-bit) qui est souvent nécessaire pour une utilisation correcte avec Simulink.

La transmission des données au simulateur s'effectue également dans l'environnement Simulink. Notons que le nombre et la sélection des groupes de données envoyés sont indépendants de ceux reçus. Un protocole similaire mais inversé par rapport à celui précédemment présenté est utilisé pour envoyer les données souhaitées en respectant les étapes suivantes :

- *Data Type Conversion* : Convertir les données en variable de simple précision (32-bit).
- *Matlab Function* : Création des groupes de données qui doivent être injectés dans X-Plane en plaçant les 8 paramètres dans le bon ordre et en ajoutant la valeur "-999" au format uint8 pour les paramètres non utilisés ou non modifiés. L'indice associé au groupe de données doit précéder chaque série de paramètres, également au format uint8.

- *Byte Pack* : Regroupement des données d'entrée dans un seul vecteur de sortie de format uint8 contenant l'en-tête et les groupes de données successifs sans se soucier de l'ordre.
- *UDP Send* : Envoi du message UDP à X-Plane avec l'adresse IP correspondante et un port qui diffère de celui utilisé pour la réception.

Un point crucial de la co-simulation repose sur la synchronisation temporelle. Il faut veiller à ce que MATLAB ne prenne pas trop de temps pour calculer et envoyer les données à X-Plane. Dans le cas contraire, les paquets sont mis en mémoire tampon et les calculs de MATLAB sont fondés sur des données de vol obsolètes, ce qui peut entraîner un comportement imprévisible. À notre connaissance, il n'existe aucune possibilité de synchroniser les horloges du logiciel de simulation de vol et de MATLAB. Même s'il est possible d'imaginer un fonctionnement séquencé de modes pause et lecture de X-Plane qui serait imposé par MATLAB, cette manipulation s'avère impraticable. La manière la plus simple de procéder est de conditionner un fonctionnement en temps réel sur les deux logiciels. Du côté de X-Plane, la simulation suit les performances en temps réel, et il est très simple de détecter un ralentissement de la simulation grâce aux messages d'avertissement. Sur MATLAB, l'utilisation du bloc de synchronisation en temps réel (*Real-Time Synchronization*) fourni par la *Simulink Desktop Real-Time Toolbox* permet d'exécuter le modèle Simulink avec une horloge en temps réel.

SECTION 6.3

Module d'apprentissage de contrôleurs robustes

Cette section a pour objectif d'illustrer les méthodes d'apprentissages de contrôleurs robustes développées dans le chapitre 4. En tirant profit de la co-simulation avec X-Plane, différents autopilotes d'avion sont développés à partir de bases de données de vol. L'étage de contrôle adressé dans les travaux se concentre sur le guidage de l'appareil, c'est-à-dire la détermination des consignes d'attitudes. Ce choix est lié aux contraintes de simulation en temps réel et aux difficultés matérielles de traiter par exemple, l'étage de contrôle de pilotage qui agit directement sur les gouvernes de l'appareil, mais qui nécessite une fréquence d'échantillonnage plus élevée.

6.3.1 Synthèse d'un autopilote de guidage en vol libre

6.3.1.1 Description du problème

La problématique de contrôle abordée dans cette partie se concentre sur l'élaboration d'un pilote automatique pour le guidage d'un avion. Dans un contexte de navigation aérienne libre, l'objectif est d'asservir l'aéronef selon un cap et une altitude souhaités. Ainsi, conformément au schéma 6.3, le pilote automatique abordé vise à suivre les signaux de référence de cap ψ_r et d'altitude h_r . Les signaux de commande composés de la référence de roulis ϕ_r et de la référence de tangage θ_r sont calculés dans MATLAB avant d'être transmis au simulateur, qui retourne alors le cap ψ et l'altitude h actuel de l'appareil.

L'étage de contrôle traité dans ce travail correspond à celui du guidage, ainsi les commandes calculées sont les angles de guidage de référence. Un étage de pilotage garanti alors le suivi des consignes de guidage en agissant directement sur les gouvernes de l'appareil. Dans nos simulations, cette fonction est assurée via l'autopilote interne de X-Plane qui réalise l'asservissement en agissant sur les ailerons, la gouverne de profondeur ainsi que son compensateur et la gouverne de direction. De plus, la vitesse de l'appareil est maintenue à une vitesse constante

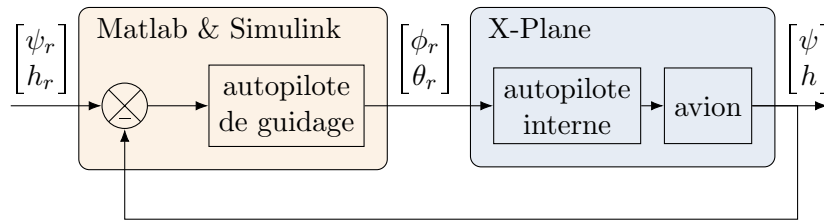


FIGURE 6.3 – Schéma de simulation de la boucle d'asservissement de guidage

dont la régulation est également effectuée par l'autopilote de pilotage. Finalement, le système sujet à l'asservissement est donc composé de la dynamique de l'avion ainsi que l'autopilote interne de pilotage de X-Plane.

L'objectif de synthèse est finalement de développer par imitation un pilote automatique de guidage d'un avion à partir de données de vol. Afin d'illustrer la méthode, nous expérimenterons successivement deux imitations distinctes. Une première expérimentation consiste à imiter l'autopilote de guidage (ou directeur de vol) déjà implémenté dans X-Plane et conçu par les développeurs. Par la suite, une seconde expérience est alors de reproduire les facultés de pilotage d'un opérateur capable de guider l'appareil lors de phases de vol libre.

Les simulations présentées dans cette section, emploie le Cessna 172SP (voir la figure 6.4) qui est un avion monomoteur à aile fixe avec un train d'atterrissage tricycle. Il s'agit d'un appareil d'entraînement populaire qui compte parmi les modèles installés dans la flotte par défaut de X-Plane et dont la vitesse de croisière est d'environ 120 nœuds. Il est équipé d'un pilote automatique utile à l'expérimentation, ainsi que de l'instrument de bord G1000 qui affiche plusieurs informations utiles au pilotage de l'aéronef. Comme le montre la figure 6.5, l'horizon artificiel indique l'attitude actuelle de l'aéronef en jaune qui est contrôlé par l'autopilote interne afin de respecter les consignes de guidage provenant du directeur de vol qui sont affichées en magenta. De plus, le compas (en bas) et l'altimètre (à droite) sont utilisés lors des sessions de vol avec le pilote pour lui indiquer explicitement les consignes de cap et d'altitude à respecter. Il peut ainsi guider l'avion selon un vol aux instruments.



FIGURE 6.4 – Cessna 172SP

6.3.1.2 Génération des données

La méthodologie d'apprentissage de contrôleur, et même de façon plus générale des méthodes d'apprentissage par imitation nécessitent l'approvisionnement d'une base de données illustrant



FIGURE 6.5 – Instrument de vol G1000 de X-Plane 11

les comportements à reproduire de manière suffisamment abondante et condensée. Ainsi, pour générer les bases de données de vol, des consignes de navigation, c'est-à-dire de cap et d'altitude sont envoyées à X-Plane afin de disposer d'un signal d'apprentissage suffisamment riche. Lors d'une première simulation, le pilote automatique de l'avion est employé dans son intégralité afin de contrôler l'aéronef. Les systèmes d'autopilote interne et du directeur de vol sont amorcés dans le simulateur ainsi que le suivi de cap et d'altitude. Pour la seconde simulation, le directeur de vol est court-circuité par les commandes de guidage d'un pilote qui a pour tâche de contrôler l'avion en se conformant aux consignes de cap et d'altitude. L'intervention du pilote est effectuée à l'aide d'un joystick dont les deux axes sont directement reliés aux angles de consigne de roulis et de tangage. Les positions du manche sont récupérées dans l'environnement Simulink de MATLAB, elles sont par la suite converties en degré avant d'être transmises à X-Plane.

Les signaux de référence de navigation sont composés d'une succession d'échelons d'amplitude et de longueur aléatoires. La génération de ces signaux d'excitation est effectuée conformément à la méthode présentée dans le chapitre 1 (section 1.5) sur la procédure d'identification de systèmes dynamiques par réseaux de neurones. Les amplitudes sont délimitées par des valeurs minimales et maximales, qui sont forcées d'apparaître dans le signal d'excitation au moins une fois afin de s'assurer que le système soit excité sur le domaine d'opération souhaité. De même, les durées des échelons sont déterminées de manière à ce que l'état d'équilibre soit atteint dans une proportion plus ou moins importante des cas. Les caps souhaités sont choisis entre 150° et 300° , et les échelons ont une durée n'excédant pas 60s. Simultanément, les paliers d'altitude spécifiés sont délimités entre 2000ft (pieds) et 4000ft et leur durée est limitée à 120s. Pour la première simulation qui implique le directeur de vol, ce dernier contrôle le taux de montée ou descente lors des phases de changement d'altitude selon une vitesse verticale de 1500ft/min. Durant l'ensemble des vols, la vitesse air de l'appareil est régulée à l'aide du pilote automatique interne selon une vitesse de consigne de 100kt (nœuds).

Pendant les phases de simulations de vol, les données sont enregistrées par X-Plane à une fréquence d'écriture de 20Hz, soit un temps d'échantillonnage de 0.05s qui est une fréquence

cohérente avec celles classiquement utilisées pour le guidage. Les vols sont effectués par temps calme autour de l'aéroport de Paris Orly. La première simulation de l'autopilote de X-Plane aboutit à la création d'une première base de données d'une durée d'environ deux heures. La seconde base de données est générée à l'aide de la seconde simulation d'une durée approximative d'une heure qui implique cette fois-ci le pilote.

À l'issue des sessions de vol, les données sont extraites des fichiers textes et récupérées dans MATLAB pour l'apprentissage hors-ligne. Dans le processus de pré-traitement, chaque base de données est ré-échantillonnée selon une période d'échantillonnage uniforme de 0.05s, convertie en unité de base du Système International (SI) puis normalisée par la méthode *min-max* avant d'être divisée en trois ensembles constitués à 70% pour l'ensemble d'entraînement, 15% pour l'ensemble de validation et 15% pour l'ensemble de test.

6.3.1.3 Imitation de l'autopilote de X-Plane

Nous nous intéressons dans cette partie à l'apprentissage par imitation de l'autopilote de guidage déjà implémenté dans X-Plane. Nous ne présumons d'aucune structure interne concernant l'autopilote. Ainsi, nous abordons le problème de contrôle en considérant les dynamiques longitudinales et latérales comme couplées. À partir de la base de données associée, deux phases d'apprentissages sont réalisées indépendamment, la première correspondant au contrôleur à identifier selon les vecteurs entrées-sorties :

$$\left\{ \begin{bmatrix} \psi_r - \psi \\ h_r - h \end{bmatrix}, \begin{bmatrix} \phi_r \\ \theta_r \end{bmatrix} \right\} \quad (6.1)$$

La seconde identification désigne celle du système à asservir qui est nécessaire pour l'évaluation des marges de stabilité et dont les vecteurs d'entrées-sorties sont :

$$\left\{ \begin{bmatrix} \phi_r \\ \theta_r \end{bmatrix}, \begin{bmatrix} \psi \\ h \end{bmatrix} \right\} \quad (6.2)$$

Nous réalisons les apprentissages conformément à la procédure détaillée dans la section 1.5. L'algorithme d'apprentissage appliqué est celui de Levenberg-Marquardt qui est généralement l'un des plus efficaces pour l'entraînement hors-ligne des réseaux peu profonds. Le critère d'arrêt prématuré est retenu lorsque le coût de validation augmente pour 100 itérations successives et le réseau final retenu correspond à celui dont le coût de validation est le plus faible. Un nombre de 10 tirages est effectué pour chaque structure et le réseau ayant la plus faible performance de validation est sélectionné. Enfin, les poids d'entrée et de sortie des réseaux sont modifiés en fonction des valeurs de normalisation afin qu'ils puissent être utilisés pour les données réelles. Une méthode essai-erreur est alors nécessaire afin de déterminer la structure et les hyper-paramètres appropriés, le tableau 6.3 fournit les résultats d'apprentissages des réseaux ainsi retenus.

Le contrôleur neuronal $K_{\text{autopilote}}^{\text{NN}}$ est obtenu selon un apprentissage sous une forme SSNNOE tandis que l'apprentissage du système $G_{\text{autopilote}}^{\text{NN}}$ nécessite l'apprentissage en deux temps où le premier, sous forme SSNNARX, initialise un second sous forme récursive SSNNOE. Les sorties estimées des modèles neuronaux sont données par les figures 6.6 et 6.7 en comparaison des sorties de la base de données pour le jeu de validation et de test. Ces figures illustrent les bonnes performances temporelles des modèles neuronaux, le contrôleur neuronal reproduit presque

TABLE 6.3 – Résultats d'apprentissage pour l'imitation de l'autopilote de X-Plane

	Structure						Résultats				
	topologie	n_{in}	n_{out}	σ_1	n_1	n_W	iter.	durée	err. app.	err. val.	err. test
$K_{\text{autopilote}}^{\text{NN}}$	SSNNOE	2	2	tanh	5	87	484	20m	7.9e-4	1.2e-3	1.0e-3
	SSNNARX	0	5	tanh	10	176	166	1m	6.7e-9	7.6e-9	6.9e-9
$G_{\text{autopilote}}^{\text{NN}}$	SSNNOE	0	5	tanh	10	176	202	12m	1.1e-4	2.3e-4	5.4e-3

parfaitement les consignes de guidage de l'autopilote tandis que le système neuronal présente des dynamiques similaires à celles d'origine même si une erreur statique tend à s'accumuler au fur et à mesure de la simulation.

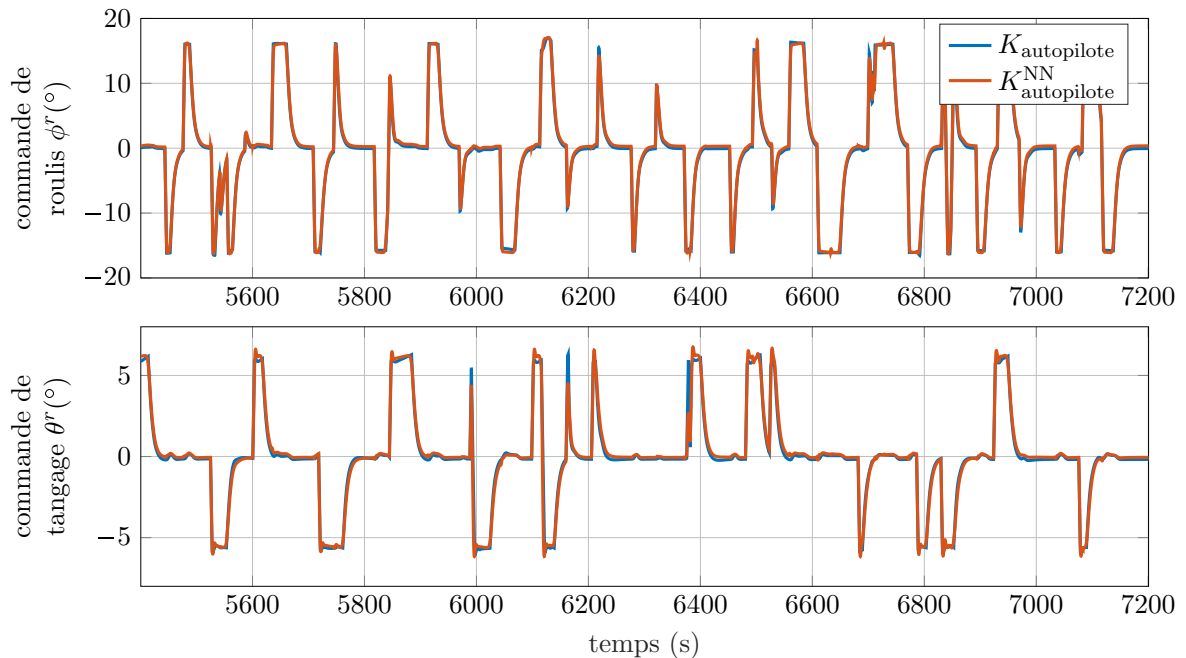


FIGURE 6.6 – Résultat d'apprentissage du contrôleur pour l'imitation de l'autopilote de X-Plane sur les données de validation et de test

Le contrôleur neuronal est finalement étudié en stabilité et en performance. D'une part, les marges de la boucle d'asservissement sont estimées *a posteriori* à l'aide du contrôleur neuronal $K_{\text{autopilote}}^{\text{NN}}$ et du modèle neuronal $G_{\text{autopilote}}^{\text{NN}}$. Le tableau 6.4 résume les marges de stabilité calculées en entrée et en sortie selon les marges de disque introduites dans la section 4.3. Le problème à l'étude étant de grande dimension, les outils de résolutions de LMI font alors face à de nombreuses difficultés numériques, ainsi les marges de stabilités sont estimées à l'aide de la méthode subsidiaire par μ -analyse comme nous l'avons présenté dans la section 3.4.3 et avec les réserves déjà évoquées. Les marges de robustesses obtenues sont satisfaisantes ainsi il n'est pas nécessaire d'effectuer un apprentissage avec prise en compte de ces marges de stabilité via la méthode d'optimisation multi-objectifs. D'autre part, l'autopilote neuronal est co-simulé avec X-Plane afin d'assurer le suivi de cap et d'altitude d'un nouveau scénario de test qui inclut des

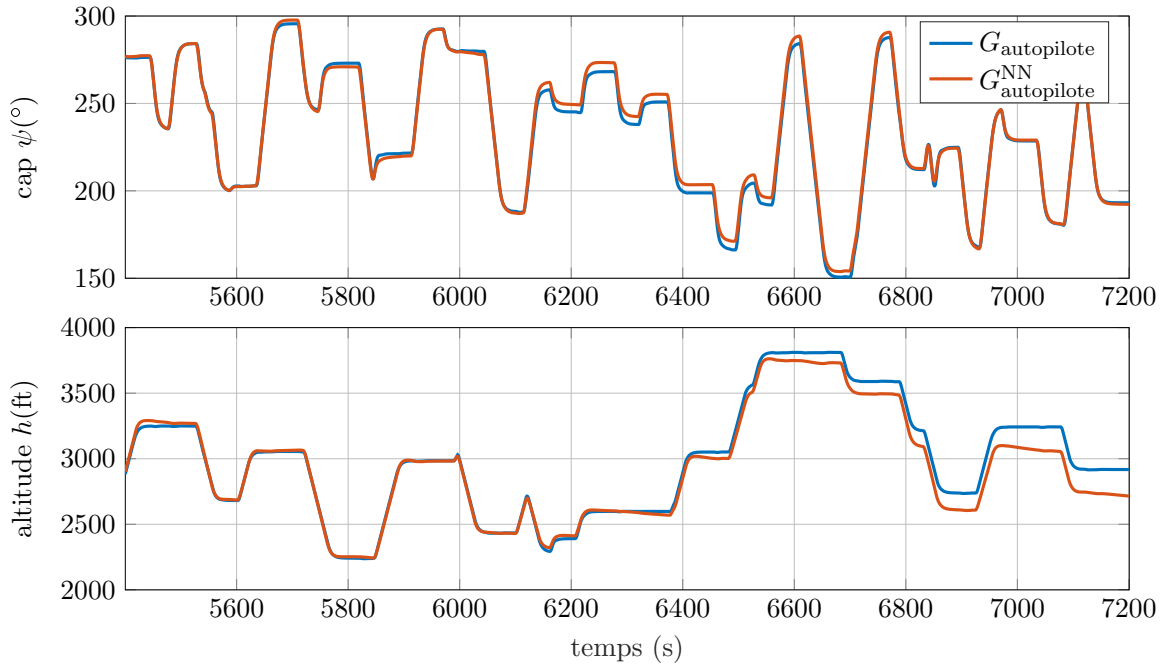


FIGURE 6.7 – Résultat d'apprentissage du système pour l'imitation de l'autopilote de X-Plane sur les données de validation et de test

consignes en échelons et en rampes. Au cours du vol, la vitesse air de l'avion est maintenue autour d'une consigne de 100kt par l'autopilote interne. La figure 6.8 présente les résultats de simulation pour un vol par temps calme en comparaison des réponses de l'autopilote de X-Plane pour ce même scénario. Les résultats montrent des performances satisfaisantes dans le contrôle du cap et de l'altitude qui coïncide presque parfaitement¹ au comportement du pilote automatique de X-Plane.

TABLE 6.4 – Marges de stabilité de l'asservissement obtenu par l'imitation de l'autopilote de X-Plane

Marge en entrée				Marge en sortie			
$\gamma^{\text{DM}_u^{\mu}}$	DM_u^{μ}	GM_u	PM_u	$\gamma^{\text{DM}_y^{\mu}}$	DM_y^{μ}	GM_y	PM_y
2.22	0.46	4.1 dB	26.2°	2.83	0.35	3.1 dB	20.0°

6.3.1.4 Imitation d'un pilote

Nous abordons à présent dans cette partie l'apprentissage d'un autopilote par imitation d'un joueur occasionnel guidant l'appareil au cours de session de vol libre. La méthode d'apprentissage est similaire à celle utilisée pour l'imitation de l'autopilote de X-Plane, en utilisant ici la seconde base de données générée à partir des réactions du pilote. L'algorithme

1. Les oscillations de commande de tangage de l'autopilote de X-Plane s'explique par l'alternance inévitable d'activation et de désactivation du mode de maintien d'altitude de l'autopilote lorsque la consigne est une rampe.

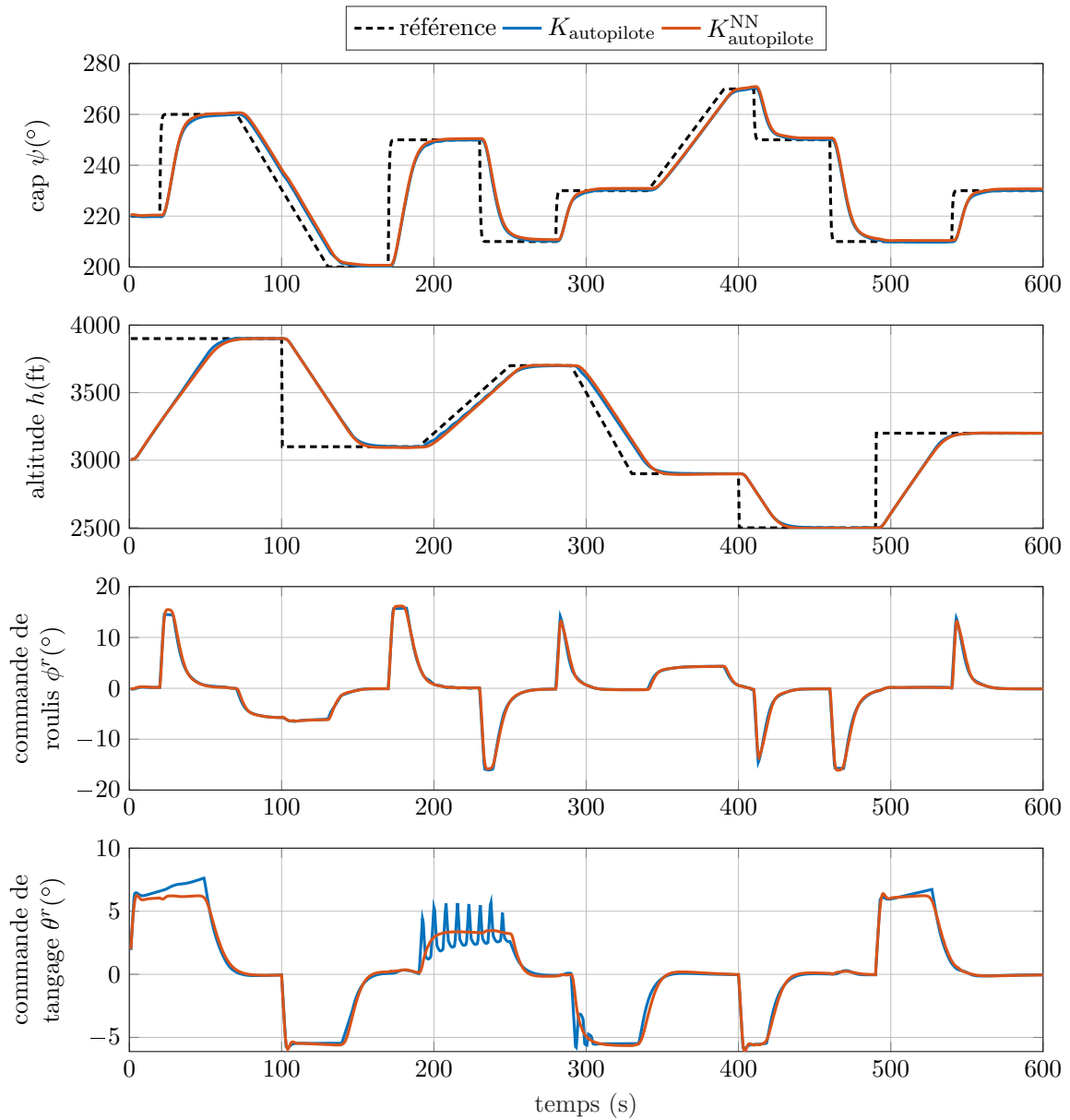


FIGURE 6.8 – Comparaison des simulations de l'autopilote de X-Plane et de celui appris pour un scénario test

de Levenberg-Marquardt est appliqué pour l'apprentissage selon un critère d'arrêt prématuré choisi à 500 itérations successives et pour 10 tirages par essai. Le tableau 6.5 présente les résultats d'apprentissages du contrôleur neuronal et du système neuronal retenus, dont là encore la topologie des réseaux a été déterminée empiriquement. Les figures 6.9 et 6.10 illustrent également les sorties estimées et attendues des réseaux respectifs.

TABLE 6.5 – Résultats d'apprentissage pour l'imitation du pilote

	Structure						Résultats				
	topologie	n_{in}	n_{out}	σ_1	n_1	n_W	iter.	durée	err. app.	err. val.	err. test
K_{pilote}^{NN}	SSNNOE	3	4	tanh	10	224	538	28m	9.1e-3	1.1e-2	9.9e-3
	SSNNARX	0	5	tanh	10	176	526	1m	1.9e-8	2.5e-8	1.4e-8
G_{pilote}^{NN}	SSNNOE	0	5	tanh	10	176	1278	49m	2.4e-4	6.5e-3	7.2e-3

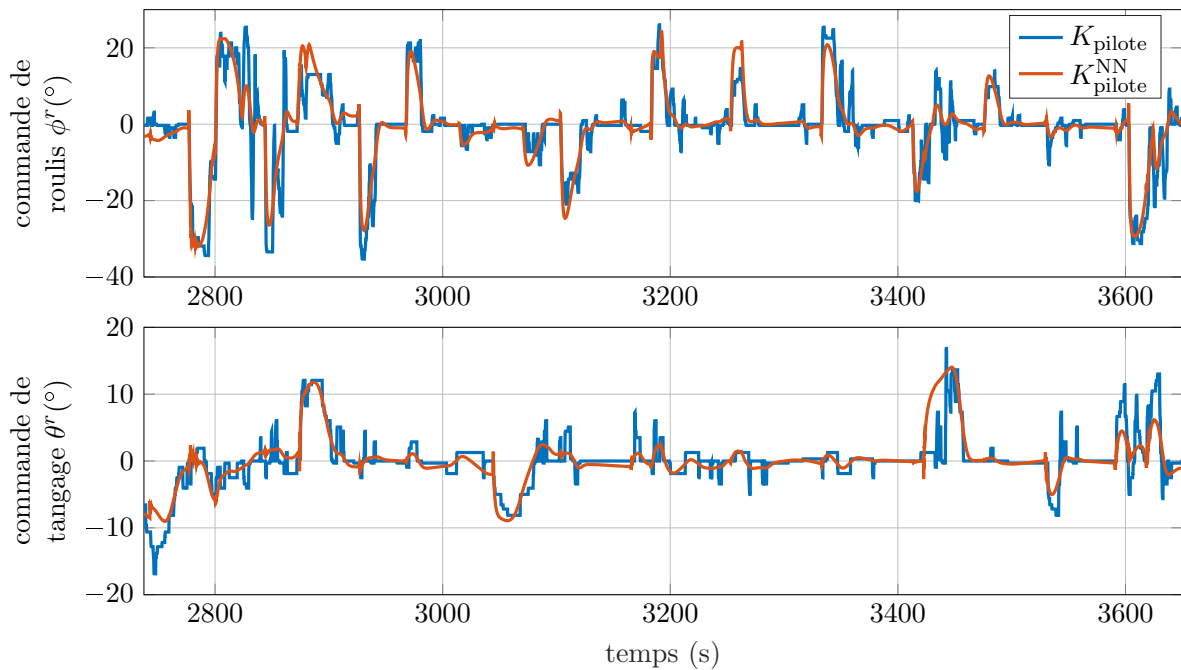


FIGURE 6.9 – Résultat d'apprentissage du contrôleur pour l'imitation du pilote sur les données de validation et de test

L'erreur d'apprentissage du contrôleur est relativement plus élevée dans ce cas d'apprentissage du pilote plutôt que lors de l'apprentissage de l'autopilote déjà implémenté, et cela malgré une structure de réseau plus importante. Comme l'illustre la figure 6.9, le comportement qui doit être assimilé par le réseau est bien plus complexe. Les signaux de commandes sont fortement agités, néanmoins, les réactions du pilote sont en moyenne reproduites par le contrôleur neuronal. Concernant l'apprentissage du système, une structure de réseau similaire au cas de l'apprentissage de l'autopilote est utilisée et l'apprentissage est également réalisé en deux temps. La figure 6.10 démontre une estimation satisfaisante du modèle malgré là encore, une erreur statique qui tend à s'accumuler. Pour rappel, l'établissement du modèle est

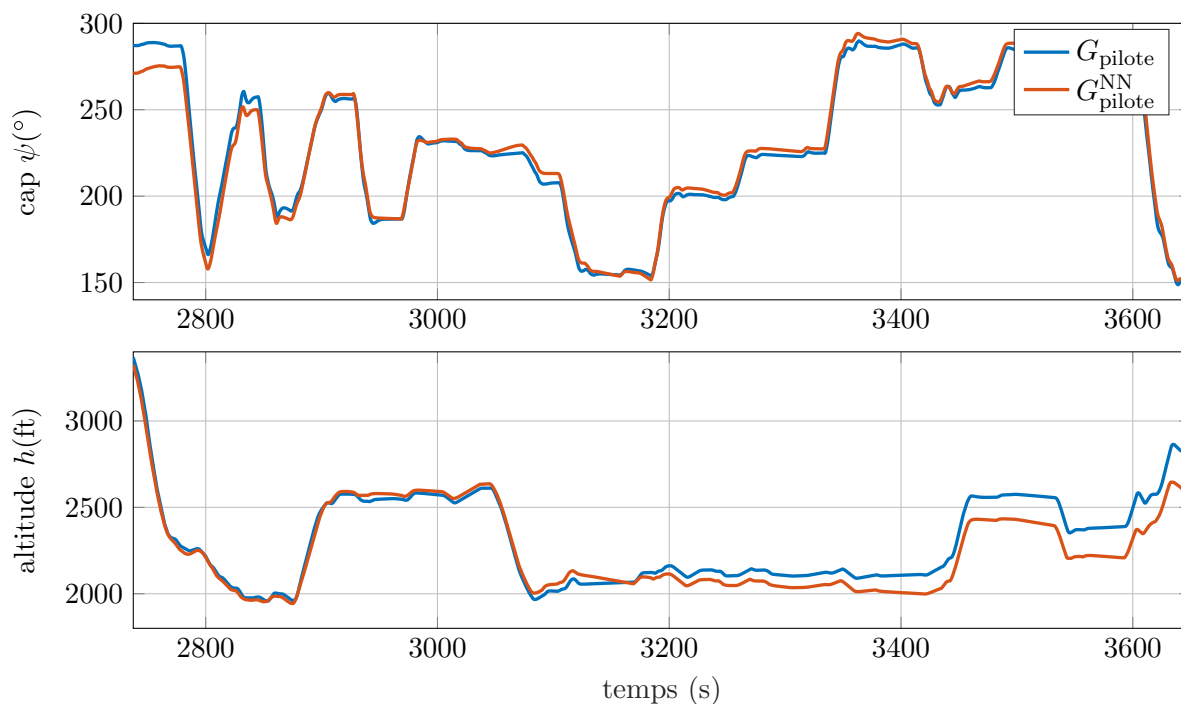


FIGURE 6.10 – Résultat d'apprentissage du contrôleur pour l'imitation du pilote sur les données de validation et de test

uniquement nécessaire pour l'évaluation des marges de stabilité.

Les marges de stabilité de l'asservissement sont calculées à l'aide du contrôleur neuronal K_{pilote}^{NN} et du modèle neuronal G_{pilote}^{NN} selon les marges de disque en entrée et en sortie résumé dans le tableau 6.6. En comparaison avec le cas précédent d'apprentissage de l'autopilote de X-Plane, les marges sont légèrement plus faibles, notamment en sortie. Les marges affichées semblent néanmoins confortables et le contrôleur neuronal est alors co-simulé avec le simulateur. La figure 6.11 expose les données de vol de suivi de cap et d'altitude pour le scénario de test. Les réponses obtenues par le contrôleur issu de l'apprentissage sont comparées à celles acquises indépendamment avec le pilote sur ce même scénario de test. Les commandes coïncident en moyenne comme ce fut le cas lors de l'apprentissage malgré les composantes plus hautes fréquences générées par les réactions d'ajustement du pilote. De l'autre côté, les suivis de consignes se ressemblent fortement. Les réactions propres au pilote sont imitées en grande partie par le contrôleur neuronal notamment en termes de temps de réponse ou de dépassement. La simulation avec le pilote présente de faibles oscillations de suivi de cap, l'autopilote neuronal parvient malgré tout à corriger cette allure. Le suivi de rampes qui n'est pas présent dans les données d'apprentissages diffère légèrement, mais il est correctement effectué. Ainsi, le comportement général du pilote semble donc être assimilé par le pilote automatique.

Bien que le contrôleur obtenu par imitation expose un comportement satisfaisant, nous souhaitons à présent consolider la robustesse du contrôleur via un apprentissage avec optimisation de la stabilité. Un tel choix, se justifie notamment dans la confiance accordée à l'estimation des marges de stabilité. En effet, le modèle neural qui est utilisé pour l'estimation des marges

TABLE 6.6 – Marges de stabilité de l'asservissement obtenu par l'imitation du pilote

Marge en entrée				Marge en sortie			
$\gamma^{DM_u^\mu}$	DM_u^μ	GM_u	PM_u	$\gamma^{DM_y^\mu}$	DM_y^μ	GM_y	PM_y
3.22	0.31	2.7 dB	17.7°	5.96	0.17	1.5 dB	9.6°

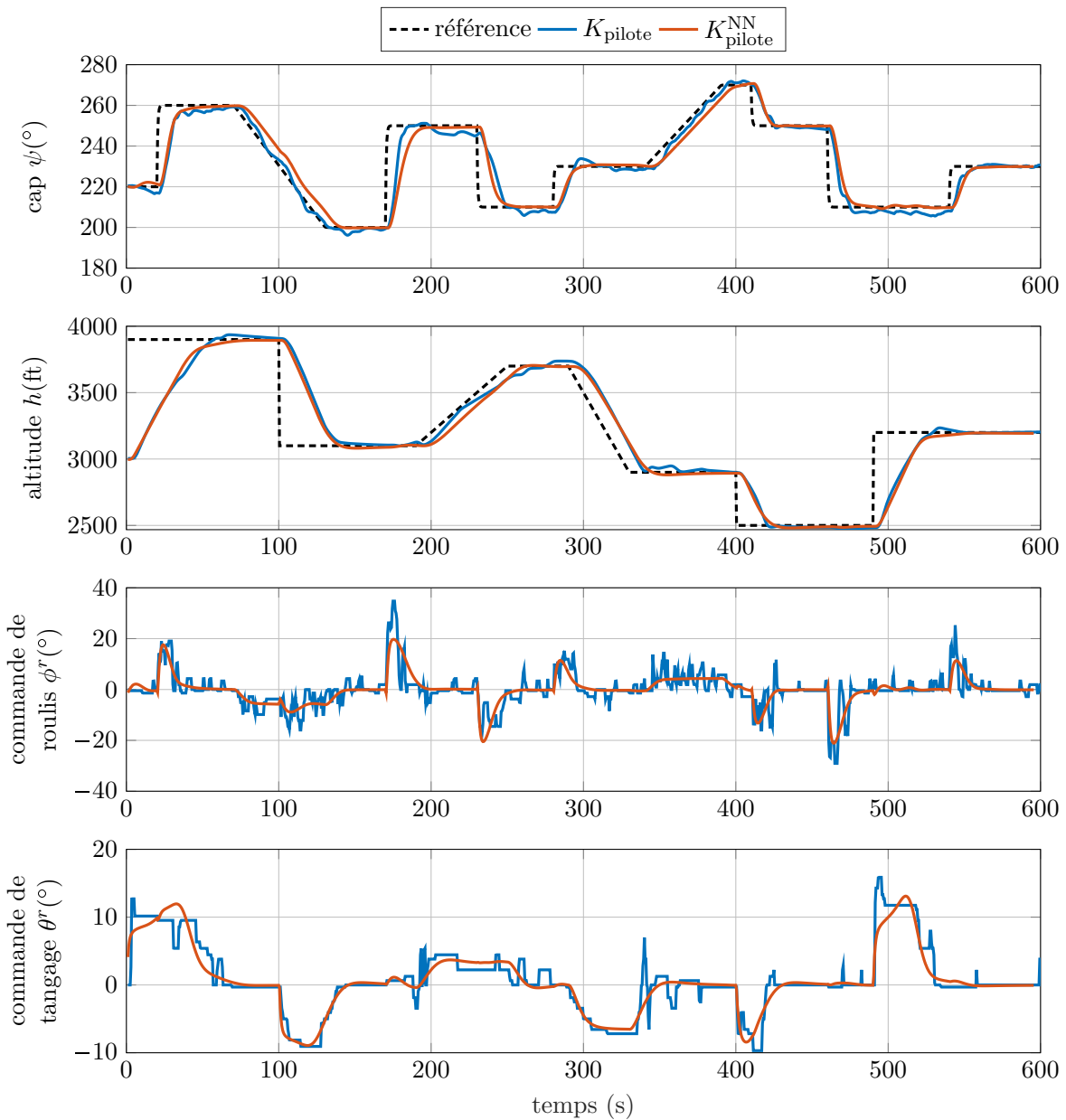


FIGURE 6.11 – Comparaison des simulations du pilote et du contrôleur appris pour un scénario test

présente une erreur d'apprentissage non-négligeable, comme nous l'avons vu précédemment sur les signaux temporels (figure 6.10). Il est alors judicieux de chercher à obtenir des marges de stabilité importantes afin de se prémunir des erreurs de modélisation du système qui est utilisé pour l'évaluation de la robustesse. Dans ce but, un apprentissage multi-objectifs du contrôleur neuronal K^{NN} est réalisé selon le problème d'optimisation suivant :

$$\begin{aligned} \min_{K^{\text{NN}}} F(K^{\text{NN}}) &= (f_{\epsilon}(K^{\text{NN}}), f_{\gamma}(K^{\text{NN}})) \\ f_{\epsilon}(K^{\text{NN}}) &= J_{mse}(K^{\text{NN}}, Z^{dt}) \\ f_{\gamma}(K^{\text{NN}}) &= \gamma, \gamma^{-1} = \min(DM_u^{\mu}, DM_y^{\mu}) \end{aligned} \quad (6.3)$$

L'apprentissage est réalisé conformément à la méthode d'entraînement avec prise en compte de stabilité du chapitre 4 (section 4.4). L'algorithme génétique utilise une population de 100 individus, la structure du réseau du contrôleur est identique à celle précédemment utilisée, ainsi, l'un des individus est initialisé à l'aide du contrôleur déjà entraîné $K_{\text{pilote}}^{\text{NN}}$ tandis que les autres sont initialisés aléatoirement dans l'espace de recherche. Le tableau 6.7 présente les résultats obtenus par l'apprentissage multi-objectifs. Les marges d'une partie des contrôleurs neuronaux synthétisés y sont détaillées, tandis que le front de Pareto associé à l'ensemble optimal de contrôleurs est donné par la figure 6.12. L'ensemble de Pareto regroupe ainsi 34 contrôleurs qui constituent les meilleurs compromis entre robustesse et performance d'imitation. Une solution ensembliste et non-unitaire procure à l'automaticien un degré de liberté supplémentaire qui fait souvent défaut lors d'un apprentissage classique mono-objectif. Le choix lui revient finalement de placer le curseur de robustesse du contrôleur de façon intelligible. Sa décision peut s'appuyer sur une série d'expérimentations des différents contrôleurs afin de sélectionner celui répondant le mieux à ses besoins.

Remarque 6.1. Bien que les calculs soient exécutés en parallèles, la durée d'apprentissage au regard du nombre d'itérations témoigne du caractère coûteux en temps de calcul de l'optimisation. La difficulté réside principalement dans l'estimation des marges de stabilité et les plusieurs secondes nécessaires à leur calcul. Cette contrainte intervient dans la fonction de coût qui est évaluée pour chaque individu et à chaque génération (c'est-à-dire itération).

TABLE 6.7 – Résultats de l'apprentissage multi-objectifs obtenu par l'imitation du pilote

	Structure					Optimisation			
	topologie	n_{in}	n_{out}	σ_1	n_1	n_{γ}	itération	durée	
	SSNNOE	3	4	tanh	10	224	2081	22h41m	
	Marge en entrée				Marge en sortie				Erreur
	$\gamma^{DM_u^{\mu}}$	DM_u^{μ}	GM_u	PM_u	$\gamma^{DM_y^{\mu}}$	DM_y^{μ}	GM_y	PM_y	J_{mse}
K_{pilote}^{*6}	1.10	0.91	8.5 dB	48.9°	1.10	0.91	8.5 dB	48.7°	9.7e-2
K_{pilote}^{*15}	1.17	0.85	7.9 dB	46.1°	1.21	0.83	7.6 dB	44.9°	2.9e-2
K_{pilote}^{*27}	1.36	0.73	6.7 dB	40.2°	2.05	0.49	4.3 dB	27.4°	1.1e-2
K_{pilote}^{*29}	3.00	0.33	2.9 dB	18.9°	3.41	0.29	2.6 dB	16.7°	7.9e-3
K_{pilote}^{*32}	3.22	0.31	2.7 dB	17.6°	4.45	0.22	2.0 dB	12.8°	6.4e-3

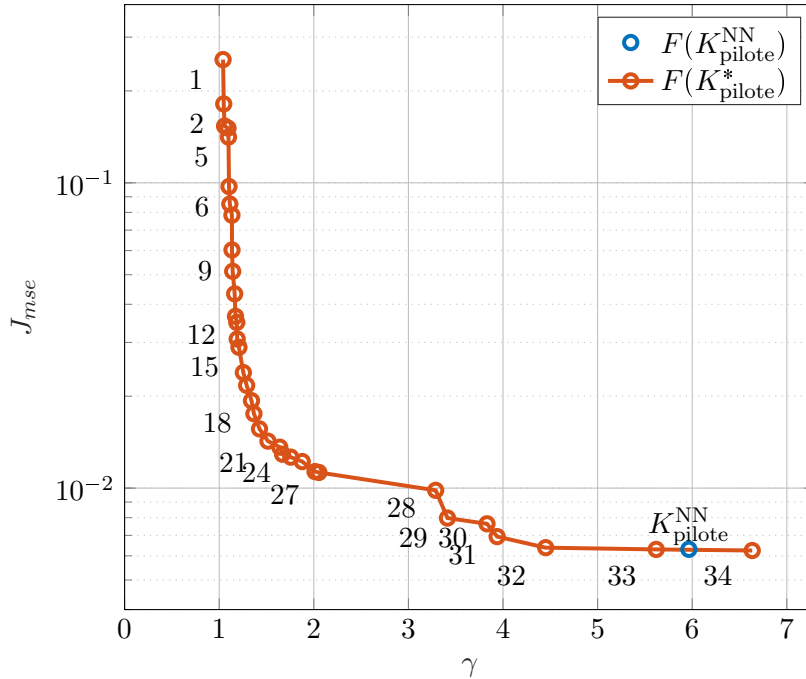


FIGURE 6.12 – Front de Pareto des contrôleurs neuronaux optimaux pour l'imitation du pilote

Un sous-ensemble de contrôleurs de Pareto est co-simulé à l'aide du simulateur de vol pour le scénario de test. La figure 6.13 présente les réponses temporelles aux consignes de cap et d'altitude des contrôleurs en comparaison avec celles obtenues indépendamment avec le pilote. Les contrôleurs manifestant des marges plus robustes respectent un comportement moins semblable à celui de pilote et également moins agressif. En particulier, le contrôleur K_{pilote}^{*6} expose des erreurs statiques de suivi importantes qui témoignent de son excès de robustesse et de l'absence d'intégrateur. Les contrôleurs ayant les meilleures performances d'imitation ont quant à eux un comportement très proche de celui du pilote.

Finalement, l'expérience menée démontre les enjeux de l'apprentissage de contrôleurs robustes. L'autopilote neuronal parvient à assimiler dans sa généralité le comportement propre au contrôleur sujet à l'imitation. Les réactions de l'autopilote de cette seconde expérience diffèrent de celles obtenues lors de la première, de la même manière que celles du pilote se distinguaient de celle de l'autopilote de X-Plane. L'allure légèrement plus agressive du pilote est reproduite, mais selon un degré d'imitation à choisir au regard de propriétés de robustesse. Toutefois, les exigences de marges de stabilité doivent être appréhendées en tout état de cause. L'interprétation des marges doit être mesurée au gré du système et de l'erreur d'apprentissage que présente le modèle. Cet écart n'est pas pris en considération lors de l'estimation de la robustesse, c'est pourquoi chercher à obtenir des marges élevées permet de se prémunir d'un écart important de modélisation au gré d'un certain conservatisme. Néanmoins, les comportements temporels observés renforcent la confiance qui est accordée aux marges de stabilités obtenues pour l'expérience.

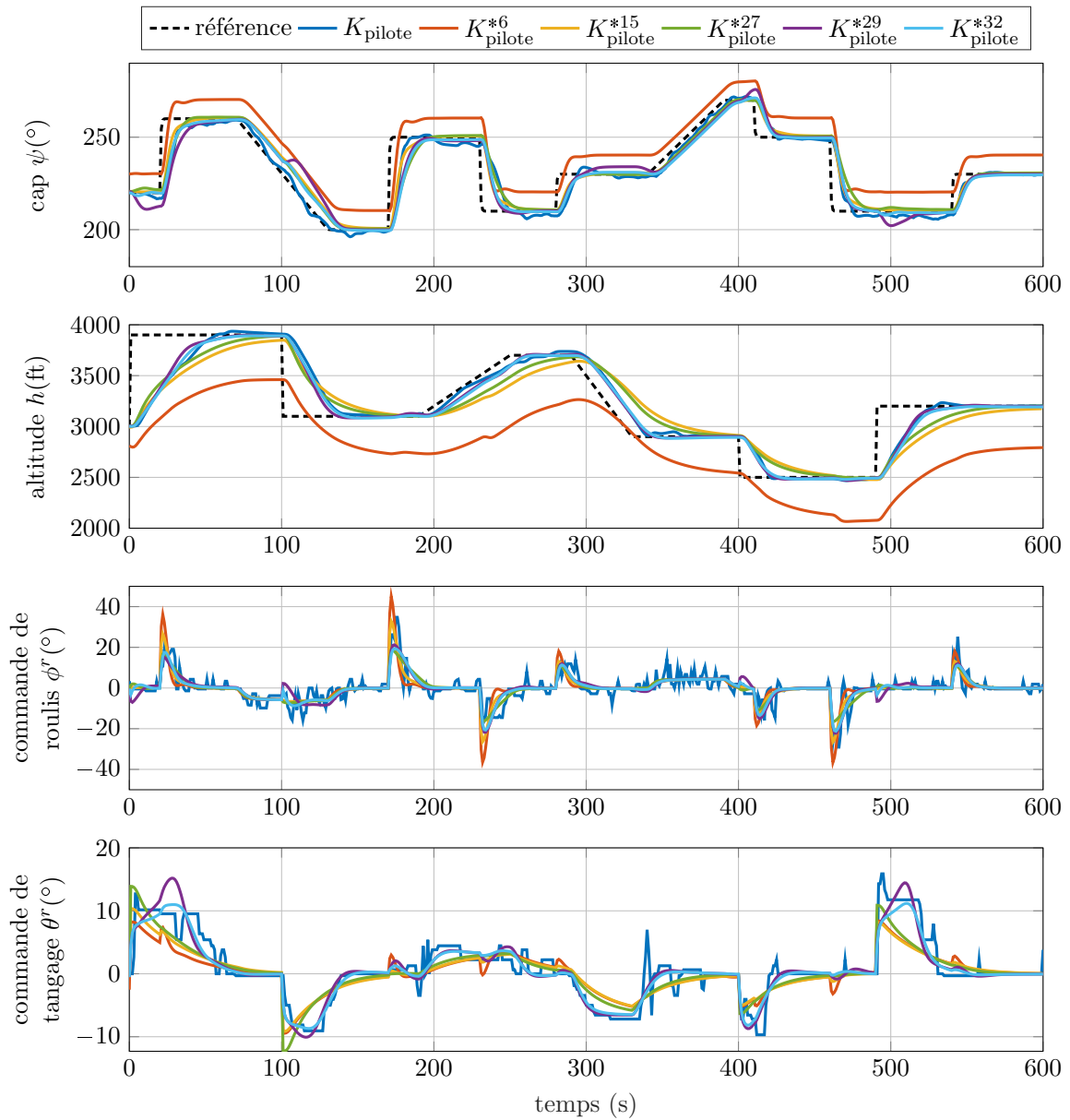


FIGURE 6.13 – Comparaison des simulations du pilote et d'un sous-ensemble des contrôleurs optimaux de Pareto pour un scénario test

6.3.2 Synthèse d'un autopilote de guidage pour le suivi du chemin de vol

6.3.2.1 Description du problème

Nous nous intéressons dans cette partie non plus au guidage d'aéronef en vol libre, mais au guidage selon le suivi d'un chemin de vol prédéfini. En général, la plupart des missions suivent des trajectoires rectilignes entre différents points d'intérêts. Ainsi, un chemin de vol consiste en une série ordonnée de points de passage ou *waypoints* définis en trois dimensions telle que :

$$\text{WP} = \{\text{wp}_1, \text{wp}_2, \dots, \text{wp}_{N_{\text{wp}}}\} \quad (6.4)$$

Entre chaque paire de points de passage consécutif, un chemin en ligne droite est considéré.

La problématique de contrôle abordée consiste à déterminer les angles de guidage commandés (de roulis et de tangage) pour que l'aéronef suive précisément le parcours voulu. L'enjeu du suivi de chemin¹ est de préserver la position de l'appareil sur le parcours au cours du temps. L'objectif de la synthèse de l'autopilote est alors d'assimiler les aptitudes d'un pilote capable d'assurer le guidage de l'avion selon un chemin de vol. La vitesse de l'avion est supposée constante et régulée à l'aide de l'autopilote interne de pilotage de X-Plane. Ce dernier assure également, là encore, le respect des consignes de guidage en agissant sur les gouvernes.

L'avion utilisé dans cette partie est le Vision SF50 (voir la figure 6.14) fabriqué par Cirrus Aircraft qui est disponible par défaut dans X-Plane. Il s'agit d'un avion à réaction monomoteur très léger offrant une vitesse de croisière de 300 nœuds. Dans un besoin de matérialisation du plan de vol, le module d'extension *Air Race Plugin* est employé en complément de X-Plane afin de représenter sous une forme visible les différents points de passage. Ces derniers sont représentés par des portes planes qui doivent être traversées les unes après les autres. Le pilote peut de cette façon, explicitement visualiser la trajectoire à suivre et sa progression à condition que les portes ne soient pas trop espacées.



FIGURE 6.14 – Cirrus Vision SF50

Définition du plan de vol. Les coordonnées des points de passage du plan de vol sont définies dans un repère local associé à la navigation. Le système de coordonnées utilisé dans nos travaux est le repère plan tangent local ENU pour *East North Up* qui est centré sur un point de référence

1. Le suivi de chemin évoqué ici correspond à l'approche souvent référée sous le terme anglais *path following*. Cette dernière se distingue d'une autre approche souvent désignée par *trajectory tracking*, et pour laquelle une paramétrisation temporelle est également imposée puisqu'il s'agit d'amener implicitement l'appareil à une certaine position à un certain moment.

arbitraire x_O . Le premier axe désigné par E pointe dans la direction de l'est, le second axe désigné par N pointe dans la direction du nord, ainsi le plan (x_O, E, N) constitue le plan tangent local et le troisième axe désigné par U est dirigé vers le haut perpendiculairement à ce plan. Dans le système de coordonnées ENU, chaque $j^{\text{ème}}$ point de passage peut alors être décomposé selon les trois dimensions associées :

$$\text{wp}_j = \begin{bmatrix} \text{wp}_j^E \\ \text{wp}_j^N \\ \text{wp}_j^U \end{bmatrix} \quad (6.5)$$

Le plan de vol est défini à l'aide d'une succession d'étapes en trois dimensions qui sont alors reliées par des trajectoires rectilignes. Chaque trajectoire est discrétisée afin de former une succession de segments réguliers de longueur s'approchant au mieux d'une valeur constante arbitraire. Les segments sont alors délimités à l'aide des points de passage qui constituent le plan de vol final. La valeur d'espacement approximative des points de passage retenue dans nos travaux est de 250m, de sorte à obtenir une bonne visualisation de la trajectoire à suivre dans le simulateur.

Logique de validation des points de passage. Lors du déplacement de l'avion en vol, le prochain point de passage à atteindre évolue également au fur et à mesure de la progression de l'avion sur la trajectoire à suivre. Par conséquent, l'algorithme de guidage doit correctement sélectionner le point de passage actuellement ciblé pour lequel les consignes de guidage sont calculées. L'indice du prochain point de passage s'incrémente alors au moment de la validation de ce dernier. La condition de validation d'un point consiste au passage de l'avion à travers la porte associée qui est représentée visuellement dans le simulateur. Bien que les portes soient de largeur et de hauteur données, la condition de validation ne tient pas compte de ce critère et seul le passage à travers le plan est considéré. Finalement, la validation du prochain point de passage wp_{j+1} intervient lors de l'évènement défini comme l'avion a traversé le plan vertical bissecteur entre le segment actuel $[\text{wp}_j, \text{wp}_{j+1}]$ et le segment suivant $[\text{wp}_{j+1}, \text{wp}_{j+2}]$. La figure 6.15 illustre les principaux concepts pour le suivi de chemin de vol.

D'un point de vue mathématique, la validation du prochain point de passage wp_{j+1} est vérifiée en définissant le vecteur bissecteur B_{j+1} associé au segment actuel de vol $S_j = \text{wp}_{j+1} - \text{wp}_j$ et au futur segment $S_{j+1} = \text{wp}_{j+2} - \text{wp}_{j+1}$. Le vecteur bissecteur est alors donné par l'équation :

$$B_{j+1} = \|S_{j+1}\| S_j + \|S_j\| S_{j+1} \quad (6.6)$$

Dans un raisonnement dans le plan Est-Nord, ce vecteur correspond au vecteur directeur de la bissectrice de l'angle formé par les deux segments, c'est-à-dire la droite séparant l'angle en deux angles égaux. Finalement, la condition de validation du point de passage s'exprime en fonction de la position de l'avion z telle que :

$$\begin{bmatrix} B_{j+1}^E \\ B_{j+1}^N \end{bmatrix} \cdot \left(\begin{bmatrix} z^E \\ z^N \end{bmatrix} - \begin{bmatrix} \text{wp}_{j+1}^E \\ \text{wp}_{j+1}^N \end{bmatrix} \right) > 0 \quad (6.7)$$

Cette condition est vérifiée continuellement à chaque pas de temps afin de détecter la validation du prochain point de passage. Si cette condition est satisfaite, l'algorithme de guidage sélectionne alors le prochain point en incrémentant l'indice j associé.

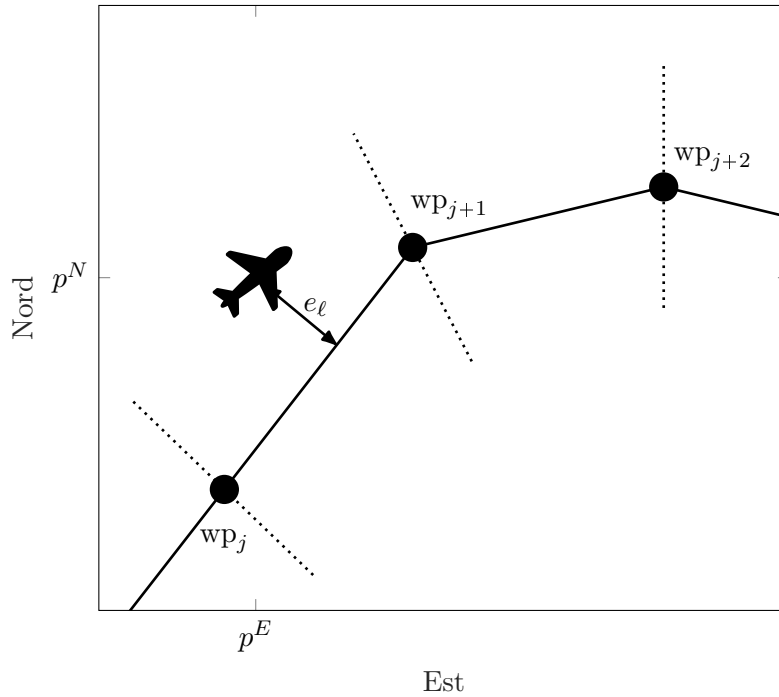


FIGURE 6.15 – Suivi de chemin de vol

Calcul des consignes de guidage et de l'erreur latérale de suivi. Au cours du vol, l'algorithme de suivi de chemin oriente le dispositif de guidage de l'avion, dans le but de maintenir la position au plus proche du chemin à respecter. Pour ce faire, plusieurs consignes de guidage sont alors calculées en fonction des propriétés du prochain point de passage à atteindre wp_{j+1} . Dans le plan latéral, le cap¹ de référence sur lequel l'avion devra s'aligner est donné par l'angle du segment $[wp_j, wp_{j+1}]$ par rapport au nord qui est défini comme suit :

$$\psi_{j+1} = \text{atan2} \left(wp_{j+1}^E - wp_j^E, wp_{j+1}^N - wp_j^N \right) \quad (6.8)$$

où atan2 désigne la fonction arc tangente des quatre quadrants. Concernant le plan longitudinal, l'altitude de référence correspond simplement à l'altitude du prochain point de passage qui est notée h_{j+1} .

Une information supplémentaire est également calculée afin d'évaluer dans le plan Est-Nord le respect du chemin de vol. Il s'agit de l'erreur latérale e_l qui désigne la déviation de la position de l'avion par rapport au chemin à suivre [Nelson *et al.*, 2006]. À cette fin, le position p de l'avion le long du chemin est déterminée par la projection orthogonale de la position actuelle z sur le segment $[wp_j, wp_{j+1}]$ tel que :

$$p = wp_j + \frac{(z - wp_j)^T (wp_{j+1} - wp_j)}{\|wp_{j+1} - wp_j\|^2} (wp_{j+1} - wp_j) \quad (6.9)$$

1. Le cap de référence désigne ici par abus de langage la route de référence c'est-à-dire la direction que doit réellement suivre l'aéronef et non uniquement la direction vers laquelle il doit être orienté.

Finalement, l'erreur latérale est évaluée dans le plan Est-Nord par une valeur relative qui est composée de la distance absolue de l'avion par rapport au chemin et du côté duquel il se trouve :

$$\begin{aligned}
 e_{\ell}^{\text{norme}} &= \left\| \begin{bmatrix} p^E \\ p^N \end{bmatrix} - \begin{bmatrix} z^E \\ z^N \end{bmatrix} \right\| \\
 e_{\ell}^{\text{signe}} &= \text{sign} \left((\text{wp}_{j+1}^E - \text{wp}_j^E)(z^N - \text{wp}_j^N) - (\text{wp}_{j+1}^N - \text{wp}_j^N)(z^E - \text{wp}_j^E) \right) \\
 e_{\ell} &= e_{\ell}^{\text{signe}} e_{\ell}^{\text{norme}}
 \end{aligned} \tag{6.10}$$

6.3.2.2 Génération des données

L'élaboration de données de vol impose une nouvelle fois de déterminer les consignes de vol, c'est-à-dire ici le chemin de vol à respecter afin de solliciter le pilote. Le parcours utilisé comporte plusieurs étapes successives de changement de cap et d'altitude. Les segments sont ensuite discrétisés en une multitude de points de passage avec un espacement d'environ 250m. Le chemin est conçu dans un repère local ENU puis converti en coordonnées géographiques pour être implémenté dans X-Plane sous forme de portes qui indiquent la trajectoire à suivre.

Durant la simulation, le pilote détermine alors les angles de guidage afin de respecter le parcours de vol qui lui est affiché dans le simulateur. Pendant l'ensemble du parcours, la vitesse air de l'avion est régulée autour de 160kt par l'autopilote interne. La figure 6.16 illustre dans le repère ENU les données de vol ainsi collectées. Ces dernières sont par la suite échantillonnées suivant une période régulière de 0.05s, elles sont converties en unité SI, normalisées selon la méthode *min-max*, puis séparées afin de constituer les bases de données d'apprentissage (70%), de validation (15%) et de test (15%).

Les données présentées ici sont obtenues lors d'une session de vol par temps orageux, c'est-à-dire qui comporte de nombreuses rafales de vent, des turbulences et une visibilité réduite. La collecte des données par temps orageux s'est révélée bénéfique dans l'apprentissage de l'autopilote qui sera présenté par la suite. La présence de nombreuses perturbations permet de générer un comportement riche de la part du pilote, et ce, malgré un parcours de vol de faible longueur. De plus, comme nous l'avons constaté dans le chapitre 4, l'insertion de rejet de perturbation dans la base de données permet alors au contrôleur neuronal d'assimiler des comportements plus complexes.

6.3.2.3 Apprentissage de l'autopilote neuronal

Nous réalisons maintenant l'apprentissage de l'autopilote neuronal à partir de la base de données de chemin de vol obtenue avec le pilote. Le premier point abordé est alors de déterminer les vecteurs d'entrées-sorties du contrôleur à identifier ainsi que ceux du système. Les entrées du contrôleur sont choisies comme étant les erreurs de suivi et les dynamiques principales de l'avion selon les deux plans longitudinal et latéral. Néanmoins, des entrées supplémentaires sont nécessaires afin d'apprendre les comportements du pilote. Les réactions du pilote sont en effet, influencées par le chemin qui est à venir, ainsi, il anticipera les changements futur de cap ou d'altitude plusieurs points de passage en avance. Dans le but de parvenir à apprendre ce comportement d'anticipation, des entrées additionnelles sont confectionnées. Ces entrées correspondent à des erreurs de suivi de cap ou d'altitude, mais par rapport à un certain nombre de points

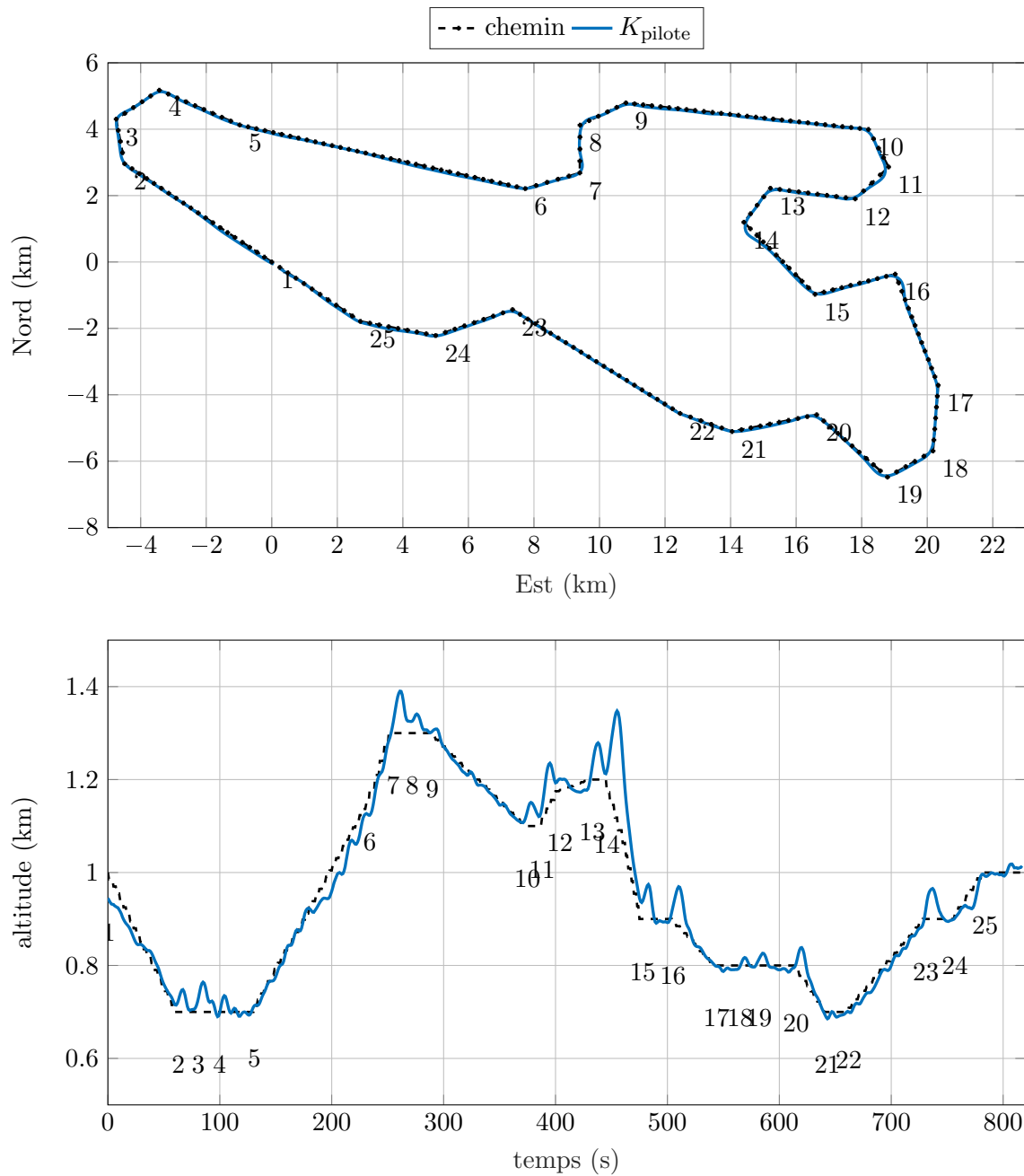


FIGURE 6.16 – Suivi du chemin de vol d'apprentissage obtenu avec le pilote

de passage à venir. Le contrôleur est finalement identifié à l'aide des vecteurs entrées-sorties suivants :

$$\left\{ \begin{array}{l} \psi_{j+1} - \psi \\ \psi_{j+2} - \psi \\ \psi_{j+3} - \psi \\ \psi_{j+4} - \psi \\ \psi_{j+5} - \psi \\ -\phi \\ h_{j+1} - h \\ h_{j+2} - h \\ -\theta \\ e_\ell \end{array} \right\}, \left[\begin{array}{l} \phi_r \\ \theta_r \end{array} \right] \quad (6.11)$$

Les vecteurs d'entrées-sorties du système sont alors conformément sélectionnés comme étant :

$$\left\{ \left[\begin{array}{l} \phi_r \\ \theta_r \end{array} \right], \left[\begin{array}{l} \psi \\ \phi \\ h \\ \theta \end{array} \right] \right\} \quad (6.12)$$

Nous détaillons à présent les résultats d'apprentissage obtenus. Concernant les contrôleurs neuronaux, aucun de ceux entraînés à l'aide des méthodes mono-objectif n'a pu apporter de résultats concluants. Les contrôleurs ont un comportement instable lorsqu'ils sont co-simulés avec X-Plane y compris sur le chemin de vol d'apprentissage et leur manque de robustesse est confirmé par l'estimation de marges de stabilité très modestes. Nous développons alors directement les résultats obtenus par la méthode d'apprentissage avec prise en compte de la stabilité.

L'apprentissage multi-objectifs est effectué en respectant le problème d'optimisation décrit par l'équation 6.3. Le système est dans un premier temps, identifié à l'aide d'un modèle neuronal et de l'algorithme de Levenberg-Marquardt. Le modèle retenu est introduit par le tableau 6.8, il représente un élément nécessaire à l'évaluation des marges de stabilité qui constitue le second critère de l'optimisation multi-objectifs. Afin de définir la boucle d'asservissement dont la robustesse sera évaluée, la dynamique de l'erreur latérale est linéarisée. Cette erreur représente la distance par rapport au chemin de référence, sa dynamique peut être obtenue en fonction de l'écart de cap entre l'aéronef et celui de référence du prochain point de passage wp_{j+1} suivant :

$$\dot{e}_\ell = v \sin(\psi_{j+1} - \psi) \quad (6.13)$$

La vitesse de l'appareil est alors supposée égale à une vitesse de référence v_r constante pour la linéarisation, ainsi la boucle de l'erreur latérale est définie selon l'intégration numérique de la dynamique suivante :

$$\dot{e}_\ell = v_r(\psi_{j+1} - \psi) \quad (6.14)$$

L'apprentissage du contrôleur est finalement mené en multi-objectifs pour tenir compte des critères de performance et de robustesse. La structure neuronale à employer est préalablement appréhendée à l'aide d'apprentissages mono-objectif. L'optimisation multi-objectifs est accomplie au moyen d'un algorithme génétique de 80 individus, dont 20 sont initialisés par un pré-apprentissage mono-objectif alors que les autres sont initialisés aléatoirement. Trois tirages, c'est-à-dire trois phases d'optimisation sont réalisés successivement pour 3000 itérations, les

fronts de Pareto résultant sont unifiés en conservant uniquement les solutions dominantes. Les résultats obtenus sont présentés dans le tableau 6.9 et le front de Pareto est illustré par la figure 6.17. L'ensemble des meilleurs compromis entre termes de robustesse et de performance d'imitation regroupe finalement 29 autopilotes pour lesquels il convient de vérifier le bon comportement temporel.

TABLE 6.8 – Résultats d'apprentissage du système pour le suivi de chemin de vol

	Structure						Résultats				
	topologie	n_{in}	n_{out}	σ_1	n_1	n_W	iter.	durée	err. app.	err. val.	err. test
G^{NN}	SSNNARX	6	5	tanh	5	335	112	10s	7.0e-6	3.5e-6	2.4e-5
	SSNNOE	6	5	tanh	5	335	773	12m	1.0e-3	3.6e-3	6.2e-3

TABLE 6.9 – Résultats de l'apprentissage multi-objectifs des autopilotes de suivi de chemin de vol obtenus par l'imitation du pilote

	Structure						Optimisation			
	topologie	n_{in}	n_{out}	σ_1	n_1	n_W	itération	durée		
	SSNNOE	4	5	tanh	10	752	3×3000	4j21h35m		
		Marge en entrée				Marge en sortie				Erreur
		$\gamma^{DM_u^\mu}$	DM_u^μ	GM_u	PM_u	$\gamma^{DM_y^\mu}$	DM_y^μ	GM_y	PM_y	J_{mse}
K_{pilote}^{*6}		2.82	0.36	3.1 dB	20.1°	2.84	0.35	3.1 dB	20.0°	5.3e-2
K_{pilote}^{*10}		3.21	0.31	2.7 dB	17.7°	3.53	0.28	2.5 dB	16.1°	3.9e-2
K_{pilote}^{*18}		2.75	0.36	3.2 dB	20.6°	6.85	0.15	1.3 dB	8.3°	2.8e-2
K_{pilote}^{*22}		2.76	0.36	3.2 dB	20.5°	15.1	0.07	0.6 dB	3.8°	2.7e-2
K_{pilote}^{*24}		4.00	0.24	2.2 dB	14.2°	662	0.00	0.0 dB	0.1°	1.6e-2
K_{pilote}^{*29}		6.32	0.16	1.4 dB	9.0°	2805	0.00	0.0 dB	0.0°	7.9e-3

6.3.2.4 Expérimentations

Les autopilotes neuronaux conçus par l'optimisation multi-objectifs sont à présent co-simulés pour des vols autonomes sur le simulateur. Le suivi du chemin de vol est implémenté conformément à la figure 6.18, les éléments nécessaires au suivi du chemin sont entièrement calculés dans MATLAB et X-Plane assure la simulation du système.

Certains contrôleurs du front de Pareto sont alors simulés sur le chemin de vol d'apprentissage, dans des conditions météorologiques calmes et pour une valeur de vitesse de l'avion similaire à celle des données, c'est-à-dire 160 nœuds. La figure 6.19 affiche les résultats selon le suivi latéral du parcours et longitudinal en fonction de la progression sur le chemin. Les comportements obtenus pour les différents points du front de Pareto démontrent le compromis réalisé par chaque contrôleur entre la robustesse et la performance d'imitation. L'autopilote le plus robuste ici K_{pilote}^{*6} , présente une erreur de suivi latérale et longitudinale très importante,

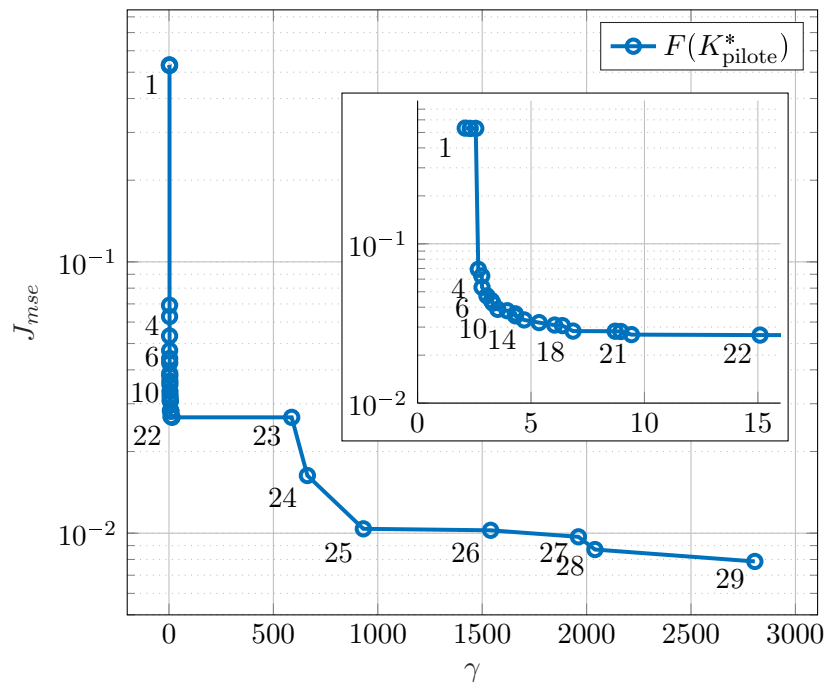


FIGURE 6.17 – Fronts de Pareto des autopilotes neuronaux optimaux pour le suivi de chemin de vol

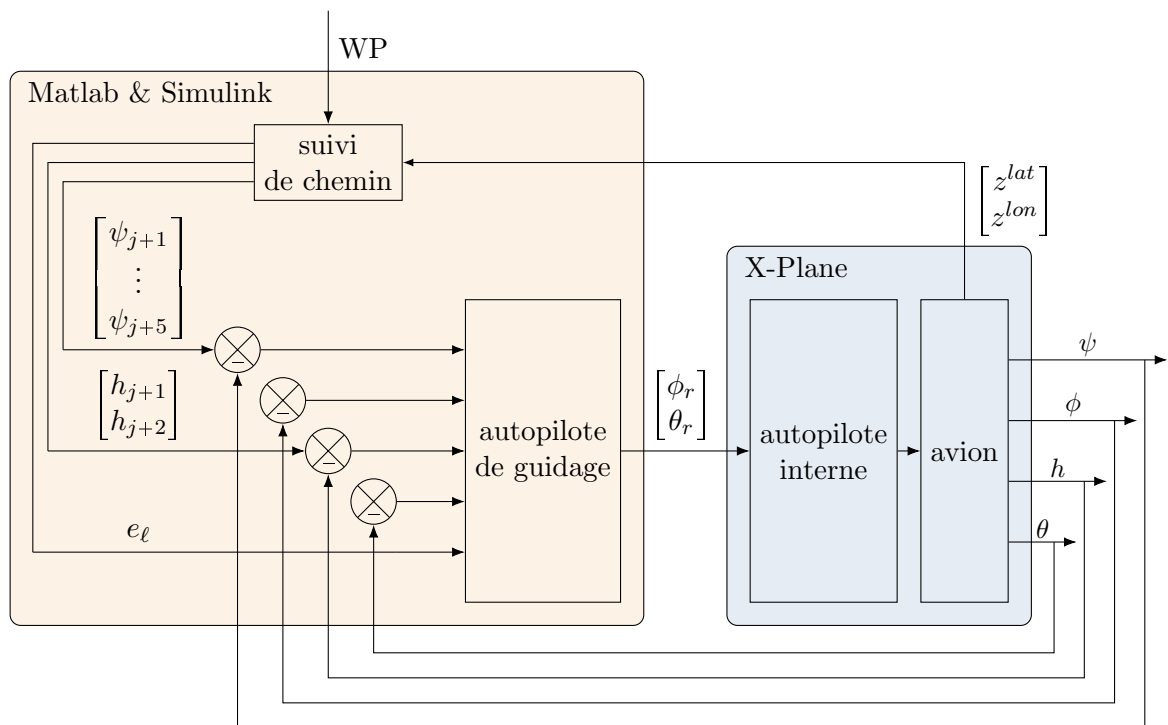


FIGURE 6.18 – Schéma de simulation de la boucle d'asservissement de guidage pour le suivi de chemin de vol

synonyme de son caractère robuste et peu intégrateur. À l'inverse, les autopilotes K_{pilote}^{*24} et K_{pilote}^{*29} , dont l'erreur d'apprentissage est la plus fiable, exposent un comportement extrêmement proche de celui du pilote pour les débuts de simulation (y compris de ses défauts comme le montre par exemple l'erreur de suivi d'altitude pour les premiers points de passage qui est similaire à celle du pilote de la figure 6.16). Néanmoins, leur manque de robustesse ne leur permet pas d'assurer la stabilité du vol jusqu'à la fin de la simulation. Finalement, les autopilotes K_{pilote}^{*10} et K_{pilote}^{*22} délimitent au vu de leurs réponses temporelles satisfaisantes, l'intervalle des compromis qui peuvent être envisagés.

Dans le but d'évaluer les propriétés de généralisation, les autopilotes sont mise à l'épreuve sur un autre chemin de vol. Ainsi, un nouveau chemin de test est généré et le suivi est réalisé dans les mêmes conditions que précédemment. Les résultats sont donnés par la figure 6.20 qui illustre également des comportements en accord avec le front de Pareto. Les autopilotes excessivement robustes présentent une erreur de suivi importante, les compromis plus performants mais avec des marges de stabilité raisonnables offrent un comportement satisfaisant, tandis que les autopilotes les plus performants mais moins robustes suivent parfaitement le chemin jusqu'à un certain point où l'instabilité se produit.

Pour finir, une expérimentation supplémentaire est réalisée afin d'observer les réactions des autopilotes lorsque le guidage opère hors de l'enveloppe de vol d'apprentissage. Pour ce faire, une variation paramétrique du système est effectuée via un changement de vitesse de consigne de l'avion. Il est alors demandé à l'autopilote interne de maintenir une vitesse plus élevée de 220 nœuds. Les simulations des contrôleurs les plus pertinents sur le front de Pareto sont présentées par la figure 6.21 pour le chemin de vol d'apprentissage. Les réponses des contrôleurs et leur agressivité dans le respect du suivi de chemin, concordent là encore avec leur degré de performance et robustesse. Néanmoins, les résultats mettent en avant la limite des autopilotes et leur sensibilité face aux fortes variations paramétriques. La vitesse de l'avion influe fortement sur les lois de guidage, or l'imitation a été réalisée pour une vitesse constante. Afin de consolider l'autopilote neuronal et de maîtriser davantage de scénarios de vol, il est alors possible d'envisager une extension par multi-modèle comme nous le présentons dans la section suivante.

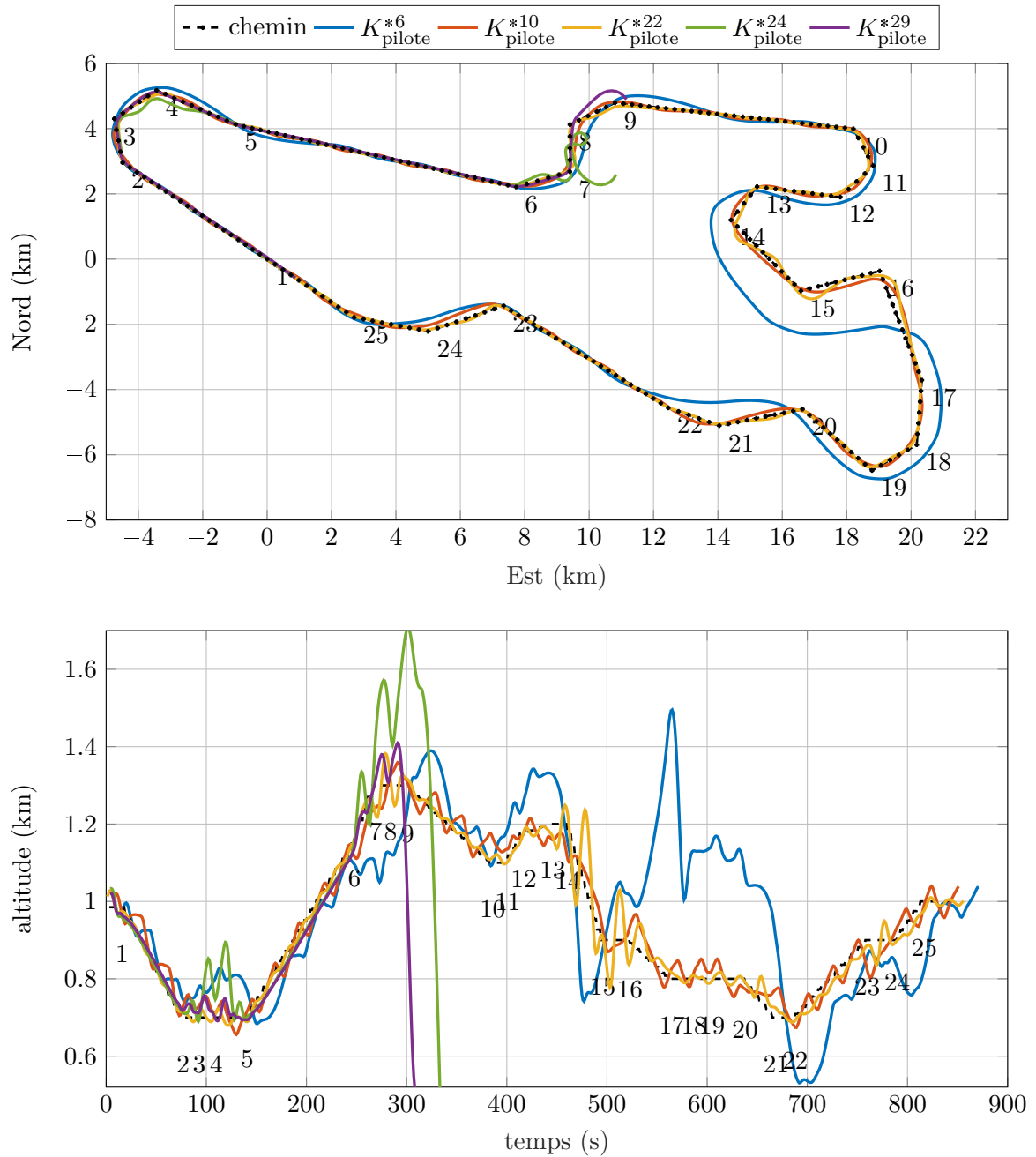


FIGURE 6.19 – Simulations d'un sous-ensemble des autopilotes optimaux pour le suivi du chemin appris

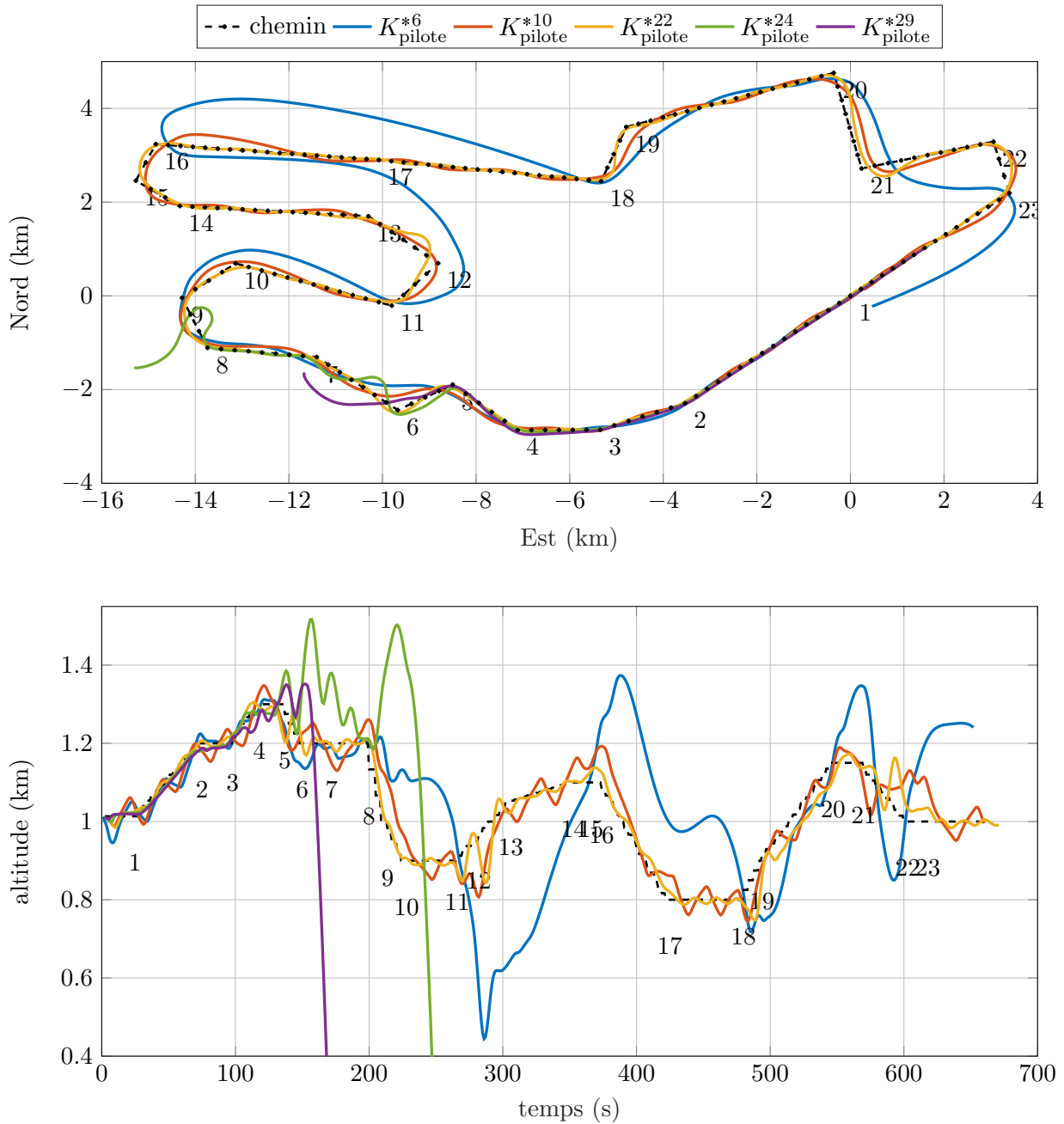


FIGURE 6.20 – Simulations d'un sous-ensemble des autopilotes optimaux pour le suivi d'un chemin de test

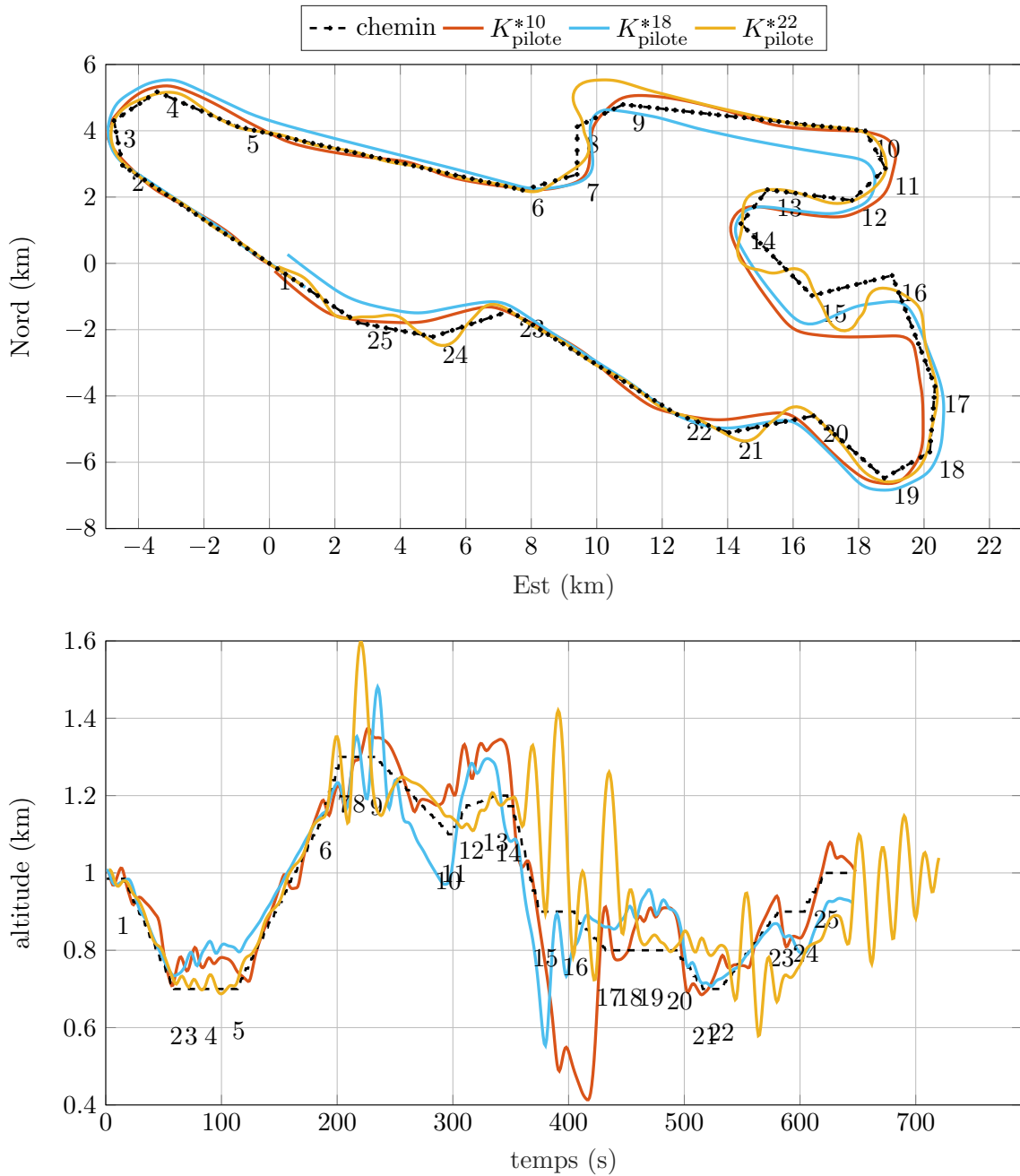


FIGURE 6.21 – Simulations d'un sous-ensemble des autopilotes optimaux pour une vitesse plus élevée

SECTION 6.4

Module de déploiement de l'autopilote par multi-modèle

Cette section illustre dans un contexte aéronautique, l'approche multi-modèle qui a été développée au chapitre 5. La méthode permet alors d'implémenter plusieurs autopilotes neuronaux qui ont été élaborés séparément et qui répondent à des besoins différents. Bien que les apports théoriques manquent encore à la démarche, cette dernière offre des perspectives intéressantes qui sont confirmées par l'application.

6.4.1 Déploiement d'un NMMAC de guidage en vol libre**6.4.1.1 Description et intérêt du problème**

Cette première sous-section reprend la problématique étudiée dans la partie 6.3.1. L'objectif est de développer un contrôleur neuronal de guidage qui permet de diriger l'avion selon un cap et à une altitude souhaités. Nous souhaitons à présent agrandir l'enveloppe de vol de l'autopilote pour d'autres vitesses de l'aéronef puisque ce paramètre était jusqu'à présent maintenue proche d'une valeur constante. À des fins d'illustrations de la méthode multi-modèle, nous supposons que la vitesse de l'avion n'est pas mesurée et exploitée par l'autopilote de guidage. En conséquence, un MMAC neuronal (NMMAC) est réalisé sur différentes valeurs de vitesses qui sont sélectionnées à l'intérieur de l'enveloppe de vol de l'avion.

L'aéronef utilisé est le Vision SF50 de Cirrus Aircraft qui de par sa puissance permet d'envisager des variations de vitesse plus importantes. L'appareil dispose d'un pilote automatique déjà implémenté dans le simulateur et notamment d'un directeur de vol qui constitue le contrôleur sujet à l'imitation. La vitesse air de l'avion, bien qu'elle soit connue par le directeur de vol d'origine, elle est supposée n'être pas mesurée pour l'expérience et ne peut donc constituer une entrée du contrôleur neuronal. Ainsi, une approche multi-modèle constituée de trois points de fonctionnement est réalisée conformément aux trois vitesses air suivantes : $v_1 = 130\text{kt}$, $v_2 = 160\text{kt}$ et $v_3 = 190\text{kt}$. Finalement, le but de l'expérience est qu'à partir de bases de données obtenues indépendamment pour différents scénarios de vol (ici trois phases de vol à différentes vitesses), de reproduire l'autopilote de guidage dans une version neuronale et organisée selon un NMMAC comme le montre la figure 6.22.

6.4.1.2 Apprentissage multi-modèle de l'autopilote de X-Plane

La première phase de l'imitation consiste à la génération de bases de données dont les signaux sont suffisamment riches pour l'apprentissage. Des références de cap entre 150° et 300° et d'altitude entre 3500ft et 6000ft sont envoyées au simulateur dont le pilote automatique a pour tâche de contrôler l'avion en se conformant à ces références. Les consignes de navigation sont sous forme d'échelons successifs dont les propriétés sont identiques à celles évoquées dans la section 6.3.1.2. Trois phases de vol sont réalisées par temps calme au cours desquelles l'autopilote interne régule la vitesse de l'avion autour des vitesses de consigne $v_1 = 130\text{kt}$, $v_2 = 160\text{kt}$ et $v_3 = 190\text{kt}$. Chacune des expériences dure 45min et les données sont enregistrées selon un temps d'échantillonnage de 0.05s. Les bases de données sont finalement constituées après pré-traitement, c'est-à-dire normalisation (*min-max*) et séparation en trois ensembles.

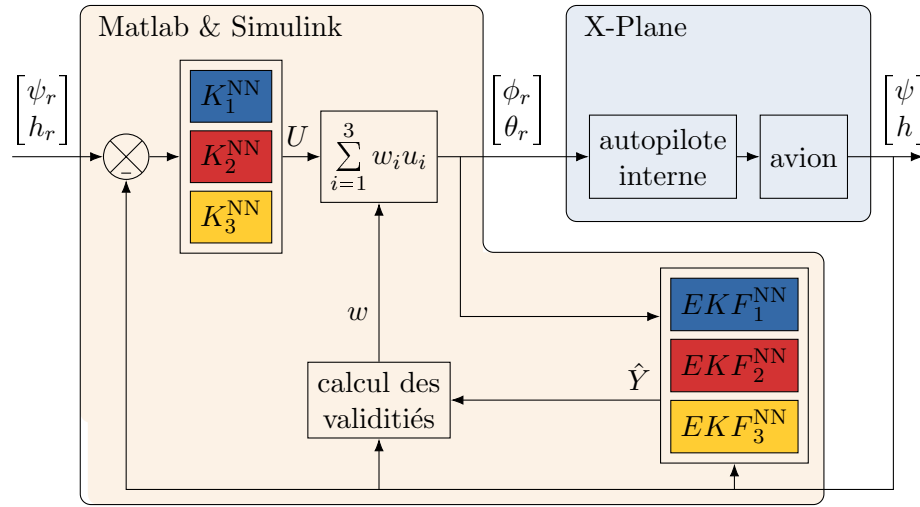


FIGURE 6.22 – Autopilote NMMAC de guidage

Une fois les données disponibles, l'apprentissage hors ligne est effectué triplement, c'est-à-dire pour chaque point de fonctionnement. Deux modèles neuronaux sont obtenus pour chaque cas, le contrôleur selon les vecteurs entrées-sorties définis par l'équation 6.1 et le système dont les entrées-sorties sont données en 6.2. Les paramètres utilisés pour l'apprentissage sont une fois encore l'algorithme Levenberg-Marquardt et un critère d'arrêt prématuré de 300 itérations. Les résultats des meilleurs réseaux retenus sont donnés dans le tableau 6.10 pour les trois vitesses.

TABLE 6.10 – Résultats d'apprentissage des trois paires modèles-contrôleurs pour l'imitation de l'autopilote de X-Plane

		Structure					Résultats					
		topologie	n_{in}	n_{out}	σ_1	n_1	n_W	iter.	durée	err. app.	err. val.	err. test
K_1^{NN}	SSNNARX	3	1	tanh	5	103	212	13s	$2.2e-7$	$2.5e-7$	$2.4e-7$	
	SSNNOE	3	1	tanh	5	103	302	3m22s	$7.1e-3$	$4.7e-4$	$9.7e-3$	
G_1^{NN}	SSNNARX	3	1	tanh	15	253	388	2m12s	$3.3e-12$	$5.4e-12$	$7.6e-12$	
	SSNNOE	3	1	tanh	15	253	303	11m16s	$5.6e-4$	$4.9e-3$	$3.5e-2$	
K_2^{NN}	SSNNARX	3	1	tanh	5	103	231	18s	$2.5e-7$	$2.9e-7$	$1.8e-7$	
	SSNNOE	3	1	tanh	5	103	159	1m56s	$6.8e-3$	$7.1e-4$	$8.4e-3$	
G_2^{NN}	SSNNARX	3	1	tanh	15	253	345	1m46s	$7.4e-12$	$8.4e-12$	$1.7e-11$	
	SSNNOE	3	1	tanh	15	253	302	11m12s	$3.6e-4$	$1.0e-3$	$2.8e-2$	
K_3^{NN}	SSNNARX	3	1	tanh	5	103	1111	52s	$2.8e-7$	$2.3e-7$	$2.7e-7$	
	SSNNOE	3	1	tanh	5	103	20	10s	$9.6e-3$	$7.1e-4$	$1.3e-3$	
G_3^{NN}	SSNNARX	3	1	tanh	15	253	423	1m18s	$5.9e-11$	$3.5e-11$	$7.0e-11$	
	SSNNOE	3	1	tanh	15	253	780	18m20s	$8.0e-6$	$3.7e-4$	$5.9e-3$	

Selon la méthode d'apprentissage de contrôleur que nous avons employée jusqu'à présent, la robustesse des différents asservissements doit maintenant être estimée par le calcul des marges de

stabilité. Néanmoins, du point de vue chronologique, les travaux présentés dans cette partie sont antérieurs à ceux qui ont permis le développement des outils d'évaluation des marges de stabilité présentés dans le chapitre 4. Ainsi, les marges sont évaluées ultérieurement pour la cohérence de la méthode présentée. Le tableau 6.11 donne les marges de disque en entrée et en sortie qui sont calculées pour les trois boucles d'asservissement de modèle-contrôleur. Les résultats obtenus indiquent que les asservissements dans le premier et le dernier cas sont faiblement robustes et que le second cas témoigne d'une robustesse presque inexistante. Ces conclusions bien pessimistes illustrent les limites de la méthode d'évaluation des propriétés de robustesse. Premièrement, de façon intrinsèque, l'estimation des marges de stabilité souffre d'un fort conservatisme. L'utilisation d'outils destinés au domaine linéaire impose alors d'encadrer les non-linéarités par des paramètres incertains bornés. Si les non-linéarités sont fortement prononcées, alors l'encadrement qui en résulte est d'autant plus large et les résultats potentiellement plus pessimistes. La seconde difficulté réside dans l'évaluation numérique de la méthode. Le réseau peut être sujet au mauvais conditionnement numérique lors du calcul des marges. Cette sensibilité peut notamment être empirée lorsque le réseau comporte davantage de neurones.

TABLE 6.11 – Marges de stabilité des trois paires modèles-contrôleurs pour l'imitation de l'autopilote de X-Plane

Modèle- contrôleur	Marge en entrée				Marge en sortie			
	$\gamma^{\text{DM}_u^\mu}$	DM_u^μ	GM_u	PM_u	$\gamma^{\text{DM}_y^\mu}$	DM_y^μ	GM_y	PM_y
n°1	4.30	0.23	2.0 dB	13.2°	6.84	0.15	1.3 dB	8.4°
n°2	161	0.01	0.1 dB	0.4°	194	0.01	0.0 dB	0.3°
n°3	9.06	0.11	1.0 dB	6.3°	9.10	0.11	1.0 dB	6.3°

6.4.1.3 Implémentation du NMMAC pour le contrôle autonome

Le NMMAC est à présent élaboré à partir des réseaux de neurones préalablement entraînés. Les contrôleurs sont directement utilisés sous leur forme neuronale pour constituer la banque de contrôleurs. Concernant la banque d'estimateurs, les éléments la constituant sont fondés sur les modèles neuronaux obtenus. Des EKF discrets sont alors conçus à partir des représentations d'états des systèmes accordées par la structure SSNN. La dernière partie du MMAC est la logique de sélection qui permet de calculer les validités (c'est-à-dire les poids) associées aux paires modèles-contrôleurs afin de former la commande globale. Les validités sont calculées par l'estimateur ultérieur de probabilité (PPE) dont la dynamique est couplée à celle des filtres de Kalman à l'aide des matrices de covariance de l'innovation $S_i(k)$. Les validités sont alors obtenues en tirant profit des erreurs d'estimation $\varepsilon_i(k) = y(k) - \hat{y}_i(k)$ selon l'équation récursive bayésienne :

$$w_i(k+1) = \frac{\left[(2\pi)^{n_y} \det(S_i(k)) \right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \varepsilon_i^T(k) S_i^{-1}(k) \varepsilon_i(k) \right) w_i(k)}{\sum_{j=1}^3 \left[(2\pi)^{n_y} \det(S_j(k)) \right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \varepsilon_j^T(k) S_j^{-1}(k) \varepsilon_j(k) \right) w_j(k)} \quad (6.15)$$

Les EKF sont des observateurs non-linéaires qui peuvent être facilement mis en œuvre, mais leur réglage peut se révéler difficile. Un premier réglage est obtenu en utilisant le MMAE seul et en alimentant le MMAE avec les données de vol utilisées pour l'apprentissage. Le réglage des EKF est ensuite affiné dans un second temps en utilisant le MMAE en co-simulation avec X-Plane. Finalement, le réglage suivant est retenu pour chaque $i^{\text{ième}}$ estimateur : la covariance du bruit de processus $Q_i = \text{diag}([10^{-4} \ 10^{-3} \ 10^{-4} \ 10^{-3}])$, la covariance du bruit d'observation $R_i = \text{diag}([1 \ 10])$ et la covariance de l'estimation initiale $P_i(k=0) = 10^3$. La covariance de l'innovation S_i est ensuite utilisée par le PPE pour calculer les validités avec les paramètres suivants : un seuil $\delta = 0,01$ et des poids initiaux $w_i(k=0) = 1/3$.

Le système de contrôle NMMAC est finalement co-simulé avec X-Plane, conformément à ce qui a été précédemment présenté dans la figure 6.22. L'autopilote neuronal assure le suivi des consignes de cap et d'altitude du scénario de test. À la distinction des cas précédents, différentes consignes de vitesse sont cette fois-ci mandatées au cours de la simulation. La vitesse de l'avion est contrôlée par l'autopilote interne inclus dans le simulateur, mais elle n'est pas prise en compte par l'autopilote neuronal dans MATLAB. La figure 6.23 détaille les résultats de simulation obtenus pour l'autopilote NMMAC de guidage et la figure 6.24 affiche une comparaison des réponses par rapport à l'autopilote à imiter de X-Plane. Les résultats montrent des performances satisfaisantes dans le contrôle du cap et de l'altitude qui correspondent à l'autopilote imité, des signaux de contrôle appropriés suivis par l'autopilote interne de l'avion, et une convergence des validités en fonction de la vitesse inconnue pour l'autopilote neuronal.

6.4.2 Déploiement d'un NMMAC de guidage pour le suivi de chemin de vol

6.4.2.1 Description du problème

Cette partie reprend les travaux présentés dans la section 6.3.2 concernant le développement d'un autopilote de guidage capable de suivre un chemin de vol prédéfini par une succession de points de passage. Nous souhaitons maintenant illustrer dans ce contexte l'intérêt de l'approche multi-modèle afin de consolider et d'adapter l'autopilote à différentes situations. L'autopilote a été développé pour une vitesse constante non-mesurée, ainsi comme il a été expérimenté dans la section 6.3.2.4, des fortes variations paramétriques comme des variations de vitesse impactent fortement les lois de guidage. Nous proposons donc d'agrandir les capacités de l'autopilote par l'ajout d'un second modèle issu d'une vitesse d'aéronef plus importante, mais également de considérer un troisième modèle associé à un scénario de panne choisi pour l'expérience. Ce dernier modèle correspond à l'événement d'arrachement du volet gauche de l'aéronef ce qui provoque un comportement aérodynamique asymétrique qui peut engendrer des problèmes de stabilités. Finalement, la banque de modèles considérée pour cette partie se compose des trois modèles $\{M_{160}, M_{220}, M_{\text{volet}}\}$ obtenus, pour la vitesse air v et l'état du volet gauche arraché ou non θ_{volet} , comme suit :

$$\begin{aligned} \text{Modèle 1 :} & \quad M_{160} = \{v = 160 \text{ kt}, \theta_{\text{volet}} = 0\} \\ \text{Modèle 2 :} & \quad M_{220} = \{v = 220 \text{ kt}, \theta_{\text{volet}} = 0\} \\ \text{Modèle 3 :} & \quad M_{\text{volet}} = \{v = 160 \text{ kt}, \theta_{\text{volet}} = 1\} \end{aligned} \tag{6.16}$$

L'objectif de l'expérience est donc finalement de développer un NMMAC afin d'imiter un pilote capable de diriger l'aéronef (Cirrus Vision SF50) le long du chemin de vol pour ces différentes conditions de vol.

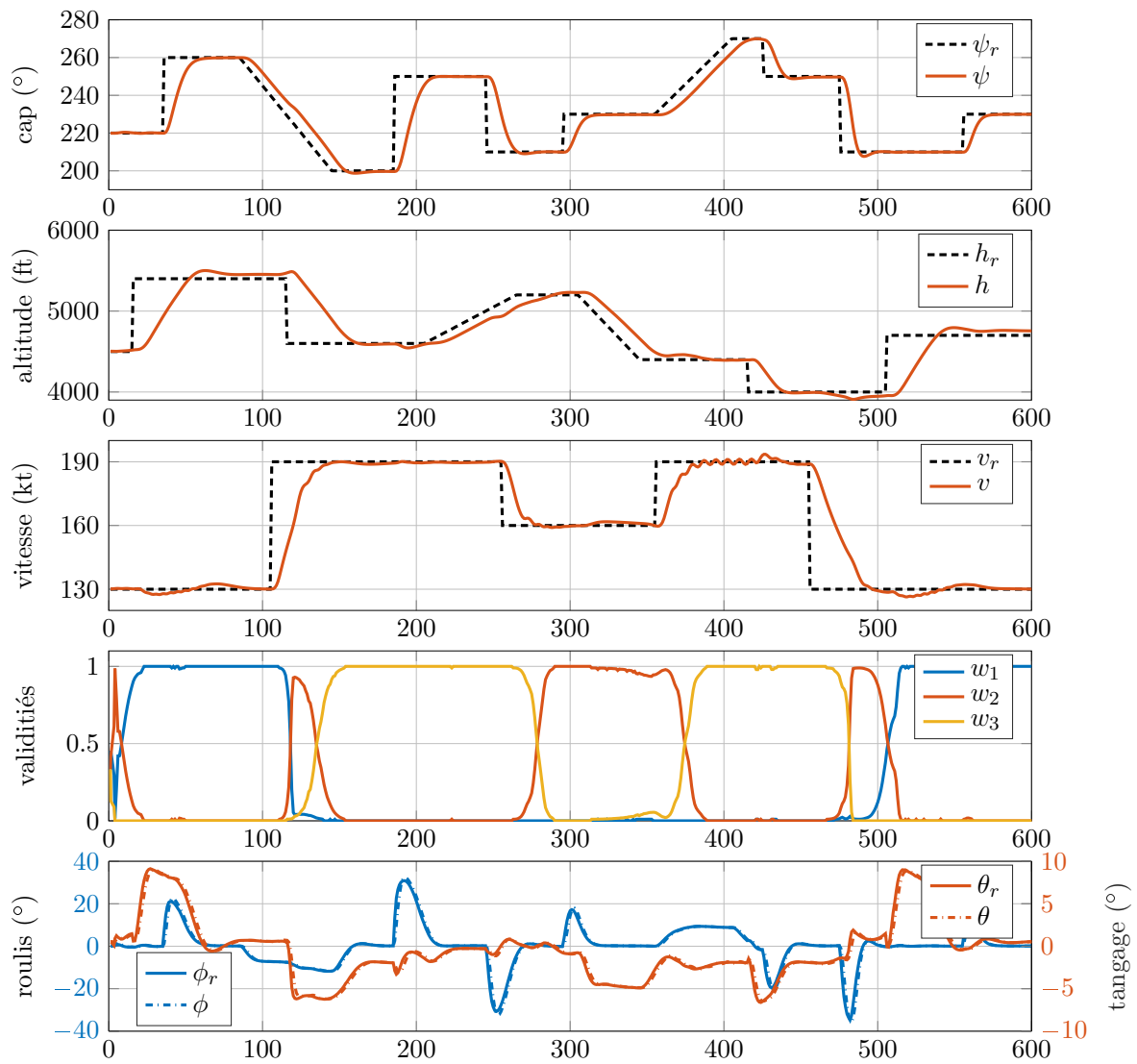


FIGURE 6.23 – Simulation de l'autopilote NMMAC de guidage pour un scénario test comportant des variations de vitesse

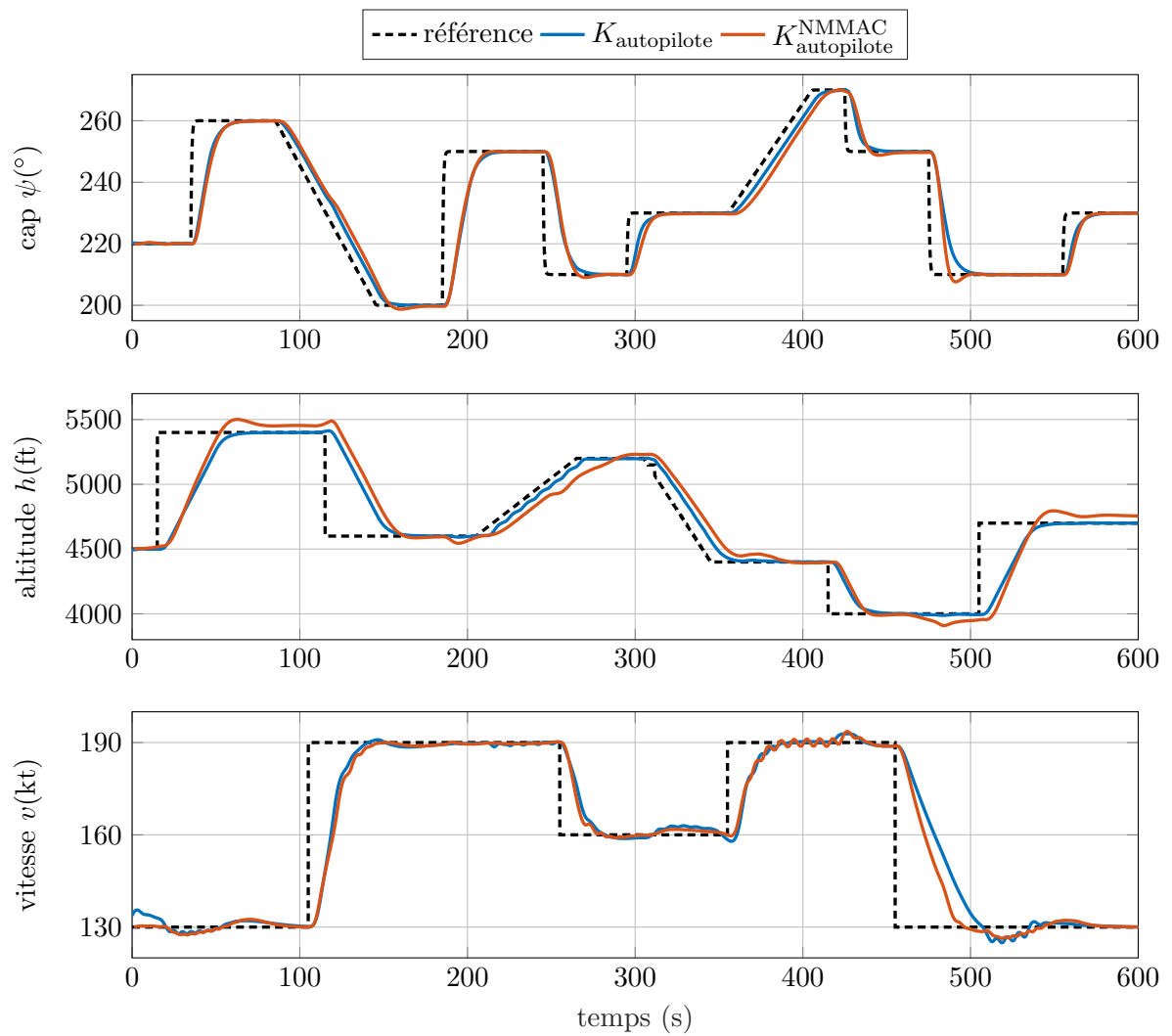


FIGURE 6.24 – Comparaison des simulations de l'autopilote de X-Plane et du NMMAC pour une scénario test comportant des variations de vitesse

6.4.2.2 Apprentissage multi-modèle d'un pilote

Les phases de générations des données reprennent les principes qui ont été détaillés dans la section 6.3.2.2. Deux nouvelles bases de données sont collectées lors de sessions de vol par temps orageux pendant lesquelles le pilote automatique interne contrôle la vitesse air de l'avion autour de $v = 220\text{kt}$ pour le second modèle puis $v = 160\text{kt}$ et avec le volet gauche arraché pour le troisième modèle. Les expérimentations sont réalisées pour un tour intégral du chemin de vol déjà exploité pour les précédents apprentissages et les données sont enregistrés à un temps d'échantillonnage de 0.05s.

L'apprentissage hors ligne des systèmes et des contrôleurs est par la suite réalisés pour les nouvelles bases de données. Les paramètres utilisés pour l'apprentissage sont l'algorithme Levenberg-Marquardt et un critère d'arrêt prématuré de 300 itérations. Le tableau 6.12 donnent les résultats des deux nouvelles modèles du système obtenues et rappelle ceux du premier modèle du système, c'est-à-dire pour les vecteurs entrées-sorties définies par l'équation (6.12).

TABLE 6.12 – Résultats d'apprentissage des systèmes du NMMAC pour le suivi de chemin de vol

	Structure						Résultats				
	topologie	n_{in}	n_{out}	σ_1	n_1	$n_{\mathcal{W}}$	iter.	durée	err. app.	err. val.	err. test
G_{160}^{NN}	SSNNARX	6	5	tanh	5	335	112	10s	7.0e-6	3.5e-6	2.4e-5
	SSNNOE	6	5	tanh	5	335	773	12m	1.0e-3	3.6e-3	6.2e-3
G_{220}^{NN}	SSNNARX	6	5	tanh	5	335	306	38s	4.5e-6	5.2e-6	6.9e-6
	SSNNOE	6	5	tanh	5	335	310	6m	9.4e-3	1.5e-2	1.6e-2
G_{volet}^{NN}	SSNNARX	4	5	tanh	5	299	311	49s	3.9e-6	4.9e-6	7.0e-6
	SSNNOE	4	5	tanh	5	299	786	18m	1.1e-3	2.3e-3	5.3e-3

L'apprentissage des contrôleurs (c'est-à-dire selon les entrées-sorties de l'équation (6.11)) est effectué en multi-objectifs de façon similaire à celle détaillée dans la section 6.3.2.3. Finalement, les front de Pareto des trois ensembles de contrôleurs sont présentés par la figure 6.25 et les résultats des deux derniers ensembles sont illustrés plus en détail par le tableau 6.13.

Les autopilotes sont par la suite co-simulés avec X-Plane en suivant le schéma d'implémentation présenté par la figure 6.18. Dans un premier temps, les expériences sont réalisées pour un vol régulé à une vitesse de 220kt, la figure 6.26 illustre les résultats de suivi du chemin de vol d'apprentissage pour une partie du second ensemble de contrôleurs K_{220}^* . Ces derniers exposent un comportement satisfaisant du suivi de chemin malgré une vitesse élevée. À titre comparatif, le contrôleur K_{160}^{*22} qui a été appris pour une vitesse de 160kt est également simulé pour ce scénario de vol. Dans un second temps, la figure 6.27 présente les résultats de simulation lorsque le volet gauche de l'appareil est arraché et que la vitesse est régulée autour de 160kt. Certains autopilotes du troisième ensemble de contrôleurs K_{volet}^* sont ainsi simulés, ces derniers affichent un comportement fortement similaire à celui du pilote imité. De même que précédemment, le contrôleur K_{160}^{*22} du premier ensemble est simulé pour ces paramètres de vol. Le comportement obtenu est en premier lieu satisfaisant puisque ce troisième scénario de vol est de nature assez proche du premier scénario (la consigne de vitesse est identique), néanmoins l'asymétrie de l'appareil du

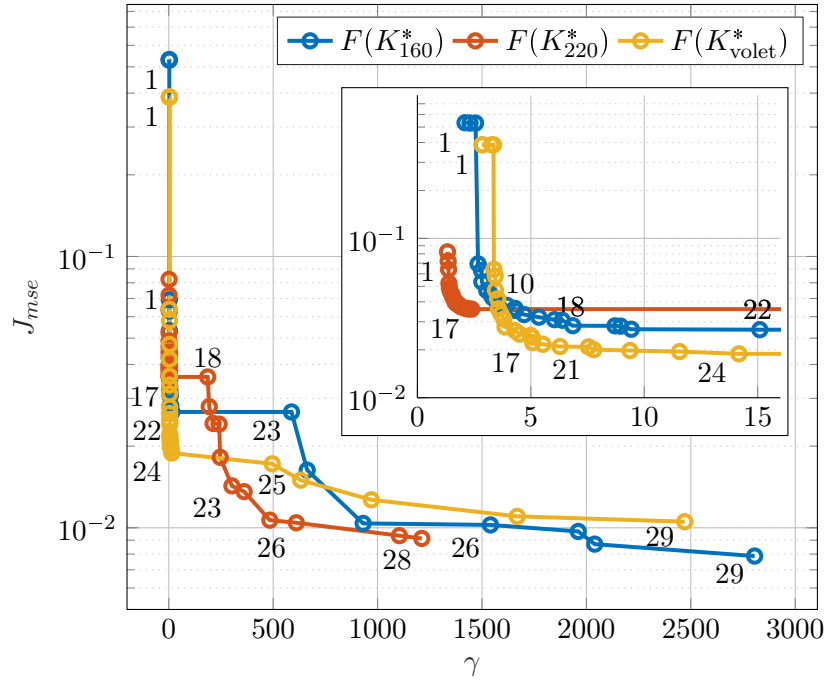


FIGURE 6.25 – Fronts de Pareto des autopilotes du NMMAC pour le suivi de chemin de vol

TABLE 6.13 – Résultats de l'apprentissage multi-objectifs des autopilotes de suivi de chemin de vol obtenus pour les deux modèles additionnels

	Structure						Optimisation		
	topologie	n_{in}	n_{out}	σ_1	n_1	$n_{\mathcal{Y}}$	itération	durée	
K_{220}^*	SSNNOE	4	5	tanh	10	752	3×3000	4j17h30m	
K_{voilet}^*	SSNNOE	4	5	tanh	10	752	3×3000	4j23h41m	
	Marge en entrée				Marge en sortie				Erreur
	$\gamma^{DM_u^\mu}$	DM_u^μ	GM_u	PM_u	$\gamma^{DM_y^\mu}$	DM_y^μ	GM_y	PM_y	J_{mse}
K_{220}^{*9}	1.32	0.75	6.9 dB	41.4°	1.66	0.60	5.4 dB	33.5°	4.0e-2
K_{220}^{*13}	1.38	0.72	6.6 dB	39.7°	1.90	0.52	4.7 dB	29.4°	3.7e-2
K_{220}^{*17}	1.55	0.64	5.8 dB	35.7°	2.39	0.42	3.7 dB	23.6°	3.5e-2
K_{220}^{*23}	6.31	0.16	1.4 dB	9.1°	302	0.00	0.0 dB	0.2°	1.4e-2
K_{voilet}^{*9}	3.57	0.28	2.5 dB	15.9°	3.56	0.28	2.5 dB	16.0°	3.6e-2
K_{voilet}^{*17}	4.23	0.23	2.1 dB	13.5°	5.08	0.20	1.7 dB	11.2°	2.2e-2
K_{voilet}^{*21}	3.95	0.25	2.2 dB	14.4°	7.77	0.13	1.1 dB	7.4°	3.0e-2
K_{voilet}^{*24}	6.51	0.15	1.3 dB	8.8°	14.1	0.07	0.6 dB	4.0°	1.8e-2

fait de l'absence du volet gauche provoque l'instabilité dès le premier virage à gauche.

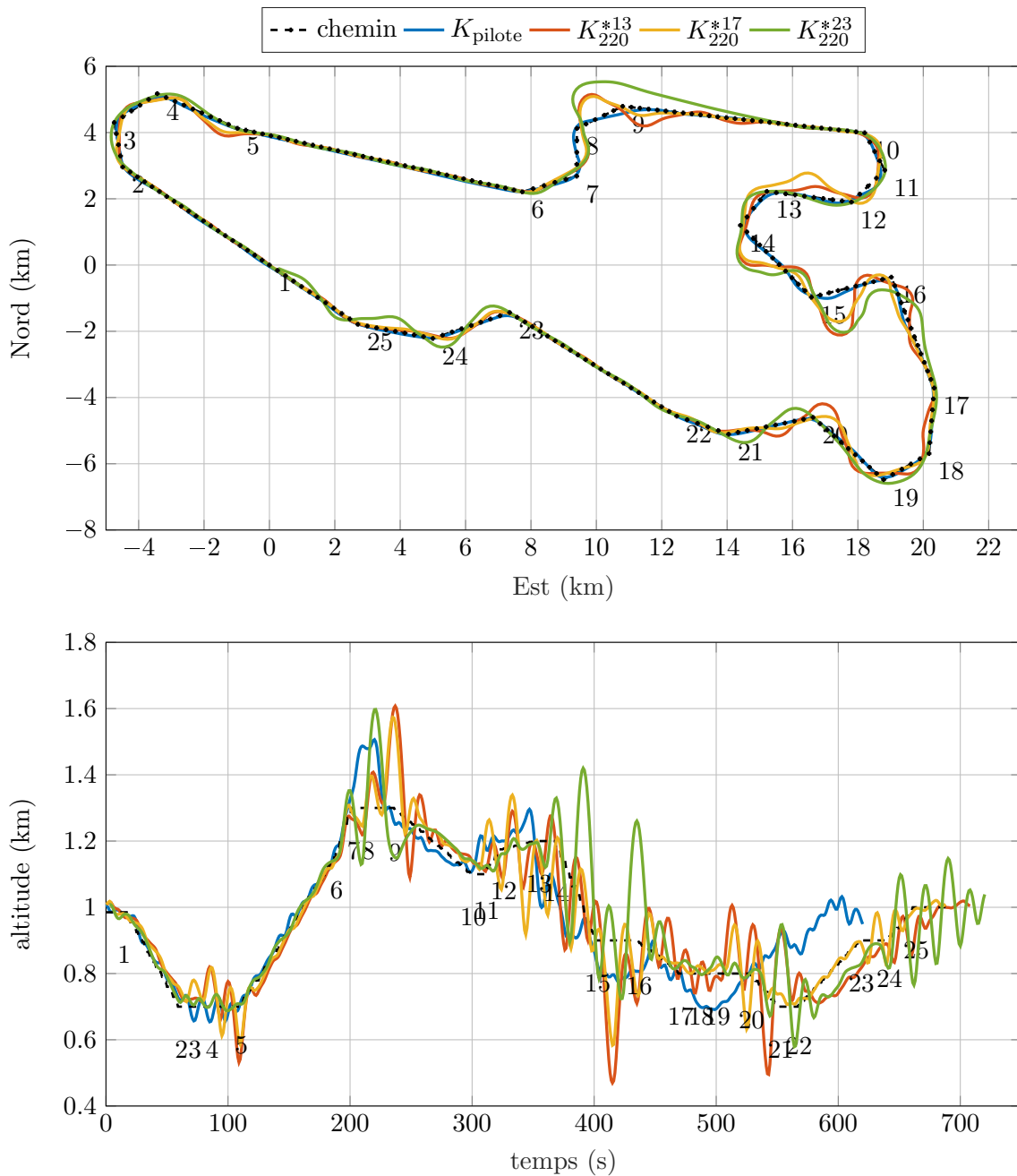


FIGURE 6.26 – Suivis du chemin de vol obtenus pour une vitesse de 220kt

6.4.2.3 Implémentation du NMMAC pour le suivi de chemin de vol

L'autopilote NMMAC de suivi de chemin de vol est finalement élaboré à partir des modèles et contrôleurs développés. La banque de modèles est composée des modèles neuronaux $M = \{G_{160}^{NN}, G_{220}^{NN}, G_{volet}^{NN}\}$ pour lesquelles des observateurs non-linéaires discret de type EKF sont conçus. Concernant les contrôleurs, nous sélectionnons un élément dans chacun

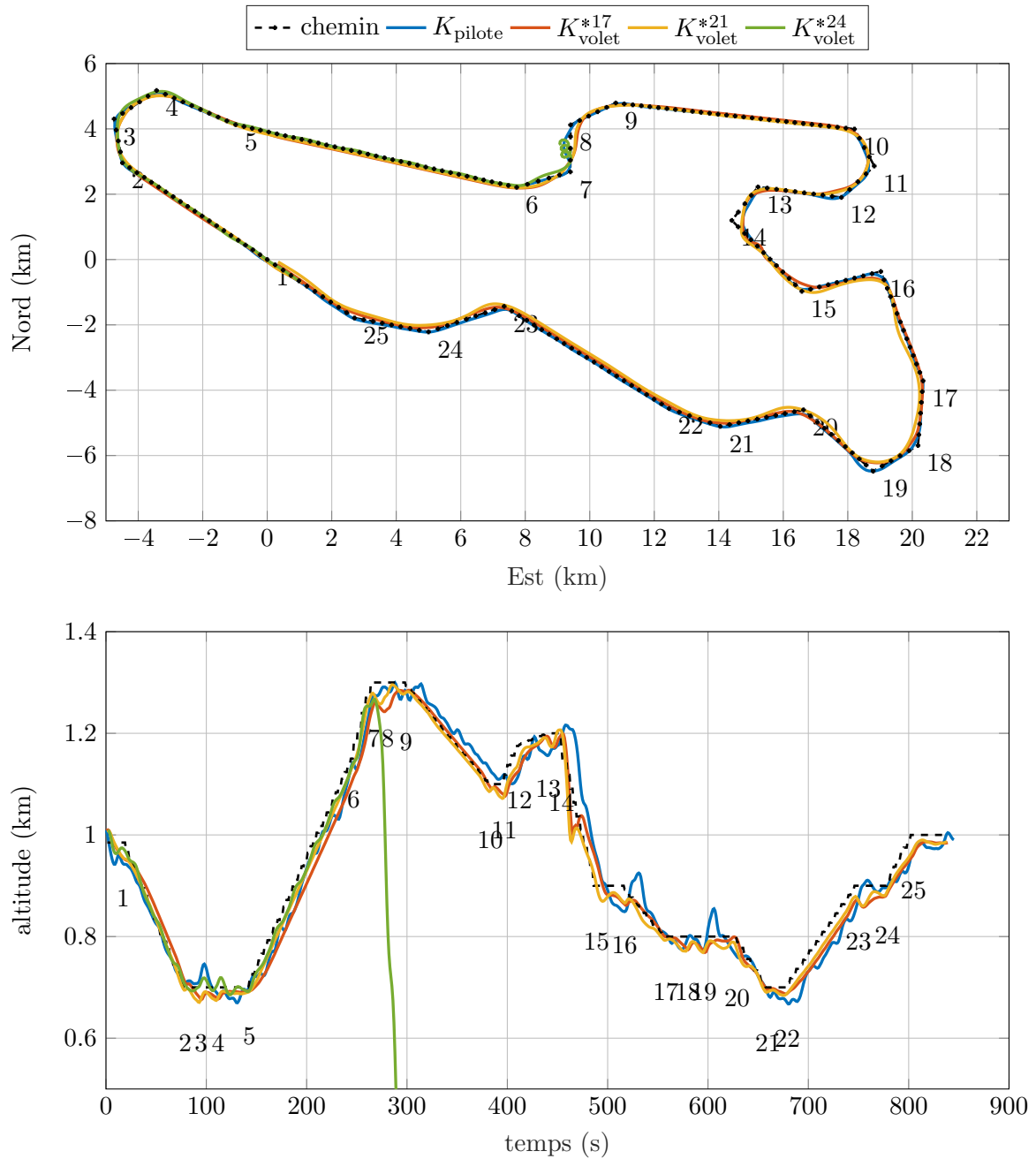


FIGURE 6.27 – Suivis du chemin de vol obtenus pour une vitesse de 160kt et lorsque le volet gauche est arraché

des différents ensembles, de façon à former la banque suivante $K = \{K_{160}^{*22}, K_{220}^{*13}, K_{\text{volet}}^{*17}\}$. Ces contrôleurs ont été choisis pour leur comportement individuel qui offre de bonnes propriétés de stabilité et de performance dans leur scénario d'asservissement respectif. L'étude de ce choix au regard des marges de stabilités constitue une perspective de recherche à explorer.

Une première phase de réglage du MMAE est réalisée à l'aide des différentes bases de données d'entraînements afin d'établir un réglage pertinent des EKF dont les comportements sont directement reliés à la convergence du PPE. Par cette méthode empirique, les réglages suivants sont finalement retenus : pour chacun des trois estimateurs, la covariance du bruit de processus $Q_i = \text{diag}([5\pi/180 \quad 2\pi/180 \quad 15 \quad \pi/180]_{\times 5}) \cdot 10^{-6}$, la covariance du bruit d'observation $R_i = \text{diag}([5\pi/180 \quad 2\pi/180 \quad 15 \quad \pi/180])$, la covariance de l'estimation initiale $P_i(k=0) = 10^2$, pour le PPE, le seuil $\delta = 0,01$ et les poids initiaux $w_i(k=0) = 1/3$.

En dernier lieu, l'autopilote NMMAC est co-simulé avec X-Plane lors de sessions de vols autonomes. La figure 6.28 présente les résultats de simulation obtenus sur le chemin de vol d'apprentissage, mais pour des variations du système, comme la modification de la vitesse de consigne entre 160kt et 220kt puis l'arrachement du volet gauche. Les résultats montrent un comportement de suivi de parcours satisfaisant, l'erreur latérale est maintenue à une valeur relativement faible ainsi que l'erreur d'altitude. Les validités exposent un comportement plus oscillant pour lesquelles nous pouvons énoncer les remarques suivantes. Premièrement lors des phases de vol proche de 160kt, le premier et troisième modèle sont fortement sollicités, l'alternance entre ces deux cas s'explique par la proximité des modèles qui ont tous les deux été établis pour cette consigne de vitesse. Comme nous l'avons évoqué pour la figure 6.27, leur différence est accentuée lorsque l'asymétrie de l'appareil impacte plus fortement le comportement aérodynamique de l'avion, c'est-à-dire lors de virage à gauche. Néanmoins, en dehors de ce cas, les deux contrôleurs assurent convenablement le suivi du chemin de vol. Deuxièmement, malgré la proximité du premier et troisième modèle, l'événement de panne d'arrachement du volet est instantanément détecté dès son apparition et la troisième validée est maintenue par la suite proche de l'unité. Enfin, le deuxième modèle intervient lorsque le comportement de l'avion n'est pas trop agité et que la vitesse est relativement proche des 220kt. Ce constat pourrait être étudié au vu des résultats d'apprentissages des différents modèles du système (tableau 6.12) puisque le deuxième modèle présente des moins bonnes performances d'identification en comparaison aux deux autres modélisations.

SECTION 6.5

Conclusions et perspectives

6.5.1 Conclusions

Ce chapitre a permis de mettre en application les résultats et méthodes développés dans nos travaux pour synthétiser des autopilotes neuronaux robustes par imitation des capacités de pilotage d'un pilote (humain ou artificiel). Pour ce faire, un outil d'apprentissage a été présenté sous forme d'une plateforme articulée autour de trois axes de recherche référés comme modules.

Tout d'abord, nous avons décrit la mise en place de l'environnement de co-simulation entre MATLAB et X-Plane. La présentation expose les éléments de développement clé pour aider le lecteur intéressé à bénéficier de ce simulateur de vol aussi performant que flexible.

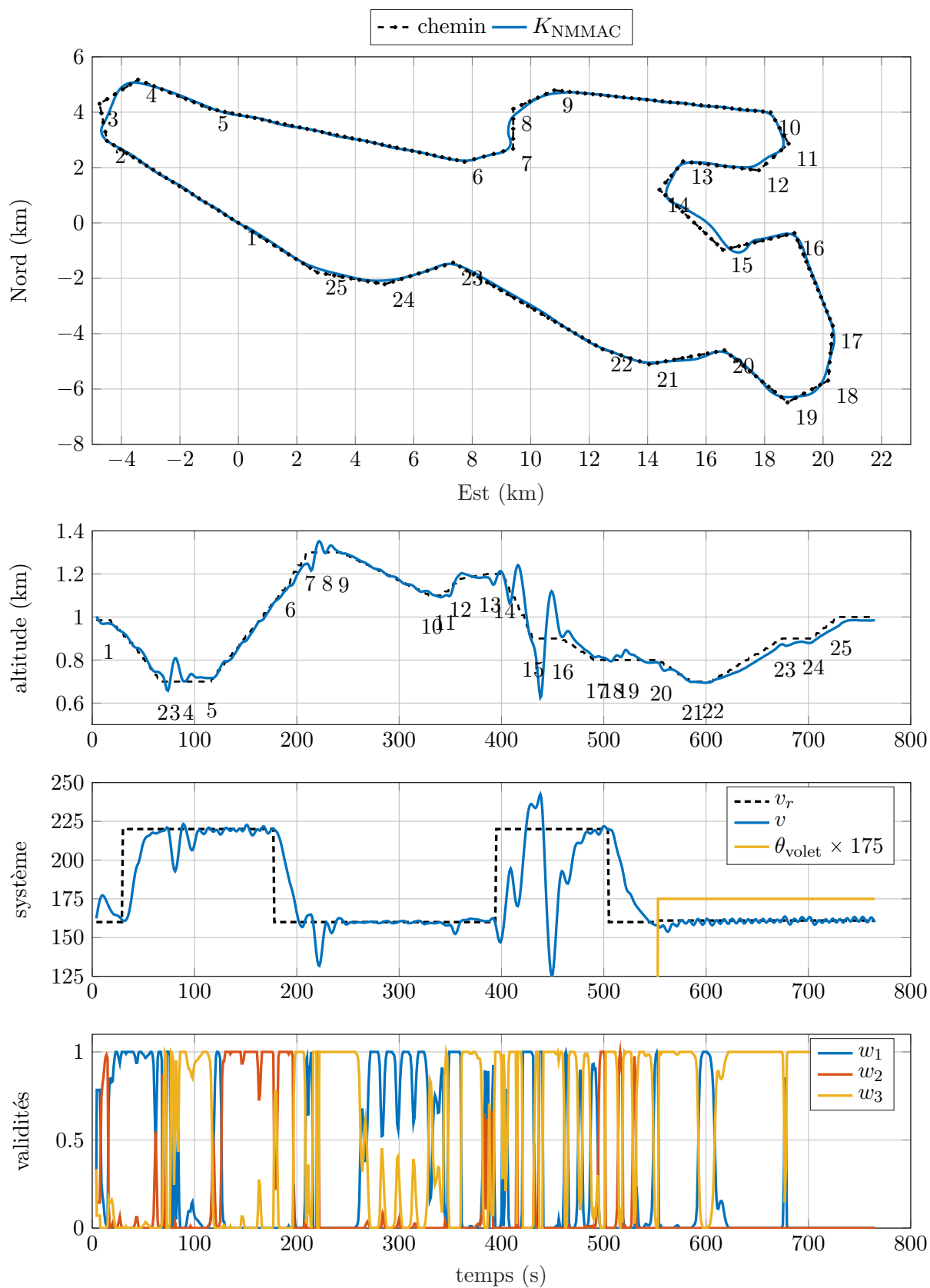


FIGURE 6.28 – Simulations de l'autopilote NMMAC de suivi de chemin de vol pour des variations du système

Notre étude a illustré par la suite les méthodes d'apprentissages de contrôleurs robustes du chapitre 4 afin d'élaborer un autopilote neuronal de guidage d'avion. Nous avons réalisé une première expérimentation dans l'objectif de développer un autopilote de guidage permettant de contrôler l'aéronef selon un cap et une altitude souhaités. Dans un premier temps, l'apprentissage se concentrait sur l'imitation de l'autopilote déjà implémenté dans X-Plane avant d'adresser dans un second temps l'imitation d'un pilote. La démarche entreprise pour ces expériences illustre les étapes d'élaboration de contrôleurs neuronaux robustes. Le contrôleur est d'abord obtenu par un apprentissage mono-objectif pour lequel les marges de stabilité de la loi de commande sont évaluées (ce qui implique également l'identification du système). Si besoin, la robustesse du contrôleur est renforcée par un apprentissage multi-objectifs qui aboutit alors à un ensemble de contrôleurs optimaux en terme de compromis entre la performance d'imitation et la robustesse. La seconde expérimentation qui a été menée a permis d'approfondir les capacités d'imitation en adressant le suivi d'un chemin de vol qui est défini par une succession de points de passage. Ce cas d'étude met alors en avant l'intérêt de l'apprentissage multi-objectifs puisque seule cette prise en compte des marges de stabilité pendant l'optimisation a permis d'imiter le pilote de façon stable. Les différents autopilotes du front de Pareto ont également été challengés pour le suivi d'un nouveau chemin de test et pour une variation paramétrique de l'avion.

Pour finir, nous avons exploité les perspectives de l'approche multi-modèle définie dans le chapitre 5 pour contribuer à l'implémentation de différents aspects de contrôle de l'autopilote neuronal. Nous avons développé cette technique dans un contexte de commande tolérante aux pannes, pour lequel une architecture NMMAC a été mise en œuvre. La banque de contrôleurs locaux est directement constituée des modèles SSNN et la logique de sélection utilise la banque de filtre de Kalman (EKF) qui ont été conçus à partir des modèles neuronaux des systèmes. Afin d'illustrer la méthode, nous avons dans un premier temps considéré la mesure de la vitesse air comme indisponible pour développer un contrôleur multi-modèle élaboré pour différentes valeurs de vitesse. Les contrôleurs ont alors été développés pour imiter l'autopilote disponible dans X-Plane dans l'objectif de suivre les signaux de référence de cap et d'altitude. Dans un second temps, nous avons consolidé l'autopilote de suivi de chemin de vol en imitant un pilote lors de nouveaux scénarios de vol pour une vitesse plus élevée et puis pour une panne où le volet gauche de l'aéronef est arraché. Par ces expérimentations, nous avons pu témoigner de la pertinence de l'approche bien que la méthode ne soit pas limitée à cette application.

6.5.2 Perspectives

Les travaux entrepris dans ce chapitre ont permis d'illustrer la méthodologie complète d'apprentissage de contrôleurs neuronaux robustes adaptée au contexte aéronautique d'autopilote de guidage. Néanmoins, la méthode n'est pas restreinte à cette application et d'autres étages de contrôle pourraient être envisagés dans ce cadre d'étude. De plus, d'autres systèmes bien différents pourraient être adressés comme d'autres véhicules autonomes, drones, robot, etc., là où un opérateur est capable d'exposer les propriétés de contrôle recherchées ou là où un contrôleur est déjà existant, mais doit être imité par exemple pour des questions d'embarquabilités.

Du point de vue de la méthode présentée, une voie de recherche a exploré pour compléter les résultats, serait d'étudier la détermination des différentes paires de modèles-contrôleurs au

regard de la stabilité et des performances globale du MMAC. Le front de Pareto obtenu pour chaque ensemble de contrôleurs offre un degré de liberté supplémentaire qui pourrait être exploité pour l'implémentation multi-modèle et les propriétés de robustesse que doivent remplir chaque contrôleur. De plus, une méthode complémentaire pourrait être développée afin d'examiner l'importance de chaque modèle de la banque et la contribution de chaque contrôleur au vu du système réel.

Conclusion générale

Conclusion

Cette thèse aborde le développement de systèmes de contrôle par imitation de comportements complexes. Différentes méthodes ont été présentées dans l'optique d'apprendre un contrôleur neuronal de façon efficace et robuste sur une base de données démontrant ces comportements. Les travaux doctoraux entrepris ont ainsi permis de rapprocher les domaines d'étude de l'apprentissage par réseaux de neurones et celui de la commande robuste.

Au cours de l'étude bibliographique, nous avons pu identifier des liens de cohésion de certaines structures de réseaux de neurones avec les modélisations linéaires à paramètres variants (LPV). Nous avons alors contribué à la mise en place de méthodes d'identification neuronale de systèmes dynamiques par des perceptrons multicouches récurrents. Les travaux se concentrent sur les structures neuronales qui peuvent être décrites par une représentation d'état, c'est-à-dire les SSNN. Ces structures présentent un comportement hybride résultant d'une contribution linéaire et non-linéaire. De plus, l'interprétation de l'architecture permet d'ajuster les apports de certains hyper-paramètres comme, l'ordre du modèle par les fenêtres de retards considérées et la dimension de la couche d'état, ou encore la complexité de l'équation d'évolution du système par les propriétés et le nombre de couches cachées non-linéaires. Une contribution importante des travaux est de formaliser le passage de ces réseaux de type SSNN dans le domaine d'étude des systèmes LPV. Le modèle équivalent présente alors un nombre de paramètres variants égal au nombre de neurones non-linéaires. Ce dernier peut également être représenté selon une transformation linéaire fractionnaire (LFT) c'est-à-dire selon une partie linéaire associée à une partie diagonale non-linéaire. Afin d'affiner cette modélisation, des méthodes ont été proposées pour notamment aboutir à une représentation non-perturbée grâce à un changement de coordonnées, puis en considérant des paramètres variants normalisés uniquement dans leur région effective. Finalement, par cette représentation LPV-LFT, nous avons développé différents outils d'analyse de stabilité et de performance de réseaux de neurones de type SSNN.

Sur ces bases théoriques, nous avons par la suite proposé des approches de développement de contrôleurs neuronaux robustes. En effet, lors de l'apprentissage d'un agent décisionnel intervenant dans une boucle de rétroaction, une problématique fondamentale porte sur la stabilité globale de la boucle et sa robustesse. Une contribution importante de la thèse a été de mettre en place une méthode afin de quantifier et d'ajuster les marges de robustesse d'un asservissement neuronal. Tout d'abord, nous avons défini le concept de marge de stabilité dans le contexte des SSNN puis nous avons développé des outils fondés sur l'analyse de systèmes LPV-LFT afin d'évaluer *a posteriori* ces marges au vu d'un contrôleur neuronal et du modèle neuronal du système. Par la suite, nous avons proposé une méthode d'apprentissage multi-objectifs permettant de concilier *a priori* les critères de robustesse avec les performances d'imitation. Cette stratégie possède notamment pour avantage de ne pas fournir un contrôleur unique, mais un ensemble de contrôleurs répartis le long du front de Pareto. Le choix du curseur entre la performance et la robustesse peut ainsi être appréhendé d'un point de vue « haut niveau » y compris pour les systèmes complexes. Une autre contribution des travaux est d'étendre les capacités du contrôleur neuronal par une méthode de contrôle adaptative multi-modèle (MMAC). Cette approche pose une structure ordonnée et flexible afin d'accumuler des connaissances pour consolider l'apprentissage, mais également pour fournir une stratégie de détection et de reconfiguration lors de pannes. Dans cette optique, nous avons présenté les différents

éléments composant le MMAC notamment les validités relatives à chaque paire modèle-contrôleur et leur estimation selon une logique bayésienne. Nous avons également développé une méthode d'analyse de stabilité du MMAC dans le cadre linéaire ainsi qu'une stratégie de réglage.

Les travaux entrepris ont finalement abouti au développement d'une méthodologie complète d'apprentissage de contrôleurs neuronaux robustes pour laquelle nous avons mis en application les principaux résultats pour le développement de pilote automatique d'avion. Tout d'abord, nous avons contribué à mettre en place un environnement de co-simulation entre MATLAB et le simulateur de vol X-Plane afin d'aborder des systèmes avec un haut degré de réalisme. Nous avons ensuite mis en pratique les différentes méthodes d'apprentissage par imitation de contrôleurs pour lesquelles les propriétés de robustesse sont établies puis renforcées si nécessaire. Pour cette étude, nous avons développé un autopilote de guidage assurant le suivi de cap et d'altitude par imitation d'un autopilote déjà implémenté puis d'un pilote. Nous avons également expérimenté l'apprentissage d'un pilote adressant le suivi d'un chemin de vol défini par une succession de points de passage. Par ces expériences, nous avons pu démontrer les contributions d'un point de vue pratique des méthodes qui ont été proposées et valider les concepts définis. Enfin, nous avons exploité les perspectives de l'approche multi-modèle pour étendre l'enveloppe de vol de l'autopilote neuronal. Nous avons ainsi présenté d'une part, la mise en œuvre d'un MMAC conçu à partir de modèles neuronaux pour développer, par imitation, un autopilote de guidage tolérant aux défaillances du capteur de vitesse. D'autre part, nous avons illustré la méthode dans le but de consolider et d'adapter l'autopilote de suivi de chemin de vol en imitant un pilote pour de nouvelles conditions de vol.

Bibliographie

- [Abbas et Werner, 2008a] ABBAS, H. et WERNER, H. (2008a). Lpv design of charge control for an si engine based on lft neural state-space models. *IFAC Proceedings Volumes*, 41(2):7427–7432. *citations pages 28 et 63*
- [Abbas et Werner, 2008b] ABBAS, H. et WERNER, H. (2008b). Polytopic quasi-lpv models based on neural state-space models and application to air charge control of a si engine. *IFAC Proceedings Volumes*, 41(2):6466–6471. *citations pages 28 et 31*
- [Apkarian et Adams, 2000] APKARIAN, P. et ADAMS, R. J. (2000). Advanced gain-scheduling techniques for uncertain systems. *In Advances in linear matrix inequality methods in control*, pages 209–228. SIAM. *citation page 50*
- [Apkarian et Gahinet, 1995] APKARIAN, P. et GAHINET, P. (1995). A convex characterization of gain-scheduled h_∞ controllers. *IEEE Transactions on Automatic Control*, 40(5):853–864. *citations pages 49, 52, 56 et 57*
- [Apkarian et al., 1995] APKARIAN, P., GAHINET, P., BECKER, G. et al. (1995). Self-scheduled h_∞ control of linear parameter-varying systems : a design example. *Automatica*, 31(9):1251–1261. *citation page 50*
- [Aschauer et al., 2015] ASCHAUER, G., SCHIRRER, A. et KOZEK, M. (2015). Co-simulation of matlab and flightgear for identification and control of aircraft. *IFAC-PapersOnLine*, 48(1):67–72. *citation page 150*
- [Athans et al., 1977] ATHANS, M., CASTANON, D., DUNN, K.-P., GREENE, C., LEE, W., SANDELL, N. et WILLSKY, A. (1977). The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method–part i : Equilibrium flight. *IEEE Transactions on Automatic Control*, 22(5):768–780. *citation page 125*
- [Balas et al., 2015] BALAS, G., HJARTARSON, A., PACKARD, A. et SEILER, P. (2015). Lpvttools : A toolbox for modeling, analysis, and synthesis of parameter varying control systems. *Software and user’s manual*. *citations pages 53 et 65*
- [Baram et Sandell, 1978] BARAM, Y. et SANDELL, N. (1978). An information theoretic approach to dynamical systems modeling and identification. *IEEE Transactions on Automatic Control*, 23(1):61–66. *citation page 128*
- [Bendtsen et Trangbæk, 2000] BENDTSEN, J. D. et TRANGBÆK, K. (2000). Transformation of neural state space models into lft models for robust control design. *citations pages 28, 62 et 67*
- [Bendtsen et Trangbæk, 2002] BENDTSEN, J. D. et TRANGBÆK, K. (2002). Robust quasi-lpv control based on neural state-space models. *IEEE Transactions on Neural Networks*, 13(2):355–368. *citations pages 28 et 67*

- [Biannic, 2010] BIANNIC, J. M. (2010). Contributions théoriques à la commande des systèmes aéronautiques et spatiaux. *Aux frontières du linéaire. Habilitation à diriger des recherches, Université Paul Sabatier de Toulouse, France, URL <http://jm.biannic.free.fr>*. *citation page 79*
- [Blight et al., 1994] BLIGHT, J. D., LANE DAILEY, R. et GANGSAAS, D. (1994). Practical control law design for aircraft using multivariable techniques. *International Journal of Control*, 59(1): 93–137. *citation page 99*
- [Boyd et al., 1989] BOYD, S., BALAKRISHNAN, V. et KABAMBA, P. (1989). A bisection method for computing the h_∞ norm of a transfer matrix and related problems. *Mathematics of Control, Signals and Systems*, 2(3):207–219. *citation page 46*
- [Boyd et al., 1994] BOYD, S., EL GHAOU, L., FERON, E. et BALAKRISHNAN, V. (1994). *Linear matrix inequalities in system and control theory*. SIAM. *citations pages 46, 47 et 135*
- [Briat, 2014] BRIAT, C. (2014). Linear parameter-varying and time-delay systems. *Analysis, observation, filtering & control*, 3:5–7. *citation page 49*
- [Broomhead et Lowe, 1988] BROOMHEAD, D. S. et LOWE, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Rapport technique, Royal Signals and Radar Establishment Malvern (United Kingdom). *citation page 20*
- [Chen et Francis, 1995] CHEN, T. et FRANCIS, B. A. (1995). Optimal sampled-data control systems. *citation page 48*
- [Cybenko, 1989] CYBENKO, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. *citation page 16*
- [Czajkowski et al., 2014] CZAJKOWSKI, A., PATAN, K. et SZYMAŃSKI, M. (2014). Application of the state space neural network to the fault tolerant control system of the plc-controlled laboratory stand. *Engineering Applications of Artificial Intelligence*, 30:168–178. *citation page 28*
- [Deb, 2011] DEB, K. (2011). Multi-objective optimisation using evolutionary algorithms : an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer. *citations pages 110 et 112*
- [Deb et al., 2002] DEB, K., PRATAP, A., AGARWAL, S. et MEYARIVAN, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197. *citation page 110*
- [Dennis Jr et Schnabel, 1996] DENNIS JR, J. E. et SCHNABEL, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM. *citation page 22*
- [Desoer et Vidyasagar, 1975] DESOER, C. A. et VIDYASAGAR, M. (1975). *Feedback Systems : Input-output Properties*, volume 55. SIAM. *citation page 54*
- [Doyle, 1982] DOYLE, J. (1982). Analysis of feedback systems with structured uncertainties. In *IEE Proceedings D-Control Theory and Applications*, volume 129, pages 242–250. IET. *citation page 55*
- [Doyle et al., 1988] DOYLE, J., GLOVER, K., KHARGONEKAR, P. et FRANCIS, B. (1988). State-space solutions to standard h_2 and h_∞ control problems. In *1988 American Control Conference*, pages 1691–1696. IEEE. *citation page 90*
- [Doyle, 1985] DOYLE, J. C. (1985). Structured uncertainty in control system design. In *1985 24th IEEE Conference on Decision and Control*, pages 260–265. IEEE. *citations pages 80 et 130*
- [Doyle et al., 1990] DOYLE, J. C., FRANCIS, B. A. et TANNENBAUM, A. R. (1990). *Feedback control theory*. Courier Corporation. *citation page 45*

- [Duc et Font, 1999] DUC, G. et FONT, S. (1999). *Commande Hinf et mu-analyse un outil pour la robustesse*. Hermes. *citations pages 45 et 130*
- [Ducard, 2009] DUCARD, G. J. J. (2009). *Fault-tolerant flight control and guidance systems : practical methods for small unmanned aerial vehicles*. Advances in industrial control. Springer. *citations pages 121 et 125*
- [El Ghaoui et Scorletti, 1996] EL GHAOUI, L. et SCORLETTI, G. (1996). Control of rational systems using linear-fractional representations and linear matrix inequalities. *Automatica*, 32(9):1273–1284. *citation page 52*
- [Fekri, 2005] FEKRI, S. (2005). *Robust adaptive MIMO control using multiple-model hypothesis testing and mixed- μ synthesis*. Thèse de doctorat, Ph. D. dissertation, Instituto Superior Tecnico, Lisbon, Portugal. *citation page 121*
- [Fekri et al., 2006] FEKRI, S., ATHANS, M. et PASCOAL, A. (2006). Issues, progress and new results in robust adaptive control. *International Journal of Adaptive Control and Signal Processing*, 20(10):519–579. *citations pages 121, 128 et 135*
- [Fekri et al., 2007] FEKRI, S., ATHANS, M. et PASCOAL, A. (2007). Robust multiple model adaptive control (rmmac) : A case study. *International Journal of Adaptive Control and Signal Processing*, 21(1):1–30. *citation page 128*
- [Fekri et al., 2008] FEKRI, S., GU, D., POSTLETHWAITE, I. et ATHANS, M. (2008). Robust adaptive fault-tolerant control of the f-14 aircraft under sensor failures. *IFAC Proceedings Volumes*, 41(2):10172–10177. *citation page 125*
- [Ferik et Adeniran, 2017] FERIK, S. E. et ADENIRAN, A. A. (2017). Modeling and Identification of Nonlinear Systems : A Review of the Multimodel Approach—Part 2. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 47(7):1160–1168. *citation page 123*
- [Feyel, 2013] FEYEL, P. (2013). *Loop-shaping robust control*. John Wiley & Sons. *citations pages 48 et 98*
- [Feyel, 2015] FEYEL, P. (2015). *Optimisation des correcteurs par les métaheuristiques. Application à la stabilisation inertielle de ligne de visée*. Thèse de doctorat, Supélec. *citations pages 3 et 4*
- [Feyel, 2017] FEYEL, P. (2017). *Optimisation de la commande robuste par les métaheuristiques*. ISTE Group. *citations pages 90, 109, 111 et 112*
- [Frasnedo, 2016] FRASNEDO, S. (2016). *Optimisation des lois de commande d’un imageur sur critère optronique. Application à un imageur à deux étages de stabilisation*. Thèse de doctorat, Université Paris-Saclay. *citation page 3*
- [Funahashi, 1989] FUNAHASHI, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192. *citations pages 16 et 65*
- [Gahinet et Apkarian, 1994] GAHINET, P. et APKARIAN, P. (1994). A linear matrix inequality approach to h_∞ control. *International Journal of Robust and Nonlinear Control*, 4(4):421–448. *citation page 46*
- [Gahinet et al., 1994] GAHINET, P., NEMIROVSKII, A., LAUB, A. J. et CHILALI, M. (1994). The lmi control toolbox. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 3, pages 2038–2041. IEEE. *citation page 47*
- [Garzon et Botelho, 1999] GARZON, M. et BOTELHO, F. (1999). Dynamical approximation by recurrent neural networks. *Neurocomputing*, 29(1-3):25–46. *citation page 18*

- [Gil *et al.*, 2013] GIL, P., HENRIQUES, J., CARDOSO, A. et DOURADO, A. (2013). On affine state-space neural networks for system identification : Global stability conditions and complexity management. *Control Engineering Practice*, 21(4):518–529. *citation page 29*
- [Gil *et al.*, 2006] GIL, P., HENRIQUES, J., DOURADO, A. et DUARTE-RAMOS, H. (2006). On state-space neural networks for systems identification : Stability and complexity. In *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–5. IEEE. *citation page 29*
- [Godfrey *et al.*, 2005] GODFREY, K., TAN, A., BARKER, H. et CHONG, B. (2005). A survey of readily accessible perturbation signals for system identification in the frequency domain. *Control Engineering Practice*, 13(11):1391–1402. *citation page 33*
- [Goodfellow *et al.*, 2016] GOODFELLOW, I., BENGIO, Y. et COURVILLE, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>. *citations pages 19 et 23*
- [Goodwin, 1977] GOODWIN, G. C. (1977). Dynamic system identification : experiment design and data analysis. *Mathematics in science and engineering*, 136. *citation page 33*
- [Grant et Boyd, 2014] GRANT, M. et BOYD, S. (2014). CVX : Matlab software for disciplined convex programming, version 2.1. <http://cvxr.com/cvx>. *citation page 47*
- [Grant et Boyd, 2008] GRANT, M. C. et BOYD, S. P. (2008). Graph implementations for nonsmooth convex programs. In *Recent advances in learning and control*, pages 95–110. Springer. *citation page 47*
- [Hagan *et al.*, 1996] HAGAN, M. T., DEMUTH, H. B. et BEALE, M. (1996). *Neural network design*. PWS Publishing Co. *citation page 22*
- [Hagan *et al.*, 2014] HAGAN, M. T., DEMUTH, H. B., BEALE, M. H. et DE JESS, O. (2014). *Neural network design*. Martin Hagan. *citations pages 19 et 39*
- [Hagan et Menhaj, 1994] HAGAN, M. T. et MENHAJ, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993. *citation page 22*
- [Hassani *et al.*, 2011] HASSANI, V., HESPANHA, J. P., ATHANS, M. et PASCOAL, A. M. (2011). Stability Analysis of Robust Multiple Model Adaptive Control. *IFAC Proceedings Volumes*, 44(1):350–355. *citations pages 128 et 135*
- [Haykin, 1999] HAYKIN, S. (1999). *Neural networks : a comprehensive foundation*. *citations pages 20 et 28*
- [Hiret, 1999] HIRET, A. (1999). *Pilotage robuste d'un missile sur un large domaine de vol synthèse et analyse dans le cadre H_∞ et LPV*. Thèse de doctorat. *citation page 55*
- [Hirwa, 2013] HIRWA, S. (2013). *Méthodes de commande avancées appliquées aux viseurs*. Thèse de doctorat, Supélec. *citation page 3*
- [Hochreiter et Schmidhuber, 1997] HOCHREITER, S. et SCHMIDHUBER, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. *citation page 20*
- [Hoffmann et Werner, 2014a] HOFFMANN, C. et WERNER, H. (2014a). Complexity of implementation and synthesis in linear parameter-varying control. *IFAC Proceedings Volumes*, 47(3):11749–11760. *citation page 50*
- [Hoffmann et Werner, 2014b] HOFFMANN, C. et WERNER, H. (2014b). A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations. *IEEE Transactions on Control Systems Technology*, 23(2):416–433. *citation page 50*

- [Hornik *et al.*, 1989] HORNİK, K., STINCHCOMBE, M. et WHITE, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. *citations pages 16 et 30*
- [Jin *et al.*, 1999] JIN, L., GUPTA, M. M. et NIKIFORUK, P. N. (1999). Dynamic recurrent neural networks for approximation of nonlinear systems. *IFAC Proceedings Volumes*, 32(2):5213–5218. *citation page 18*
- [Kersting et Buss, 2017] KERSTING, S. et BUSS, M. (2017). How to systematically distribute candidate models and robust controllers in multiple-model adaptive control : A coverage control approach. *IEEE Transactions on Automatic Control*, 63(4):1075–1089. *citation page 125*
- [Kretchmar, 2000] KRETCHMAR, R. M. (2000). *A synthesis of reinforcement learning and robust control theory*. Colorado State University. *citation page 63*
- [Krizhevsky *et al.*, 2012] KRIZHEVSKY, A., SUTSKEVER, I. et HINTON, G. E. (2012). Image-net classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105. *citation page 19*
- [Kuipers et Ioannou, 2010] KUIPERS, M. et IOANNOU, P. (2010). Multiple model adaptive control with mixing. *IEEE Transactions on Automatic Control*, 55(8):1822–1836. *citation page 128*
- [Lachhab *et al.*, 2008] LACHHAB, N., ABBAS, H. et WERNER, H. (2008). A neural-network based technique for modelling and lpv control of an arm-driven inverted pendulum. *In 2008 47th IEEE Conference on Decision and Control*, pages 3860–3865. IEEE. *citation page 29*
- [Lagarias *et al.*, 1998] LAGARIAS, J. C., REEDS, J. A., WRIGHT, M. H. et WRIGHT, P. E. (1998). Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147. *citation page 137*
- [LeCun *et al.*, 1989] LECUN, Y., BOSER, B., DENKER, J., HENDERSON, D., HOWARD, R., HUBBARD, W. et JACKEL, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2. *citation page 19*
- [LeCun *et al.*, 1991] LECUN, Y., KANTER, I. et SOLLA, S. A. (1991). Eigenvalues of covariance matrices : Application to neural-network learning. *Physical Review Letters*, 66(18):2396. *citation page 35*
- [Leys *et al.*, 2013] LEYS, C., LEY, C., KLEIN, O., BERNARD, P. et LICATA, L. (2013). Detecting outliers : Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4):764–766. *citation page 36*
- [Ljung, 1998] LJUNG, L. (1998). System identification. *In Signal analysis and prediction*, pages 163–173. Springer. *citations pages 16, 17, 33 et 34*
- [Lofberg, 2004] LOFBERG, J. (2004). Yalmip : A toolbox for modeling and optimization in matlab. *In 2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE. *citation page 47*
- [Luzar et Czajkowski, 2016] LUZAR, M. et CZAJKOWSKI, A. (2016). Lpv system modeling with ssn toolbox. *In 2016 American Control Conference (ACC)*, pages 3952–3957. IEEE. *citation page 28*
- [Mahalanobis, 1936] MAHALANOBIS, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India. *citation page 127*

- [Mahdianfar *et al.*, 2011] MAHDIANFAR, H., OZGOLI, S. et MOMENI, H. R. (2011). Robust multiple model adaptive control : Modified using ν -gap metric. *International Journal of Robust and Nonlinear Control*, 21(18):2027–2063. *citations pages 125 et 128*
- [Mayne *et al.*, 2006] MAYNE, D. Q., RAKOVIĆ, S., FINDEISEN, R. et ALLGÖWER, F. (2006). Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7):1217–1222. *citation page 127*
- [Megretski et Rantzer, 1997] MEGRETSKI, A. et RANTZER, A. (1997). System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830. *citation page 140*
- [Møller, 1993] MØLLER, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533. *citation page 22*
- [Morelli et Klein, 2016] MORELLI, E. A. et KLEIN, V. (2016). *Aircraft system identification : theory and practice*, volume 2. Sunflyte Enterprises Williamsburg, VA. *citations pages 33 et 35*
- [Narendra et Parthasarathy, 1990] NARENDRA, K. et PARTHASARATHY, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27. *citation page 17*
- [Nelles, 2020] NELLES, O. (2020). *Nonlinear System Identification : From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes*. Springer Nature. *citations pages 16, 30, 33 et 87*
- [Nelson *et al.*, 2006] NELSON, D. R., BARBER, D. B., MCLAIN, T. W. et BEARD, R. W. (2006). Vector field path following for small unmanned air vehicles. In *2006 American Control Conference*, pages 7–pp. IEEE. *citation page 166*
- [Nørgård *et al.*, 2000] NØRGÅRD, P. M., RAVN, O., POULSEN, N. K. et HANSEN, L. K. (2000). Neural networks for modelling and control of dynamic systems-a practitioner’s handbook. *citations pages 17, 21, 31, 33, 37, 38 et 87*
- [Ogunmolu *et al.*, 2016] OGUNMOLU, O., GU, X., JIANG, S. et GANS, N. (2016). Nonlinear systems identification using deep dynamic neural networks. *arXiv preprint arXiv :1610.01439*. *citation page 20*
- [Packard, 1994] PACKARD, A. (1994). Gain scheduling via linear fractional transformations. *Systems & control letters*, 22(2):79–92. *citation page 52*
- [Packard et Doyle, 1993] PACKARD, A. et DOYLE, J. (1993). The complex structured singular value. *Automatica*, 29(1):71–109. *citation page 80*
- [Pascanu *et al.*, 2013] PASCANU, R., MIKOLOV, T. et BENGIO, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR. *citation page 23*
- [Patan, 2008] PATAN, K. (2008). *Artificial neural networks for the modelling and fault diagnosis of technical processes*. Springer. *citations pages 16, 28 et 30*
- [Pinguet *et al.*, 2020] PINGUET, J., FEYEL, P. et SANDOU, G. (2020). A design and analysis method for estimator-based multiple model adaptive control. In *2020 8th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 74–81. IEEE. *citation page 7*
- [Pinguet *et al.*, 2021] PINGUET, J., FEYEL, P. et SANDOU, G. (2021). A neural autopilot training platform based on a Matlab and X-Plane co-simulation. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1200–1209. IEEE. *citation page 7*

- [Pouilly-Cathelain, 2020] POUILLY-CATHELAIN, M. (2020). *Synthèse de correcteurs s'adaptant à des critères multiples de haut niveau par la commande prédictive et les réseaux de neurones*. Thèse de doctorat, université Paris-Saclay. *citations pages 3 et 39*
- [Pulido *et al.*, 2019] PULIDO, B., ZAMARREÑO, J. M., MERINO, A. et BREGON, A. (2019). State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems. *Engineering Applications of Artificial Intelligence*, 79:67–86. *citation page 28*
- [Richard *et al.*, 2019] RICHARD, A., MAHÉ, A., PRADALIER, C., ROZENSTEIN, O. et GEIST, M. (2019). A comprehensive benchmark of neural networks for system identification. *citation page 20*
- [Riedmiller et Braun, 1993] RIEDMILLER, M. et BRAUN, H. (1993). A direct adaptive method for faster backpropagation learning : The rprop algorithm. *In IEEE international conference on neural networks*, pages 586–591. IEEE. *citation page 22*
- [Rivals et Personnaz, 1996] RIVAL, I. et PERSONNAZ, L. (1996). Black-box modeling with state-space neural networks. *In Neural Adaptive Control Technology*, pages 237–264. World Scientific. *citation page 27*
- [Rosa, 2011] ROSA, P. (2011). *Multiple-model adaptive control of uncertain LPV systems*. Thèse de doctorat, Citeseer. *citation page 128*
- [Rosa *et al.*, 2007] ROSA, P., ATHANS, M., FEKRI, S. et SILVESTRE, C. (2007). Further evaluation of the rmmac method with time-varying parameters. *In 2007 Mediterranean Conference on Control Automation*, pages 1–6. *citation page 128*
- [Rosa *et al.*, 2009] ROSA, P., SHAMMA, J. S., SILVESTRE, C. et ATHANS, M. (2009). Integration of the stability overlay (so) with the robust multiple-model adaptive control (rmmac). *In 2009 17th Mediterranean Conference on Control and Automation*, pages 223–228. IEEE. *citation page 128*
- [Rumelhart *et al.*, 1986a] RUMELHART, D. E., HINTON, G. E. et WILLIAMS, R. J. (1986a). Learning internal representations by error propagation. Rapport technique, California Univ San Diego La Jolla Inst for Cognitive Science. *citation page 22*
- [Rumelhart *et al.*, 1986b] RUMELHART, D. E., HINTON, G. E. et WILLIAMS, R. J. (1986b). Learning representations by back-propagating errors. *nature*, 323(6088):533–536. *citation page 22*
- [Safonov, 1993] SAFONOV, M. (1993). A multiplier method for computing real multivariable stability margin. *In 12th IFAC world congress, Sydney*, volume 2, pages 275–278. *citation page 55*
- [Schenker et Agarwal, 1997] SCHENKER, B. et AGARWAL, M. (1997). Dynamic modeling using neural networks. *International journal of systems science*, 28(12):1285–1298. *citations pages 27 et 30*
- [Scherer et Weiland, 2015] SCHERER, C. et WEILAND, S. (2015). *Linear matrix inequalities in control*. *citation page 47*
- [Seiler *et al.*, 2020] SEILER, P., PACKARD, A. et GAHINET, P. (2020). An introduction to disk margins [lecture notes]. *IEEE Control Systems Magazine*, 40(5):78–95. *citations pages 99 et 100*
- [Shamma, 1988] SHAMMA, J. S. (1988). *Analysis and design of gain scheduled control systems*. Thèse de doctorat, Massachusetts Institute of Technology. *citation page 49*
- [Shamma, 2012] SHAMMA, J. S. (2012). An overview of lpv systems. *Control of linear parameter varying systems with applications*, pages 3–26. *citation page 49*

- [Shamma et Athans, 1990] SHAMMA, J. S. et ATHANS, M. (1990). Analysis of gain scheduled control for nonlinear plants. *IEEE Transactions on Automatic Control*, 35(8):898–907. *citation page 49*
- [Sims et al., 1969] SIMS, F., LAINIOTIS, D. et MAGILL, D. (1969). Recursive algorithm for the calculation of the adaptive Kalman filter weighting coefficients. *IEEE Transactions on Automatic Control*, 14(2):215–218. *citations pages 121 et 123*
- [Sjöberg et al., 1995] SJÖBERG, J., ZHANG, Q., LJUNG, L., BENVENISTE, A., DEYLON, B., GLORENNEC, P.-Y., HJALMARSSON, H. et JUDITSKY, A. (1995). *Nonlinear black-box modeling in system identification : a unified overview*. Linköping University. *citation page 16*
- [Söderström et Stoica, 1989] SÖDERSTRÖM, T. et STOICA, P. (1989). *System identification*. Prentice-Hall International. *citations pages 15, 33 et 87*
- [Sun et al., 2019] SUN, S., CAO, Z., ZHU, H. et ZHAO, J. (2019). A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681. *citation page 22*
- [Suykens et al., 1997] SUYKENS, J. A., DE MOOR, B. et VANDEWALLE, J. (1997). Robust nl_q neural control theory. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 4, pages 2396–2401. IEEE. *citations pages 28 et 67*
- [Suykens et al., 1995a] SUYKENS, J. A., DE MOOR, B. L. et VANDEWALLE, J. (1995a). Nonlinear system identification using neural state space models, applicable to robust control design. *International Journal of Control*, 62(1):129–152. *citations pages 27, 28 et 62*
- [Suykens et al., 1995b] SUYKENS, J. A., VANDEWALLE, J. P. et de MOOR, B. L. (1995b). *Artificial neural networks for modelling and control of non-linear systems*. Springer Science & Business Media. *citations pages 28 et 65*
- [Tao et al., 2016] TAO, X., LI, N. et LI, S. (2016). Multiple model predictive control for large envelope flight of hypersonic vehicle systems. *Information Sciences*, 328:115–126. *citation page 125*
- [Van Veldhuizen, 1999] VAN VELDHUIZEN, D. A. (1999). *Multiobjective evolutionary algorithms : classifications, analyses, and new innovations*. Air Force Institute of Technology. *citation page 106*
- [Veenman et al., 2016] VEENMAN, J., SCHERER, C. W. et KÖROĞLU, H. (2016). Robust stability and performance analysis based on integral quadratic constraints. *European Journal of Control*, 31:1–32. *citation page 140*
- [Vinnicombe, 2000] VINNICOMBE, G. (2000). *Uncertainty And Feedback, H Loop-shaping And The V-gap Metric*. World Scientific. *citation page 125*
- [Vogl et al., 1988] VOGL, T. P., MANGIS, J., RIGLER, A., ZINK, W. et ALKON, D. (1988). Accelerating the convergence of the back-propagation method. *Biological cybernetics*, 59(4):257–263. *citation page 22*
- [Werbos, 1990] WERBOS, P. J. (1990). Backpropagation through time : what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560. *citation page 23*
- [Williams et Zipser, 1989] WILLIAMS, R. J. et ZIPSER, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280. *citation page 23*
- [Wu, 1995] WU, F. (1995). *Control of linear parameter varying systems*. Thèse de doctorat, University of California, Berkeley. *citation page 50*

- [Wu et Dong, 2006] WU, F. et DONG, K. (2006). Gain-scheduling control of lft systems using parameter-dependent lyapunov functions. *Automatica*, 42(1):39–50. *citation page 52*
- [Zamarreno *et al.*, 2000] ZAMARRENO, J., VEGA, P., GARCIA, L. et FRANCISCO, M. (2000). State-space neural network for modelling, prediction and control. *Control Engineering Practice*, 8(9):1063–1075. *citation page 28*
- [Zamarreño et Vega, 1998] ZAMARREÑO, J. M. et VEGA, P. (1998). State space neural network. properties and application. *Neural networks*, 11(6):1099–1112. *citations pages 27, 28 et 30*
- [Zames, 1966] ZAMES, G. (1966). On the input-output stability of time-varying nonlinear feedback systems part one : Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE transactions on automatic control*, 11(2):228–238. *citations pages 54 et 55*
- [Zhou *et al.*, 1996] ZHOU, K., DOYLE, J. C., GLOVER, K. *et al.* (1996). *Robust and optimal control*, volume 40. Prentice hall New Jersey. *citations pages 46, 52, 54 et 80*

Titre : Contribution à la synthèse de contrôleurs neuronaux robustes par imitation

Mots clés : Commande robuste, Réseaux de neurones, Apprentissage par imitation, Contrôle adaptatif multi-modèle

Résumé : Cette thèse s'intéresse à l'élaboration de systèmes de contrôle par imitation de comportements ou de décisions répondant à des exigences complexes. L'objectif est de réaliser l'apprentissage d'un contrôleur neuronal de façon efficace et robuste sur une base de données regroupant ces comportements. L'approche retenue unifie les outils de la commande robuste avec ceux de la modélisation par réseaux de neurones. Des méthodes d'identification de systèmes dynamiques sont tout d'abord développées selon des structures neuronales en cohésion avec les représentations de systèmes linéaires à paramètres variants. L'accès à ce domaine d'étude ouvre alors la voie à l'analyse de stabilité et de performance de ces types de modèles neuronaux. Les travaux proposent par la suite d'exploiter ces propriétés afin de répondre aux enjeux de robustesse inhérents à l'apprentissage de lois de commande. La méthode d'identification de contrôleurs robustes

qui est proposée, repose sur l'évaluation des marges de stabilité de l'asservissement neuronal. Il est alors permis de consolider la robustesse des contrôleurs à travers une stratégie d'apprentissage avec optimisation de la stabilité par une formulation multi-objectifs. En complément, le déploiement des contrôleurs est effectué selon une méthode de contrôle adaptative multi-modèle. La démarche est finalement appliquée aux pilotes automatiques d'avion via une co-simulation avec un simulateur de vol caractérisé par sa grande fiabilité de modélisation. Les problématiques de contrôle abordées sont, dans un premier temps de guider l'appareil selon un cap et une altitude donnés, tandis qu'une seconde expérimentation se concentre sur le suivi d'un chemin de vol constitué d'une série de points de passage. Les autopilotes neuronaux sont développées par l'imitation d'un autopilote existant puis par l'imitation d'un pilote.

Title : Contribution to the synthesis of robust neural controllers by imitation

Keywords : Robust control, Neural networks, Imitation learning, Multiple model adaptive control

Abstract : This thesis focuses on developing control systems by imitating behaviors or decisions meeting complex requirements. The objective is to perform the learning of a neural controller efficiently and robustly on a database containing these behaviors. The chosen approach unifies robust control tools with those of neural network modeling. Methods for identifying dynamic systems are first developed according to neural structures in cohesion with the representations of linear systems with varying parameters. Access to this field of study opens the way to stability and performance analysis of these neural models. The work then proposes to exploit these properties to address the robustness issues inherent to the learning of control laws. The proposed method of identifying ro-

bust controllers is based on evaluating the stability margins of the neural feedback loop. It is then possible to consolidate the robustness of the controllers through a learning strategy with stability optimization by a multi-objective formulation. In addition, the deployment of the controllers is performed using a multi-model adaptive control method. The approach is finally applied to aircraft autopilots via a co-simulation with a flight simulator characterized by its high modeling reliability. The control issues addressed are, in the first step, to guide the aircraft according to a given heading and altitude, while a second experiment focuses on following a flight path consisting of a series of waypoints. The neural autopilots are developed by imitating an existing autopilot and then by imitating a pilot.