UNIVERSITÉ DE STRASBOURG

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

---

Mesures et observations d'états

Mémoire de soutenance à
l'Habilitation à Diriger des Recherches

SPÉCIALITÉ : AUTOMATIQUE, MATHÉMATIQUES APPLIQUÉES

Benoît Schwaller
Laboratoire LGECO, EA 3938 INSA Strasbourg

15 mai 2017

Composition du jury :

M. Basset	Pr. à l'Université de Haute Alsace, S61	membre invité
M. Darouach	Pr. à l'Université de Lorraine, S61	président du jury
M. Farza	Pr. à l'Université de Caen, ENSICAEN, S61	rapporteur externe
T.M. Guerra	Pr. à l'Université de Valenciennes et Hainaut-Cambresis, S61	rapporteur externe
C. Prud'homme	Pr. à l'Université de Strasbourg, S26	rapporteur interne
J. Ragot	Pr. à l'Université de Lorraine, S61	membre invité
Y. Rémond	Pr. à l'Université de Strasbourg, S60	garant HDR

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES STRASBOURG

iut LOUIS PASTEUR

Sommaire

Liste des figures

Remerciements

À tous mes proches et amis fidèles ...

Chapitre 1

CV synthétique

Académie :	**Strasbourg**
Section C.N.U. :	**61** ième
Nom patronymique :	**SCHWALLER**
Prénoms :	**Benoît René**
Date et lieu de naissance :	**Né le 16 01 65 à Strasbourg**
Nationalité :	**Française**
Situation de famille :	**Marié, deux enfants**
Adresse personnelle :	**49, route de Rountzenheim 67620 Soufflenheim**
Numéro de téléphone :	**03 88 86 71 83.**
Fonctions :	**Maître de conférences**
Établissement actuel :	**IUT de Schiltigheim, département Génie Industriel et Maintenance, Allée d'Athènes, 67300 Schiltigheim**
Laboratoire de recherche :	**Institut de Mécanique des Fluides et des Solides de Strasbourg.**
Titres universitaires français :	**Doctorat de l'Université Louis Pasteur : « Plasticité du réflexe vestibulo oculomoteur et commande adaptative, modèles et procédures ». Soutenance le 30 Novembre 1993 à l'Université Louis Pasteur de Strasbourg. Mention : très honorable avec félicitations du jury. Directeur de thèse : Jean Louis Eichhorn.** **Prix de thèse Adrérus 1994.**
Titre universitaire étranger :	**« Diplom Ingenieur der Fachhochschule Offenburg », Fachbereich Regelungstechnik (option automatique), obtention en Juillet 1990, mention Bien.**

Mots clés Identification paramétriques, traitement du signal, logique floue, systèmes experts, intégration de systèmes distribués, temps réel, électronique embarquée, acquisition de données haut débit, mesures en continu distribuées.

Chapitre 2

Activités d'enseignement

2.1 En tant qu'ATER

Pendant l'année 1993-1994, j'ai occupé un demi poste d'attaché temporaire d'enseignement et de recherche à l'UFR des Sciences Physiques de l'Université Louis Pasteur. J'y ai dispensé des cours de métrologie, traitement du signal, langage C.

2.2 En tant que Maître de Conférences

2.2.1 1994-1998, IUT de Colmar, département GTR

Dans le cadre de la création du département de Génie des Télécommunications et des Réseaux (GTR) de l'IUT de Colmar, j'ai été engagé par l'Université de Haute Alsace dès la rentrée universitaire 1994. Ma participation à l'ensemble des réflexions et travaux visant à la définition des programmes pédagogiques a fait partie de la mise en place des objectifs communs. Deux années de démarrage, suivies de deux années de stabilisation ont été nécessaires pour amener ce nouveau département à maturité.
L'équipe enseignante s'est étoffée en 1995 de nouveaux arrivants et en 1996, le département a remis à plat ses enseignements en vue d'une harmonisation nationale des programmes GTR. Toutes ces évolutions m'ont amené à enseigner un nombre assez important de matières différentes en cours, TD et TP, d'où un travail de rédaction de cours renouvelé chaque année, ainsi qu'un travail de logistique pour créer les séries de TP.

La liste des différentes matières où je suis intervenu est la suivante : électronique numérique, utilisation d'UNIX, téléphonie, communications hertziennes, administration UNIX.

La mise en place des salles de TP pour lesquelles j'ai été sollicité s'est décomposée comme suit :

- entre 1994-1995 : gestion de l'appel d'offre public pour l'équipement de la salle d'informatique du département, comportant un serveur UNIX, un serveur PC NT, et 14 postes PC étudiants ; mise en place du serveur informatique TR7000 (*Telmat*) du département, et configuration des 14 postes étudiants en émulateurs de terminaux X ; achats et mise en place de la salle de TP d'électronique numérique et d'informatique industrielle ;
- entre 1995-1996 : mise en place d'une salle de 7 serveurs PC UNIX SVR4 pour les TP d'administration UNIX ; mise en place d'un partenariat de formation avec la société *Telmat* pour les problèmes touchant à l'administration de machines UNIX.

2.2.2 1999-2015, IUT de Schiltigheim, département GIM

Fin 1998, j'ai obtenu une mutation à l'IUT Louis Pasteur de Schiltigheim au département Génie Industriel et Maintenance (GIM).

Entre 1999 et 2015, j'ai enseigné l'électronique numérique, l'informatique industrielle, la métrologie, l'automatique, le traitement du signal, l'électrotechnique et l'électronique de puissance.

Responsabilités et mise en place de salles de TP

— renouvellement de la salle de TP d'informatique industrielle avec la mise en réseau de 9 postes clients Linux et d'un serveur de stockage, achat de nouvelles platines de câblage, pistage et câblage des coffrets contenant les cartes à microprocesseurs étudiées ;

- définition et mise en place de la salle de métrologie commune au département mesures physiques et GIM ainsi que les TP sous LabView ;
- définition et installation avec une collègue d'une salle dédiée à l'enseignement de la sécurité électrique, ainsi qu'à la formation à l'habilitation électrique ;
- mise en place des TP d'automatisme de première année GIM, ainsi que des TP de simulation d'asservissements en deuxième année GIM ;
- Responsabilité de la salle de TP de machines tournantes et d'électronique de puissance.

2.2.3 Enseignements en temps que vacataire, responsabilité de filière

Parallèlement aux cours à l'IUT, il m'a été proposé de donner des cours et TP en DEA à l'UFR des sciences Physiques de l'ULP :

- **1996-1998 :** 15h etd de cours et 13h etd de TP de C au DEA de physique subatomique à l'Institut de Recherches Subatomiques (IRES) de Strasbourg ;
- **2000-2001 :** 15h etd de cours et l'écriture d'un polycopié de métrologie et de calcul d'erreur au DEA de Mécanique et Ingénierie ;
- **2001-2002 :** 27h etd de cours de métrologie ;
- **2002-2003 :** réorganisation de ce module sous forme de stages encadrés en laboratoires.
- **2003-2004 :** 12h etd de cours de métrologie ;
- **2004-2005 :** 9h etd de cours de métrologie.

2.2.4 Direction d'étude

Entre 2001 et 2003, j'ai été responsable intérimaire de la direction des études de la filière Ingénierie des Sciences Pour le Vivant (ISPV) à l'École Nationale Supérieure de Physique de Strasbourg (ENSPS), suite au départ à la retraite du Pr. *Charles Kopp*. La gestion des emplois de temps, des intervenants vacataires, des projets de fin d'études ainsi que la mise en place d'un enseignement d'Intelligence Artificielle ont fait partie des missions qui m'ont été confiées.

Chapitre 3

Activités de recherche 1990-2008

3.1 1990-1994, activités à l'Institut de Mécanique des Fluides de Strasbourg

3.1.1 Sujet de thèse

De fin 1990 à l'été 1994, j'ai effectué mon travail de thèse à l'Institut de Mécanique des Fluides de Strasbourg (IMFS, URA CNRS 854, département SPI), au sein de l'équipe « Systèmes Biomécaniques » (SBM).
Le sujet de cette thèse consistait en la modélisation, la simulation et l'exploration fonctionnelle du système de l'équilibre humain. Ce travail faisait suite à de nombreuses études mécaniques des capteurs comme des effecteurs du Réflexe Vestibulo-Oculomoteur (RVeO), système asservissant les mouvements oculaires aux mouvements de la tête.
Les buts poursuivis étaient articulés autour de trois axes de recherches complémentaires :

- la modélisation du système oculomoteur complet, en tenant compte des 3 paires de muscles concernés par le mouvement de chaque œil, ainsi que de leurs grandeurs d'influences ;
- la modélisation automatique du RVeO en incluant les six capteurs d'accélérations angulaires de la tête, et toutes les composantes adaptatives de ce réflexe ;
- la conception d'un atelier robotisé d'exploration fonctionnelle servant d'aide au diagnostic instrumenté et de validation clinique des modèles mathématiques du RVeO.

3.1.2 Bilan scientifique

L'ensemble de tests instrumentés mis en œuvre sur site à l'hôpital de Strasbourg en service ORL et d'ophtalmologie était une intégration de systèmes temps réel, de traitement du signal, d'électronique, d'intelligence artificielle, et de base de données. Cet ensemble a été utilisé en routine clinique, pendant environ une décennie (Schwaller 1993, Conraux et al. 1993, Schwaller et al. 1993).
J'ai ensuite pris mes fonctions de maître de conférences à l'IUT de Colmar, au département « Génie des Télécom et Réseaux » en septembre 1994.

3.2 1994-1998, activités à l'Université de Haute Alsace (UHA)

3.2.1 Présentation du cadre de recherche

Fin 1995, le Groupe de Recherches en Physique des Hautes énergies (GRPHE) a démarré son activité au sein de l'Université de Haute Alsace en tant que laboratoire de recherche en physique des particules. Il s'agissait d'un laboratoire mulhousien pluridisciplinaire comportant des physiciens des hautes énergies, un physicien du solide et deux maîtres de conférences automaticiens. Ce groupe a travaillé en lien avec l'Institut de Recherches Subatomiques de Strasbourg (IRES) à la mise au point et au test instrumenté de capteurs de position de particules à hautes énergies pour l'expérience Compact Muons Solenoid (CMS) au CERN.

3.2.2 Implication du GRPHE dans le projet

Dans ce cadre, j'ai été amené à concevoir les systèmes d'acquisition de données (DAQ) nécessaires aux tests des prototypes de capteurs dans les faisceaux du CERN. *J'ai assumé la responsabilité de l'intégration de système du trigger et de la DAQ des « milestones tracking CMS » pour l'année 1997. Pour réaliser cette mission, il a

fallu prévoir et réaliser l'infrastructure matérielle et logicielle sur la ligne de faisceau X5 du « West array » pour gérer 40000 canaux d'acquisition. L'accueil des groupes de physiciens et la coordination des prises de données a fait partie intégrante de la mise en place des expériences.
Les physiciens ont leurs habitudes de travail en matières de logiciels de calcul et traitements post-stockage. Ceci a nécessité le développement d'outils spécifiques de formatage des données permettant d'utiliser toutes les infrastructures réseau et calculateurs du CERN. Les logiciels temps réel et les cartes trigger spécifiques pour ces expériences ont été développés sans avoir pu être entièrement testés en laboratoire. En effet, certaines conditions de travail en faisceau ne sont pas simulables. Ces chaînes instrumentales complexes sont une synthèse où le génie logiciel, le génie électronique et micro électronique (ASICs analogiques et hybrides) ainsi que la physique sont associés. Les tests complets et les validations se font sur site, simultanément avec les prises de données réelles. En effet, tous les morceaux d'une chaîne ne sont assemblés que pendant le temps d'une expérience de quelques jours ou semaines.

3.2.3 Bilan des campagnes de mesures 1997

Les tests de 1997 du tracking CMS ont permis aux physiciens d'atteindre tous les objectifs des programmes expérimentaux servant de base à la rédaction des « Technical Design Report » de 1998. Le GRPHE a contribué à la préparation des sites expérimentaux durant le printemps.
Les prises de données ont été menées au cours de 3 périodes de 2 à 3 semaines chacune durant l'été et l'automne. Cet ensemble de machines distribuées est restée en service jusqu'en décembre 1999 date de fin de la recherche−développement sur les capteurs de l'expérience CMS. Cet appareillage a permis la prise de plusieurs To de données (Drouhin et al. 2000, Ackermann et al. 1999, Benhammou et al. 2000, Fessler et al. 1999, Schwaller, Drouhin, Pallarès, Fontaine, Benhammou, Charles, Huss & Hoffmann 1998, Drouhin et al. 1997), (Schwaller, Hoffmann, Drouhin, Pallarès, Fontaine, Benhammou, Charles & Huss 1998, Schwaller et al. 1997*a*,*b*).

3.2.4 Encadrement de doctorants

Début 1996, le groupe de recherches en physique des hautes énergies (GRPHE) a accueilli deux doctorants. L'encadrement d'un des deux (F. Drouhin) m'a été confié à 100%, avec pour sujet de thèse le développement de systèmes massifs d'acquisition de données.
Cette thématique a officialisé une collaboration de recherche entre la société TELMAT et le GRPHE existant de fait depuis un an. Début 1997, Mme A. Pallarès a complété l'équipe « acquisition de données » du GRPHE pendant environ un an après avoir soutenu sa thèse, en attendant l'obtention d'une bourse postdoctorale.
J'ai eu le plaisir de diriger cette petite équipe pendant toute la durée du projet « milestones tracking CMS ». Ceci a été une expérience humaine et scientifique passionnante dans le cadre d'un projet international européen du CERN en lien avec une équipe italienne. Le travail a été mené à son terme, et tous les objectifs scientifiques fixés ont été atteints.

F. Drouhin a soutenu sa thèse de doctorat intitulée : « conception d'un système d'acquisition de données dans le cadre des tests Milestone Barrel 1 et Milestone Forward 1 » le 4 décembre 1998 à Colmar, et a obtenu la mention très honorable avec les félicitations du jury. le jury était composé comme suit :

Président :	G. Binder	Pr. à l'UHA, S61
Directeur :	D. Huss	Pr. à l'UHA, S29
Rapporteur interne :	J.M. Meyer	Pr. à l'UHA, S61
Rapporteur externe :	J.P. Meinadier	Pr. au CNAM Paris, S61
Rapporteur externe :	M. Naïmi	Pr. à l'UTBM, S27
Membre du jury	P. Siegrist	Coordinateur Technique du *Tracker* de CMS au CERN
Membre invité	P.G. Verdini	Chercheur de l'Istituto Nationale di Fisica Nucleare, Italie

F. Drouhin est devenu maître de conférence à l'IUT de Colmar en mai 1999 au département GTR, section 61.

3.3 1998, mutation à l'IUT Louis Pasteur de Schiltigheim, quadriennal 2000-2004

Courant 1998, j'ai obtenu ma mutation à l'IUT Louis Pasteur de Schiltigheim, et pour mon affectation recherche j'ai réintégré l'Institut de Mécanique des Fluides et des Solides de Strasbourg (IMFS, UMR 7507 ULP/CNRS, département SPI). Dans le cadre du plan quadriennal 2000-2004 de l'IMFS j'ai pris la codirection de l'équipe Systèmes Biomécaniques et Cognitifs (SBMC), et mon programme scientifique s'est décomposé en deux parties distinctes :

— reprendre mes études théoriques sur l'identification de systèmes et finaliser une théorie originale utilisant les principes de stabilité asymptotique de *Liapounoff*, applicable pour des sommes algébriques de fonctions, et des équations différentielles linéaires ;
— utiliser les outils du traitement du signal appliqué au développement d'un instrument de mesure des champs de vitesses et des débits massiques dans des fluides chargés.

Cette dualité apparente s'explique du fait même de l'existence d'une équipe EEA au sein d'un Institut de Mécanique. Le laboratoire émargeait au département SPI du CNRS (aujourd'hui Ingénierie), et la vocation du petit groupe EEA dont j'ai eu la responsabilité jusque fin 2004 a été de mettre en application les outils de l'automatique et du Traitement du Signal au service de projets concrets intéressant la mécanique. Ceci implique que les algorithmes, bibliothèques de calculs, et les expériences tirées d'une thématique sont mises à profit, dès que c'est possible pour venir en appui à divers projets.

3.3.1 Identification paramétrique *in vivo* du « RVeO 3D »

En 1999, j'ai commencé à diriger la thèse de monsieur S. Faucheux, boursier MENRT. Cette direction s'est passée en collaboration avec le Pr. A. Buizza de l'Institut d'Informatique et de Systèmes de l'Université de Pavie.
Spécialiste internationalement reconnu en modélisation du vivant, il a constitué un partenaire privilégié d'étude des systèmes neuromécaniques. Cette thèse concernait le prolongement de mes propres travaux de thèse ; nous nous proposions d'effectuer des mesures du RVe0 pour les 3 degrés de liberté. Le corollaire était d'en effectuer une identification paramétrique aussi complète que possible, afin de caractériser les Canaux Semi-Circulaires (CSC) chez l'Homme.
L'identification paramétrique d'une paire de CSC a été menée en deux étapes.

— Dans un premier temps, utiliser une approche fréquentielle statistique connue (algorithme de Marquardt-Levenberg sur fonction de transfert) pour ajuster un modèle de réponse entre la vitesse du fauteuil de stimulation du patient, et la vitesse de rotation de l'oeil.
— Dans un deuxième temps, nous avons testé une méthode d'identification paramétrique en ligne originale développée par mes soins. Les résultats obtenus par cette deuxième approche ont pu être confrontés à l'algorithme de Marquardt-Levenberg. Ceci a permis de tester sur un vrai cas concret très bruité des algorithmes qui n'avaient été analysés que sur le plan théorique. Cette méthode d'identification dans le domaine temporel, applicable à des systèmes uniquement linéaires (à l'époque), se base sur la théorie de la stabilité selon *Liapounoff* qui en assure la convergence.
  Les résultats ainsi obtenus sont, dans leur majorité, comparables à ceux obtenus par le calcul de la fonction de transfert, ce qui a validé cette nouvelle approche.

Bilan scientifique

Les constantes de temps identifiées par les deux méthodes ne sont pas de l'ordre de grandeur de la constante de temps mécanique des CSC (résultat escompté au départ). Ces dernières se rapprochent plutôt de la constante de temps apparente du RVe0 horizontal.
Les retombées de cette constatation sont multiples. Ceci remet en question l'hypothèse classique d'une absence de stockage de vitesse sur les CSC verticaux, et pose un certain nombre de questions quant aux modèles communément admis décrivant le système de l'équilibre.
Malheureusement les tests des CSC verticaux ne sont pas aptes à nous donner une mesure directe des caractéristiques mécaniques des vestibules, et ce résultat ferme la porte à une caractérisation directement exploitable pour des praticiens ORL. L'hypothèse de l'absence de stockage de vitesse sur les CSC verticaux est issue d'études basées sur le RViO, la tête restant droite. Nos mesures semblent suggérer que les stockages de vitesse présents sur les trois CSC sont tous modulés par l'accélération de la gravité (C.Kopp et al. 1998, Schwaller 2001, Willinger et al. 2002, Faucheux 2003, Faucheux et al. 2003*b*,*a*, 2006, 2007).
La thèse a été soutenue avec succès le 6 février 2003, à Strasbourg. Il s'agissait d'une cotutelle franco-italienne, le jury était donc composé à parité de français et d'italiens :

Codirecteur :	B. Schwaller	Mc. à l'ULP S61
Codirecteur :	A. Buizza	Pr. à l'Université de Pavie, Italie
Rapporteur interne :	F. Braun	Pr. à l'ULP S63
Rapporteur externe :	J. Ragot	Pr. à l'INPL Nancy S61
Rapporteur externe :	M.G. Signorini	Pr. à l'Institut Polytechnique de Milan, Italie
Membre examinateur :	G. Magenes	Pr. à l'Université de Pavie, Italie
Membre invité :	C.M. Kopp	Pr. à l'ULP S60
Membre invité :	A. Gentine	Pr. à l'ULP Faculté de Médecine

3.3.2 Recherches sur la fluxmétrie Doppler

Parallèlement, j'ai mené un projet de développement métrologique à l'IMFS sur la fluxmétrie Doppler de fluides chargés et opaques.
Ce développement a servi de moteur au montage de la cellule de métrologie et d'électronique de l'UMR 7507. Cette structure s'est véritablement mise en place en 2001, et ce grâce à la venue d'un ingénieur de recherche en électronique. Ceci a permis d'envisager des développements de nouveaux capteurs ultrasonores, alliant la technologie des transducteurs au traitement du signal embarqué en continu.
L'usage de la vélocimétrie ultrasonore pulsée est souvent guidé par le besoin de mesurer localement la vitesse et l'intensité turbulente d'un fluide. Ces mesures sont faites conjointement sur un profil de l'écoulement, permettant ainsi d'en déduire entre autre le débit, la viscosité, le cisaillement.
L'accès à ces deux grandeurs locales se fait au travers des deux premiers moments de la densité spectrale de puissance du signal Doppler. Or le calcul de ces moments sans filtrage préalable engendre un biais considérable, en particulier lorsque le rapport signal sur bruit est faible. Ce type de situation est rencontré dans un grand nombre d'applications. C'est le cas des fluides faiblement chargés (très peu de rétro diffusion) que l'on trouve par exemple dans l'assainissement, c'est aussi celui des fluides fortement chargés (absorption importante) rencontrés, entre autres, dans les boues de forages, l'agro-alimentaire ou encore les cosmétiques.
La méthode proposée consiste, pour chaque volume de mesure, à estimer le spectre par FFT, puis à identifier les différentes composantes d'un modèle spectral théorique, en utilisant la méthode d'ajustement non linéaire de Levenberg-Marquardt. Cette approche a permis la séparation des bruits du signal utile. Le spectre nouvellement obtenu permet d'obtenir la vitesse moyenne dans le volume de mesure par calcul du moment d'ordre 1 du spectre. L'intensité turbulente peut être calculée à partir du moment centré d'ordre 2, moyennant la connaissance de la géométrie du volume de mesure. Les tests réels ont été effectués sur des suspensions chargées en bentonites et laponites, dans un écoulement de 2cm de diamètre, sur une boucle hydraulique spécifique, conçue à l'IMFS pour ce type de tests.

Bilan scientifique

Ces recherches ont permis le dépôt d'un brevet français concernant la vélocimétrie ultrasons. Une extension européenne concernant la vélocimétrie à « grande profondeur » est venue compléter les résultats de ces développements métrologiques. Cette dernière étude concernait la mesure de vitesses pour des tranches d'écoulement pour lesquelles ces vitesses sont trop élevées par rapport au temps que mettent les ondes sonores pour effectuer l'aller-retour dans le milieu. Ceci a permis d'ouvrir le champ d'application de notre instrument à des écoulements en charge ou en surface libre. La profondeur d'exploration peut atteindre ainsi le mètre, pour des vitesses moyennes de l'ordre de 2 m/s (Fischer et al. 2001, Fischer, P.Schmitt & Schwaller 2004, Fischer, Hurter, Schmitt, Schwaller & Scrivener 2004, Fischer, Schmitt & Schwaller 2004).

3.4 Quadriennal 2004-2008

En 2002, l'IMFS s'est restructurée en 2 secteurs d'activités, « Fluides et Environnement » d'une part, et « Matériaux et Santé » d'autre part. Chaque secteur était lui-même constitué d'Unités Thématiques de Recherches (UTR), couvrant chacune un certain nombre de projets contractuels.
J'ai été proposé pour la codirection de l'une d'entre elles, intitulée « Écoulement de Fluides Chargés et Assainissement » (EFCA). Pour ce quadriennal, l'UTR EFCA s'est focalisée sur trois objectif :

- l'étude du phénomène de transport solide ;
- le développement d'un fluxmètre Doppler industriel ;
- le développement de systèmes experts de traitement de données en continu.

J'ai été amené à intervenir dans chacun de ces trois axes.

3.4.1 Action Spécifique Conjointe : étude du phénomène de transport solide

En août 2002, le département SPI du CNRS a validé une action spécifique conjointe (ASC) CNRS-Ministère de la recherche dont j'ai eu la responsabilité scientifique, intitulée « Phénomènes de transport solide et de mélange : liaisons entre échelles microscopiques et macroscopiques » (Bulletin du SPI, n^o 34, août 2002).
Il s'agissait d'un programme ayant l'ambition de développer les technologies ultrasons mises en œuvres au laboratoire pour accéder aux mesures de granulométrie et de concentration en particules de fluides chargés opaques. On trouve fréquemment ce type de fluides dans les industries pétrolières, cosmétiques et en génie civil.
La notification de cette ASC a eu lieu le 17/11/2003, son arrivée au bureau de la Recherche de l'ULP a lieu le 7/1/2004. L'ouverture de la ligne budgétaire correspondante s'est faite le 20/2/2004.

Le but de ces recherches a consisté à modéliser l'écoulement des suspensions argileuses notamment lors des phases de démarrage, les modélisations disponibles dans la littérature ne permettant d'aucune manière de simuler les comportements observés. La difficulté d'une telle approche réside dans le fait qu'on ne dispose en laboratoire que d'une longueur d'étude (et donc d'une durée d'étude) très limitée : en effet, comme il s'agit de caractériser des phénomènes de thixotropie qui dépendent par nature de toute l'histoire antérieure des contraintes, seule la zone de fluide ayant subit la même histoire peut être considérée comme utile. En somme pour un essai donné, seul les quelques mètres de fluide initialement au repos dans la partie plane d'une boucle hydraulique fournissent des données interprétables.

Bilan d'équipe

Lors de l'étude du comportement des suspensions à seuil d'écoulement, il a été possible de distinguer deux types de thixotropie : une thixotropie vraie, celle classiquement décrite dans la littérature, c'est à dire le phénomène au cours duquel toute zone de l'écoulement ayant subi une même histoire de contrainte se trouvera dans le même état structurel et une thixotropie que nous avons qualifiée d'apparente où des zones de l'écoulement ayant subie la même histoire de contrainte se trouveront dans des états structurels différents. Les paramètres de l'écoulement sont alors principalement gouvernés par la position et le déplacement de l'interface entre zones gélifiées et zones fluidifiées. Nous avons ainsi mis en évidence plusieurs seuils de contraintes correspondant aux différentes zones de l'écoulement et avons pu modéliser assez fidèlement l'évolution de ses paramètres. Pour ces études, des suspensions de Bentonite et de Laponite ont principalement servi de modèle physique aux études expérimentales microscopiques et macroscopiques.
La modélisation du comportement rhéologique de tous les mélanges étudiés (solutions polymères, suspensions de bentonite et mélanges bentonite-polymères et/ou tensioactif) a constitué une autre partie de ce programme de recherche. Dans ce cadre, un modèle à cinq paramètres, basé sur le modèle de *Heschel-Bulkley*, a été proposé. L'idée était de considérer que les indices d'écoulement n et la consistance K ne sont pas constants mais varient en fonction de la vitesse de cisaillement. Les résultats obtenus avec ce modèle ont été comparés à d'autres modèles rhéologiques. Il en ressort que la meilleure corrélation des résultats expérimentaux a été obtenue avec le modèle proposé sur une très large gamme de vitesses de cisaillement. En outre, ce modèle, qui se caractérise par le nombre réduit de ses paramètres (cinq) par rapport à d'autres modèles (jusqu'à huit), permet de comparer les comportements rhéologiques de différents fluides de différentes natures en se basant sur les indices physiques qu'il fournit.

Bilan personnel

Mon apport a consisté à la mise en œuvre d'algorithmes d'identification développés en support de la thèse de *S. Faucheux*. Cette approche permet de résoudre l'identification paramétrique en ligne de systèmes physiques modélisés par des équations différentielles linéaires ou non linéaires à une entrée une sortie, et d'ordre 3 au maximum.
Les identifications s'appuient sur une structure observatrice originale (Schwaller et al. 2009), dont les conditions de stabilité ont été analysées et démontrées.

3.4.2 Le projet MES-FLUX

Ce projet, présenté au Ministère de l'industrie, a été labellisé le 30 janvier 2003 par les experts du comité RITEAU (Réseau de recherche et d'Innovation Technologique EAU et technologies de l'environnement). J'en ai assuré la codirection scientifique.
Le but visé par ce projet technologique a consisté au développement d'un dispositif ultrasonore en réseau d'eau pour la mesure conjointe du débit et de la concentration des matières en suspension dans des réseaux d'assainissement.

Diverses réglementations pour la protection de l'environnement prévoient la mesure des flux de pollution contenus dans les rejets industriels et urbains. Ces mesures impliquent la connaissance des débits et des concentrations de pollution, dont les matières en suspension (MES) sont un paramètre fondamental. Les moyens de mesure restent limités pour ces grandeurs :

- la débitmétrie en conduite à surface libre (collecteur d'assainissement par exemple) reste relativement imprécise ;
- la connaissance des concentrations de MES repose sur l'échantillonnage, nécessairement peu représentatif dans le temps et dans la veine fluide, et très coûteux.

L'objectif du projet devait consister à mettre au point un capteur ultrasonore pour la mesure conjointe de ces deux paramètres dans tous types de conduites en diamètre inférieur à 1 m. Ce capteur devait déterminer, par

une analyse temps-fréquence-direction des échos ultrasonores, la vitesse de l'écoulement et la concentration de MES par tranche de hauteur, ainsi que la répartition granulométrique des solides contenus.
Le capteur développé devait permettre de mesurer les flux de pollution beaucoup plus précisément, par un dispositif simple d'emploi et peu coûteux.

La démarche proposée consistait à construire un système expérimental d'après les brevets déposés par l'IMFS, à caractériser finement son comportement en laboratoire, puis à définir une méthodologie d'emploi au moyen d'essais de terrain (société IRH Environnement). Les points clés du développement concernaient la conception des transducteurs, de l'électronique associée et le traitement du signal.
La technologie devait être valorisée par les partenaires industriels membres du consortium : ULTRAFLUX (construction et commercialisation de matériel) et IRH (prestations de service). Le groupement d'intérêt public GEMCEA, associé au projet, pouvait agir comme prescripteur. Au plan scientifique cette technologie ouvre des perspectives intéressantes pour l'étude des fluides chargés et des dynamiques sédimentaires en général.

Bilan d'équipe

Plusieurs tests ont été effectués avec le prototype, en station dépuration, sur collecteur et en rivière. Des mesures effectuées sur la station d'épuration de Varangéville ont permis de valider sur site réel la mesure de profils de vitesses et ont permis de tester l'algorithme de mesure de hauteur d'eau. Sur ce site ont également été accomplis les premières mesures en continu pour effectuer des estimations granulométriques. Un enregistrement avec le prototype sur 24 heures du débit et des estimations de matières en suspension en entrée de station dépuration montrent des variations totalement cohérentes, à savoir une diminution nocturne des charges en MES en raison de l'arrêt de l'activité humaine et de façon plus fine une diminution par effet de dilution lors d'augmentation temporaire de débit par apport d'eau claire en journée.
Des enregistrements granulométriques ont été effectués, afin de finaliser les algorithmes de classification granulométrique par taille de particules. Les résultats obtenus ont montré que l'on observe non seulement les variations cycliques journalières de concentration massiques et de variation de tailles moyennes de particules, mais que l'on peut également estimer les concentrations massiques de matières en suspension provenant d'eaux usées et celles ayant pour origine les eaux pluviales.

Bilan personnel

Mon apport à ce projet a consisté au développement et au test de l'électronique numérique embarquée du capteur. Cet ensemble constitue la chaîne temps réel d'acquisition de données de tout le système.
La durée de cette action s'est étalée de septembre 2004 à juin 2007.

3.4.3 Le projet MARÉE

Cette action a pris pour nom : les Mesures Automatisées en RÉseau et leur Expertise (MAREE).
Ce projet s'est matérialisé dans un premier temps sous la forme d'une action ANVAR Lorraine tripartite d'aide à l'embauche. Le consortium était formé par la société IRH Environnement pour le volet industriel, et par l'IMFS et le Centre de Recherche en Automatique de Nancy (CRAN) pour l'expertise scientifique. Ce projet auquel j'ai fortement contribué au lancement, a débuté en avril 2003, pour se terminer en mai 2004.

Parmi les facteurs limitant l'usage de la mesure en continu dans le cycle de l'eau on peut citer : le manque d'incitation réglementaire, les limites de l'instrumentation disponible, la complexité d'exploitation de nombreuses données, qui résulte autant du point précédent que de la nature des phénomènes observés.
Le projet devait s'attacher à lever l'obstacle de cette complexité. Dans la pratique il s'avère souvent nécessaire de recourir à l'expertise dès la validation des mesures, puis lors de leur interprétation. L'expertise s'impose également lors de la constitution et de l'utilisation de capteurs logiciels, tout comme lors de la mise en œuvre de paramètres de substitution.
Le projet MAREE se proposait de tirer parti d'un certain nombre de réflexions et d'outils existants pour construire une méthodologie générale d'expertise des mesures en continu. Elle reposait principalement sur :

- l'adaptation aux mesures en continu et en temps réel d'un système expert ;
- la définition de capteurs logiciels basés sur différents algorithmes de traitement du signal, notamment la logique floue.

Les bénéfices attendus pour l'exploitation des stations de mesure (en situation d'expertise comme de routine) devaient concerner les opérations de maintenance des systèmes, d'interprétation des données, de prise de décision suite à un événement.

Bilan d'équipe

Mon ancien thésard S. Faucheux a passé un an de postdoc au CRAN, financé par une aide à l'embauche ANVAR. Une application grandeur réelle concernant la gestion d'une tour aéro-réfrigérante a été testée dans le cadre d'un projet européen intitulé « Coolsense ».

Bilan personnel

Mon apport personnel a consisté à assurer la jonction mesures-logique floue-conditionnement de l'information pour tous les points d'entrées-sorties du système expert.

Chapitre 4

Recherches sur les observations d'état

La restructuration de l'IMFS, et le recentrage thématique qui a suivi m'ont conduit à demander mon intégration au sein du Laboratoire du Génie de la Conception (LGECO) de l'INSA Strasbourg (EA 3938). Mes thématiques se sont recentrées sur la conception d'algorithmes embarquables, et le développement d'observateurs d'états non linéaire. Ces thématiques cadraient avec l'axe de recherche « modélisation-optimisation » du laboratoire.

4.1 Contexte international du domaine de Recherche

En automatique, la représentation d'état permet de modéliser un système dynamique sous forme matricielle en utilisant des variables d'état. Une fonction d'état est une fonction des variables d'état qui définissent l'état d'équilibre d'un système.
Dans les grandes lignes, on sépare alors un modèle en plusieurs parties distinctes :

- si le modèle possède une structure causale, des entrées, variables dans le temps, peuvent influer sur la réponse d'un système ; ces dernières le plus souvent connues, permettent alors le contrôle du système ;
- les variables d'états représentent les grandeurs physiques, et varient dans le temps ;
- les lois de comportement, qui utilisent les variables d'état à travers des fonctions, linéaires ou nom, influent directement sur la dynamique des grandeurs physiques ;
- les grandeurs de sorties, variables dans le temps, mesurables ou non, sont le résultat d'une combinaison de variables d'états, à travers des fonctions, linéaires ou non.

Très souvent, il arrive que l'on connaisse la structure et les lois de comportement des modèles des systèmes physique que l'on souhaite caractériser. Par contre, pour de nombreuses raisons pratiques et ou technologiques, on n'accède pas directement par la mesure à de nombreuses grandeurs physiques. Ces dernières peuvent de surcroît être de mauvaise qualité (bruits de mesure importants, conditions de pause incorrectes, dérives ...), et distribuées sur un réseau de communication. La connaissance de ces variables d'états est déterminantes dans bien des cas :

- pour déterminer à un instant donné quelle grandeur physique est prépondérante sur les sorties mesurées ;
- pour élaborer une démarche diagnostic dans le cas d'une surveillance de procédé ;
- pour élaborer une stratégie de commande permettant de contrôler la dynamique des variables d'états ;
- pour éventuellement identifier les paramètres des lois de comportements, ou analyser leurs dérives.

L'observation d'état est une branche mathématique de la théorie des systèmes, permettant de reconstruire une image complète d'un système à partir d'un nombre limité de points de mesures. De 2008 à 2016, mon activité recherche s'est centrée sur le développement de nouveaux algorithmes d'observation d'états adaptés à une gamme plus ou moins large de systèmes dynamiques non linéaires.
Initiée par les travaux de *Luenberger* Luenberger (1966), cette démarche, connue et utilisée de longue date pour des modèles linéaires, reste un verrou scientifique dans le domaine du non linéaire, du fait des difficultés rencontrées pour la mise en place d'une méthode de synthèse des paramètres, et la caractérisation des domaines de stabilités de tels outils. L'utilisation de ces méthodes de calculs débouchent sur de nombreuses applications. On peut citer par exemple de façon non exhaustive :

- la modélisation de systèmes biomécaniques et de systèmes cognitifs ;
- la caractérisation de fluides complexes ;
- la modélisation de procédés thermiques ;
- la modélisation dans le domaine de l'eau et l'environnement ;
- la modélisation de réacteurs chimiques ou biologiques ;
- la modélisation de systèmes robotiques ;

- la modélisation de véhicules ou d'aéronefs ;
- la cryptographie et le diagnostic de procédés ;
- la synchronisation avec des systèmes chaotiques ;
- la commande par retour d'états ...

Depuis les travaux de (Luenberger 1966), les observateurs d'état sont utilisés autant en modélisation, qu'en contrôle ou identification de systèmes linéaires ou non linéaires. Ils continuent de faire l'objet de nombreuses études, du fait du type de non linéarités et ou du type de transformations que l'on effectue sur le modèle de représentation des lois de comportement.
On peut entre autre citer les travaux de (Bastin & Gevers 1988) concernant certains systèmes non linéaires transformables en une forme canonique d'observation. Pour réduire l'ordre et de la complexité de systèmes dynamiques, (J.H. Stoev 2002) exploite des observateurs non linéaires utilisant des splines. Dans le même ordre d'idée, la forme canonique de *Luenberger* étendue (ELO) (Krener & Isidori 1983, Zeitz 1987, Bastin & Gevers 1988, Zheng et al. 2009) consiste à produire une approximation à l'ordre 1 des non linéarités (Bestle & Zeitz 1983, Ciccarella et al. 1993), et cette approche peut être étendue à des approximations d'ordre plus élevées (Röbenack & Lynch 2004).
Le même auteur (Röbenack & Lynch 2006) démontre qu'un observateur utilisant une forme d'observation canonique non linéaire partielle (POCF) possède des conditions d'existence, d'observabilité et d'intégrabilité plus restreintes que la forme canonique ELO. L'avantage de ce type d'approche est de faciliter l'optimisation nécessaire à la synthèse des gains de l'observateur. L'inconvénient réside dans le fait de négliger des composantes de la dynamique des systèmes.

Dans un autre ordre d'idées, les observateurs non linéaires à mode glissant utilisent une approche quasi Newtonienne, appliquée après une pseudo-dérivation du signal de sortie (Veluvolu et al. 2007, Efimov & Fridman 2011). L'inconvénient réside dans le fait d'amplifier le bruit de mesure à travers les pseudo-dérivations.

Une autre approche est celle des observateurs utilisant le filtrage de *Kalman* étendu (EKF) pour réaliser la transformation de systèmes non linéaires (Boker & Khalil 2013, Rauh et al. 2013) ; (Besançon et al. 2010) exploite un observateur adaptatif basé sur le filtrage de *Kalman* appliqué à des systèmes physiques sur lesquels une transformation de *Liénard* est possible. Dans un tel cas de figure, la difficulté réside dans la synthèse des gains de l'observateur.

La principale difficulté rencontrée dans l'utilisation d'observateurs avec les systèmes non linéaires réside dans la mise en place d'une méthode de synthèse des paramètres de ces observateurs. C'est pour contourner cette difficulté qu'est apparue la technique des observateurs à grands gains proposée par (Tornambè 1992, Bullinger et al. 1998, Besançon et al. 2004, Nikiforov et al. 2005, Farza et al. 2011, Mobki et al. 2015). Cette approche consiste à réduire l'erreur d'observation dans une gamme d'amplitudes prédéterminées en ayant la possibilité de s'abstraire des paramètres. Le point faible de la méthode vient de sa sensibilité au bruit et aux fluctuations exogènes du système.
Pour certaines utilisations des observateurs, notamment le cryptage, le bruit n'est pas un obstacle (Ibrir 2009, Ghosh et al. 2010, Martínez-Guerra et al. 2011). Ceci peut s'avérer très différent dans le cas de processus industriels, comme le montre (Bodizs et al. 2011), où les performances de divers observateurs utilisant l'approche ELO, EKF ou filtre de *Kalman* intégré sont comparées. L'influence du bruit et des incertitudes a bien été mise en évidence. Les résultats obtenus par les observateurs ELO sortent du lot, qui permettent une estimation d'état sur des systèmes fortement perturbés. Les observateur ELO intégrant une stratégie proportionnelle intégrale (Söffker et al. 1995, Morales & Ramirez 2002) démontrent en plus leur capacité de compensation des biais de mesures.

L'impact des incertitudes paramétriques des modèles non linéaires sur les observations d'états est bien mis en évidence dans les études proposées par (Chen et al. 2011, Bouraoui et al. 2015, Khalifa & Mabrouk 2015). Une possible solution à ce type de problème est l'usage d'observateurs adaptatifs (Tyukina et al. 2013, Alma1 & Darouach 2014, Farza et al. 2014), en particulier sur des systèmes discrétisés et perturbés. Une autre approche (Mazenc & Dinh 2014, Thabet et al. 2014) consiste à utiliser des observateurs à intervalles, où l'on recherche la marge d'incertitude dans laquelle se situe chaque variable d'état.

D'autres types d'observateurs non linéaires existent. On peut citer ceux utilisant une décomposition de type *Takagi-Sugeno* (Bezzaoucha et al. 2013, Guerra et al. 2015), où les lois de comportement sont décrites par la superposition de plusieurs modèles simples. La contribution de chacun d'eux est pondérée par une fonction floue, en fonction du Contexte d'utilisation des variables d'état. L'avantage est alors d'utiliser des modèles d'ordre et de complexité réduits. L'inconvénient provient alors de la difficulté de concevoir les règles pilotant les commutations floues permettant de garantir la stabilité de l'observation.

On peut enfin citer les techniques exploitant les symétries et les invariants (Menini & Tornambè 2011), ou encore

les techniques immersives (Back & Seo 2008) qui permettent de reformuler les non linéarités en augmentant l'ordre des équations différentielles à observer.

4.2 Positionnement des recherches menées

La ligne directrice de mes travaux tient en quelques mots : facilité de mise en œuvre, de caractérisation des conditions de stabilité, de robustesse au bruit. Ceci a conduit a quelques règles qui ont guidé la philosophie des études menées.

- A aucun moment on ne cherche à solutionner un quelconque système d'équation. Toute la démarche se base exclusivement sur l'analyse en stabilité des systèmes proposés.
- A aucun moment on ne linéarise les lois de comportement des systèmes que l'on cherche à observer.
- Pour faciliter la simplicité de mise en œuvre des algorithmes proposés quand à la synthèse des gains de l'observateur, on privilégie l'utilisation de critères algébriques simples au détriment des outils d'optimisation classiquement utilisés. L'emploi de ce type de techniques simples est rendu possible par une renormalisation de l'échelle de temps des équations d'état.
- L'analyse de stabilité des observateurs développés se fait sur les erreurs d'observation et non sur les variations d'états (où distances d'états).
- Toutes les transformations utilisées sont temporelles et continues.
- Tous les algorithmes proposés sont mis en œuvre et testés à l'aide d'un compilateur et de bibliothèques de calculs qui sont des développements propres, implantables sur tous types de plate-forme, allant des environnements embarquables aux calculateurs distribués.

Une première contribution (Schwaller et al. 2008) a permis l'observation de systèmes comportant des non linéarités polynomiales, jusqu'à l'ordre 3. La stabilité de ces observateur, bien que globalement asymptotique, dépend toutefois du type de non linéarités.

Les années 2009 à 2012 ont marqué une pause dans mes activités de Recherches. Un accident sportif survenu en 2009, deux opérations au genoux, une longue rééducation ont été suivi en 2011 et 2012 du décès de mon père et beau-père. Tous ces évènements ont fortement ralenti les développements suivants. J'ai toutefois pu répondre à deux invitations, l'une au Colloque National de la Recherche en IUT (Schwaller et al. 2009), l'autre à une conférence invitée au Laboratoire de Mécanique et Génie Civil de Montpellier (Schwaller 2011).

En modifiant la démonstration de la stabilité faite en (Schwaller et al. 2008) pour la rendre indépendante du type de non linéarités, il a été possible (Schwaller et al. 2013) de perfectionner l'observateur non linéaire proportionnel intégral pour des systèmes à une entrée une sortie décomposables selon la transformation de *Fliess* (Fliess 1990) et uniformément observables au sens de *Hermann et Gauthier* (Hermann & Krener 1977, Gauthier & Bornard 1981). La partie intégrale de l'observateur permet la caractérisation des variations exogènes lentes venant « polluer » les mesures. Le type de non-linéarités traitées (devant être intégrables), et l'ordre (limité à 3) des équations différentielles sont rigoureusement définis, et la synthèse des paramètres fait appel au critère algébrique des polynômes normaux d'amortissement de *Naslin* (Naslin 1960, 1963). La mise en œuvre de l'observateur s'en trouve grandement simplifiée.
Toutefois, on peut noter que les conditions limites de stabilité peuvent s'avérer restrictives, en particulier pour les systèmes d'ordre 2, du fait de l'augmentation de la constante de *Lipschitz*.

Plus généralement, un grand nombre de systèmes non-linéaires à entrées multivariables et une seule sortie (MISO) sont représentables par :

$$\dot{\underline{z}}(t) = \underline{s}(t) \tag{4.1a}$$

$$y(t) = \underline{d}^T \ \underline{z}(t) + \Phi(t) \tag{4.1b}$$

$$\underline{s}(t) = \begin{bmatrix} s_1 \left[\ \underline{z}(t), \underline{u}_1(t) \ \right] \\ \dots \\ s_n \left[\ \underline{z}(t), \underline{u}_1(t) \ \right] \end{bmatrix} \tag{4.1c}$$

$$\underline{d}^T = \begin{bmatrix} d_1 & \dots & d_n \end{bmatrix} \tag{4.1d}$$

avec :
n : l'ordre du système d'équations différentielles
m : le nombre d'entrées indépendantes
$\underline{u}_1(t)^T$: le vecteur $[u_{11}(t), \dots u_{m1}(t)]$ des m entrées indépendantes
$y(t)$: la variable de sortie mesurable
$\underline{z}^T(t)$: le vecteur d'état $[\ z_1(t) \dots z_n(t)\ ]$
$\underline{d}^T$: le vecteur des paramètres de sortie du système

$\Phi(t)$: la fonction non linéaire des entrées $\underline{u}_1(t)$
$s_i\,[\,\underline{z}(t), \underline{u}_1(t)\,]$: une des n fonctions non linéaires du vecteur d'état $\underline{s}(t)$.

De tels systèmes se retrouvent souvent pour des systèmes robotiques non linéaires, sous la forme de fonctions trigonométriques. D'autres systèmes contiennent des non linéarités polynomiales (attracteurs étranges, fonctions de *Bernoulli*, retours élastiques non linéaires), de fractions polynomiales, ou différentes fonctions usuelles simples ...
Les n fonctions non linéaires du vecteur $\underline{s}(t)$ utilisent comme entrées un vecteur de m entrées indépendantes $\underline{u}_1(t)$, de même que le vecteur d'état $\underline{z}(t)$. Une telle représentation permet entre autre la description de système bilinéaires. Dans nos étude, nous nous limitons aux fonctions continues en tous points de type C^1.
On considère que la sortie mesurable est une combinaison linéaire de $\underline{z}(t)$, superposée à une fonction non linéaire $\Phi(t)$. Pour un ingénieur ou un physicien, beaucoup d'applications prennent une telle forme.
Souvent, les systèmes non linéaires (4.1) sont transformables (Fliess 1990) sous une forme canonique de régulation (Zeitz 1985), et s'écrivent alors :

$$\dot{\underline{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{f}(t) \tag{4.2a}$$
$$y(t) = \underline{c}^T\,\underline{x}(t) + \Phi(t) \tag{4.2b}$$
$$\underline{A} = \delta_{ij} \quad j = i+1,\ i = 1 \ldots n-1 \tag{4.2c}$$
$$\underline{c}^T = \left[\begin{array}{cccc} c_1 & \ldots & c_n \end{array}\right] \tag{4.2d}$$
$$\underline{f}(t)^T = \left[\begin{array}{cccc} 0 & \ldots & 0 & \Psi\,[\,\underline{x}(t),\,\underline{U}(t)\,] \end{array}\right] \tag{4.2e}$$

avec :
$\underline{u}_i(t)$: les $(i-1)$ dérivées temporelles de $\underline{u}_1(t)$, pour lesquelles $\underline{u}_i(t)^T = [u_{1i}(t), \ldots u_{mi}(t)]$ avec $i = 2 \ldots n$
$\underline{U}(t) = [\underline{u}_1(t) \ldots \underline{u}_n(t)]$: la matrice d'entrées de dimension $n \times m$, où l'ensemble des n vecteurs $\underline{u}_i(t)$
$x_i(t)$: les $(i-1)$ dérivées temporelles de $x_1(t)$
$\underline{x}^T(t)$: le vecteur d'état $[\,x_1(t), \ldots x_n(t)]$
$\underline{c}$: le vecteur des paramètres de sortie du système transformé
$\theta \leqslant n$: l'index du dernier coefficient $c_i \neq 0$
$\Psi\,[\,\underline{x}(t), \underline{U}(t)]$: une fonction scalaire non linéaire de type C^1
$\underline{A}$: une matrice $n \times n$ dont la dernière ligne est nulle.

Dans (Schwaller et al. 2016) un nouvel observateur adapté à cet espace transformé est proposé, afin de relâcher les conditions limites de stabilité étudiées en (Schwaller et al. 2013), et d'élargir le champ d'application aux systèmes d'ordre n, pour couvrir le même domaine que celui proposé par (Gauthier et al. 1992).
Dans la § 4.3.1, nous le décrirons en détail, ainsi que le rôle de chacun de ses composantes.

Cette nouvelle approche, est entièrement déterministe, et la seule exigence est que la fonction non linéaire $\Psi\,[\,\underline{x}(t), \underline{U}(t)]$ soit au moins de type C^1, et que sa dérivée temporelle $d\,\Psi\,[\,\underline{x}(t), \underline{U}(t)]\,/dt$ soit globalement de type *Lipschitz* en $\underline{x}(t)$ (Raghavan & Hedrick 1994).
Cette hypothèse peut être assouplie en supposant que cette dérivée est seulement localement *Lipschitz*, ou peut être transformée de manière adéquate, ce qui est possible pour de nombreuses applications pratiques (équations de *Düffing, van der Pol, Bernoulli*, pendules inversés, modèles non linéaires de friction pour moteurs à courant continu ou actionneurs, bioréacteurs, attracteurs étrange ...).

Dans la § 4.3.2, on étudiera la dynamique du nouvel observateur, et l'on démontrera l'équation différentielle définissant la distance d'état système physique-observateur.
Dans la § 4.3.3, on démontrera la stabilité des estimations d'état, et leur caractère exponentiellement convergent.
Dans la § 4.3.4, on réalisera la synthèse des gains de l'observateur. Pour la réaliser à l'aide de critères algébriques simples, on choisit pour le reste de l'étude de reformuler (4.2) sous la forme d'un système en temps non dimensionnel (Gille et al. 1988), et amène à la définition suivante.

Définition 1. *Soit une pulsation normative $\omega_o = 2\pi/T_o$, qui doit nous permettre de changer à volonté l'échelle de temps du système* (4.2) *:*

$$\dot{\underline{x}}(\tau) = \underline{A}\,\underline{x}(\tau) + \underline{f}(\tau) \tag{4.3a}$$
$$y(\tau) = \widetilde{\underline{c}}^T\,\underline{x}(\tau) + \Phi(\tau) \tag{4.3b}$$
$$\widetilde{\underline{c}}^T = \left[\begin{array}{cccc} \widetilde{c}_1 & \ldots & \widetilde{c}_n \end{array}\right] \tag{4.3c}$$
$$\underline{f}(\tau)^T = \left[\begin{array}{cccc} 0 & \ldots & 0 & \widetilde{\Psi}\,[\,\underline{x}(\tau), \underline{U}(\tau)] \end{array}\right] \tag{4.3d}$$

avec :

$$\tau = \omega_o\, t, \quad \dot{x}_n(t) = \dot{x}_n(\tau)\, {\omega_o}^n \tag{4.4a}$$

$$u_{ij}(t) = u_{ij}(\tau)\, {\omega_o}^{i-1}, \quad x_i(t) = x_i(\tau)\, {\omega_o}^{i-1} \tag{4.4b}$$

$$\widetilde{c}_i = c_i\, {\omega_o}^{i-1}, \quad z_i(t) = z_i(\tau), \quad i = 1 \dots n \tag{4.4c}$$

Le système (4.4) définit la dilatation ou la contraction de la représentation d'état, ainsi que ses nouveaux paramètres, sans changer la forme du signal $x_i(\tau)$. Cette nouvelle représentation d'état va jouer un rôle important pour définir les conditions limites de stabilité de nos observations moins restrictive que (Schwaller et al. 2013), et pour réaliser la synthèse des gains du nouvel observateur. Pour la fonction Ψ, cela se traduit par la relation de changement d'échelle temporelle suivante :

$$\Psi\,[\,\underline{x}(t), \underline{U}(t)] = {\omega_o}^n\ \widetilde{\Psi}\,[\,\underline{x}(\tau), \underline{U}(\tau)] \tag{4.5}$$

Un des objectifs majeurs qui a motive l'étude de ce nouveau type d'observateur consiste à dissocier les estimations d'état utilisées pour reconstruire les fonctions $\Psi(t)$ des états utilisés pour réaliser la correction d'erreurs, afin de contrer la sensibilité au bruit bien connue des observateurs à grands gains. On cherche également à conserver la structure PI des corrections d'erreur de (Schwaller et al. 2013), qui permet de compenser les perturbations exogènes du modèle du système physique, et de garder un gain statique unitaire à la dynamique de convergence du nouvel observateur.

Le passage de la forme transformée (4.2) à la représentation initiale (4.1) se fait à l'aide de :

$$\underline{z}(t) = \underline{g}(t) \tag{4.6a}$$

$$\underline{g}(t)^T = \left[\ g_1\,[\,\underline{x}(t), \underline{U}(t)\,]\ \dots\ g_n\,[\,\underline{x}(t), \underline{U}(t)\,]\right] \tag{4.6b}$$

avec : $\underline{g}(t)$ le vecteur de n fonctions de transformations non linéaires inverses $g_i\,[\underline{x}(t),\ \underline{U}(t)]$ qui lient $\underline{x}(t)$ à $\underline{z}(t)$. En temps non dimensionnel, ces fonctions inverses s'écrivent :

$$\underline{z}(\tau) = \underline{g}(\tau) \tag{4.7a}$$

$$\underline{g}(\tau)^T = \left[\ g_1\,[\,\underline{x}(\tau), \underline{U}(\tau)\,]\quad \dots\quad g_n\,[\,\underline{x}(\tau), \underline{U}(\tau)\,]\right] \tag{4.7b}$$

avec (4.4) pour réaliser les changements de variables de $x_i(t)$ et $u_{ij}(t)$.

Si le nouvel observateur décrit en § 4.3.1 est à même de fournir une estimation robuste et non biaisée de $\underline{x}(t)$, ceci n'est pas toujours le cas pour $\underline{z}(t)$. Du fait de la non linéarité des fonctions $g_i\,[\,\underline{x}(t), \underline{U}(t)\,]$, de petites perturbations superposées aux estimées $\underline{x}(t)$ peuvent augmenter considérablement les perturbations affectant les estimées de $\underline{z}(t)$.

Le but de nos recherchent actuelles consiste à résoudre ce type de problèmes, et est abordé dans la § 4.4. On y introduira le concept d'observation des fonctions de transformations inverses $g_i\,[\,\underline{x}(t), \underline{U}(t)\,]$. Ce faisant, on exploite la résistance au bruit des observateurs (Bodizs et al. 2011), afin de disposer d'un outil capable de limiter l'effet du bruit de mesure sur les estimées de $\underline{z}(t)$.

Enfin, en § 4.5, deux simulations viennent illustrer les développements des sections § 4.3 et § 4.4.

4.3 Structure et synthèse de l'observateur sous forme canonique de régulation

4.3.1 Définitions de l'observateur

Dans un premier temps, on isole la composante $x_1(\tau)$ de (4.3b) qui doit servir à déterminer l'erreur d'observation. Pour obtenir $y_1(\tau)$, l'estimée de la variable $x_1(\tau)$, trois cas distincts doivent être envisagés. Pour $\theta = 1$:

$$y_1(\tau) = \frac{y(\tau) - \Phi(\tau)}{\widetilde{c}_1} \tag{4.8}$$

Pour $\theta = 2$, il vient :

$$\dot{y}_1(\tau) = -\frac{\widetilde{c}_1}{\widetilde{c}_2}\ y_1(\tau) + \frac{y(\tau) - \Phi(\tau)}{\widetilde{c}_2} \tag{4.9}$$

Dans le cas le plus général où $\theta > 2$, $y(\tau) - \Phi(\tau)$ est filtré par :

$$\dot{\underline{w}}(\tau) = \underline{K}\,\underline{w}(\tau) + \underline{k}\,[\,y(\tau) - \Phi(\tau)\,] \tag{4.10a}$$

$$\underline{K} = \begin{bmatrix} 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \dots & 0 & 0 & 1 \\ -\dfrac{\widetilde{c}_1}{\widetilde{c}_\theta} & \dots & \dots & -\dfrac{\widetilde{c}_{\theta-1}}{\widetilde{c}_\theta} \end{bmatrix} \tag{4.10b}$$

$$\underline{w}(\tau)^T = \begin{bmatrix} y_1(\tau) & \dots & y_{\theta-1}(\tau) \end{bmatrix}, \quad \underline{w}(0) = \underline{0} \tag{4.10c}$$

$$\underline{k}^T = \begin{bmatrix} 0 & \dots & 0 & 1/\widetilde{c}_\theta \end{bmatrix} \tag{4.10d}$$

Pour analyser l'effet du filtrage, on réécrit (4.3b) sous forme scalaire, en ignorant $\widetilde{c}_{\theta+1}\ \dots\ \widetilde{c}_n$, qui sont tous nuls :

$$y(\tau) - \Phi(\tau) = \sum_{i=1}^{\theta} \widetilde{c}_i\, x_i(\tau) \tag{4.11}$$

Si (4.11) est inséré dans (4.10a), (4.9) ou (4.8) comme une fonction de θ, il vient :

$$\sum_{i=1}^{\theta} \widetilde{c}_i\, y_i(\tau) = \sum_{i=1}^{\theta} \widetilde{c}_i\, x_i(\tau) \tag{4.12a}$$

$$\dot{y}_{\theta-1}(\tau) = y_\theta(\tau) \quad \theta \geqslant 2 \tag{4.12b}$$

La transformée de *Laplace* de (4.12a) donne la fonction de transfert :

$$y_1(s)/x_1(s) = 1 \tag{4.13}$$

Pour la suite des développements, on utilise $y_1(\tau)$ pour déterminer l'erreur d'observation

Définition 2. *Pour construire les estimations d'état $\underline{v}(\tau)$ du système* (4.3)*, on définit la structure observatrice suivante :*

$$\dot{\breve{\underline{x}}}(\tau) = \underline{A}\,\breve{\underline{x}}(\tau) + \widetilde{\underline{f}}(\tau) + \breve{\underline{h}}\,\Delta y_1(\tau) \tag{4.14a}$$

$$\dot{\widehat{\underline{x}}}(\tau) = \underline{A}\,\widehat{\underline{x}}(\tau) + \breve{\underline{A}}\,\breve{\underline{x}}(\tau) + \widehat{\underline{h}}\,\Delta y_1(\tau) \tag{4.14b}$$

$$\Delta y_1(\tau) = x_1(\tau) - \widehat{x}_1(\tau) \tag{4.14c}$$

$$\widetilde{\underline{f}}(\tau)^T = \begin{bmatrix} 0 & \dots & 0 & \widetilde{f}(\tau) \end{bmatrix} \tag{4.14d}$$

$$\dot{I}_0(\tau) = h_0\,\Delta y_1(\tau) \tag{4.14e}$$

$$\widetilde{f}(\tau) = I_0(\tau) + \widetilde{\Psi}\,[\,\underline{v}(\tau), \underline{U}(\tau)\,] \tag{4.14f}$$

$$\breve{\underline{x}}(\tau)^T = \begin{bmatrix} \breve{x}_2(\tau) & \dots & \breve{x}_n(\tau) \end{bmatrix} \tag{4.14g}$$

$$\widehat{\underline{x}}(\tau)^T = \begin{bmatrix} \widehat{x}_1(\tau) & \dots & \widehat{x}_{n-1}(\tau) \end{bmatrix} \tag{4.14h}$$

$$\underline{v}(\tau)^T = \begin{bmatrix} \widehat{x}_1(\tau) & \breve{\underline{x}}(\tau)^T \end{bmatrix} \tag{4.14i}$$

$$\widehat{\underline{x}}(0) = \breve{\underline{x}}(0) = \underline{0}, \quad I_0(0) = 0 \tag{4.14j}$$

$$\breve{\underline{h}}^T = \begin{bmatrix} \underline{0}^T & h_1 \end{bmatrix} \tag{4.14k}$$

$$\widehat{\underline{h}}^T = \begin{bmatrix} h_n & \dots & h_2 \end{bmatrix} \tag{4.14l}$$

$$\underline{h}^T = \begin{bmatrix} \widehat{\underline{h}}^T, & h_1 \end{bmatrix} \tag{4.14m}$$

$$\underline{A} = \delta_{ij}, \quad j = i+1,\ i = 1 \dots n-1 \tag{4.14n}$$

$$\breve{\underline{A}} = \begin{bmatrix} 0 & \dots & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ \vdots & & 0 & \vdots \\ 0 & \dots & \dots & 1 \end{bmatrix} \tag{4.14o}$$

avec $\breve{\underline{x}}(\tau)$ (4.14g) *et $\widehat{\underline{x}}(\tau)$* (4.14h) *deux vecteurs d'états distincts de dimension $n-1$, couplés à travers les matrices $\underline{A}$* (4.14n) *et $\breve{\underline{A}}$* (4.14o)*, toutes deux de dimension $(n-1)\times(n-1)$. Les vecteurs $\breve{\underline{h}}$ et $\widehat{\underline{h}}$ sont également de dimension $n-1$. La matrice $\underline{A}$ est construite à l'aide de l'opérateur de Kronecker, qui met à 1 la diagonale supérieure.*

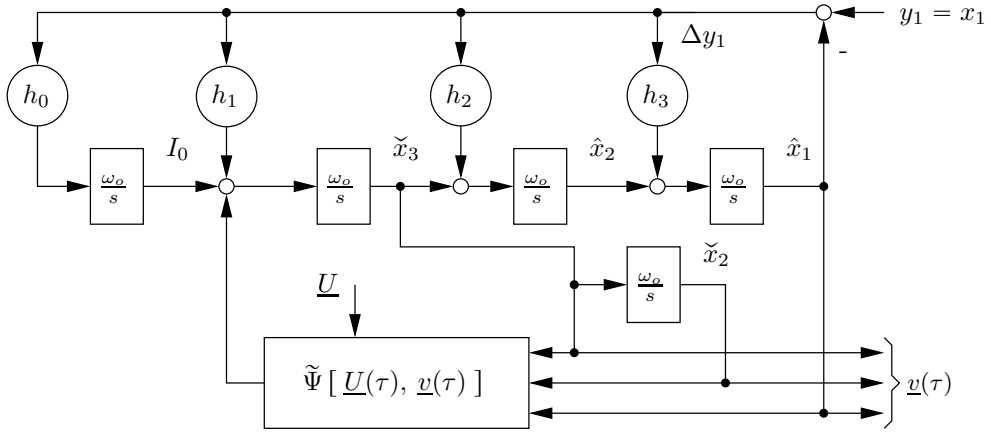


FIGURE 4.1: Observateur d'ordre 3

La FIGURE 4.1 page 29 illustre le diagramme fonctionnel d'un tel observateur à l'ordre 3.
Le vecteur augmenté $\underline{v}(\tau)$ ((4.14i),(4.14h) et (4.14g)) est utilisé comme estimation de $\underline{x}(\tau)$ et comme variable de la fonction $\widetilde{\Psi}\,[\,\underline{v}(\tau),\underline{U}(\tau)\,]$ (4.14f). Le vecteur d'état $\underline{\widehat{x}}(\tau)$ (4.14b) est un observateur exploitant l'erreur d'observation $\Delta y_1(\tau)$ (4.14c) *via* les gains h_i (4.14m), et sert à corriger les distances d'état système-observateur. Sur la FIGURE 4.1, par exemple, on a :

$$\begin{aligned}
\underline{\widehat{x}}(\tau)^T &= \left[\,\widehat{x}_1(\tau) \quad \widehat{x}_2(\tau)\right]\\
\underline{\breve{x}}(\tau)^T &= \left[\,\breve{x}_2(\tau) \quad \breve{x}_3(\tau)\right]\\
\underline{v}(\tau)^T &= \left[\,\widehat{x}_1(\tau) \quad \breve{x}_2(\tau) \quad \breve{x}_3(\tau)\right]\\
\underline{\widehat{h}}^T &= \left[\,h_3 \quad h_2\right]\\
\underline{\breve{h}}^T &= \left[\,0 \quad h_1\right]
\end{aligned}$$

Le choix d'utiliser deux vecteurs d'états $\underline{\widehat{x}}(\tau)$ et $\underline{\breve{x}}(\tau)$ est motivé du fait des $n-1$ intégrations successives de $\dot{\breve{x}}_n(\tau)$, pour lesquelles aucune réinjection d'erreur $\underline{\widehat{h}}\ \Delta y_1(\tau)$ n'est utilisée. Ceci permet d'augmenter la robustesse des estimations aux bruits de mesures, qui affectent en général la variable $y_1(\tau)$. On contourne ainsi un point faible bien connu des observateurs à grand gain, à savoir leur sensibilité aux bruits. Le deuxième avantage provient de la fonction non linéaire $\widetilde{\Psi}\,[\underline{v}(\tau),\underline{U}(\tau)]$ qui n'est plus sujette aux conditions restrictives de (Schwaller et al. 2013), et couvrent l'ensemble des systèmes transformables de *Fliess* (Fliess 1990). Le vecteur $\underline{\widetilde{f}}(\tau)$ (4.14d), de dimension $n-1$, compense les effets de $\underline{f}(\tau)$, ainsi que les possibles perturbations exogènes au système (4.2) grâce à l'intégrateur $I_0(\tau)$ (4.14e). On note qu'à l'ordre deux, pour un gain $h_0 = 0$ inhibant l'intégrateur I_0, l'observateur devient similaire à celui proposé par (Gauthier et al. 1992) pour un système SISO.

4.3.2 Caractérisation de la dynamique de l'erreur d'observation

On cherche à présent à caractériser la dynamique de l'observateur en cherchant une équation différentielle liant les deux variables d'états $\underline{\widehat{x}}(\tau)$ et $\underline{\breve{x}}(\tau)$ à l'erreur d'observation $\Delta y_1(\tau)$ et à ses dérivées successives. On introduit toutes les notations utiles à la mise sous forme matricielle de l'erreur d'observation. Pour y parvenir, on déduit de (4.14b) la relation récursive utilisée pour générer les estimations successives $\widehat{x}_i(\tau)$ en fonction de l'erreur $\Delta y_1(\tau)$:

$$\begin{aligned}
\widehat{x}_{i+1}(\tau) &= \dot{\widehat{x}}_i(\tau) - h_{n-i+1}\ \Delta y_1(\tau)\\
i &= 1 \ldots n-2
\end{aligned} \tag{4.15}$$

Pour simplifier l'expression des dérivées successives des états $\widehat{x}_i(\tau)$, on introduit :

$$\underline{\widetilde{x}}(\tau)^T = \left[\,\widetilde{x}_1(\tau) \quad \ldots \quad \widetilde{x}_n(\tau)\right] \tag{4.16}$$

dont les composants sont définis par :

$$\widetilde{x}_{i+1}(\tau) = \frac{d^{\,(i)}\,\widehat{x}_1(\tau)}{d\tau^{\,(i)}} \quad i = 0 \ldots n-1 \tag{4.17}$$

Par conséquent :

$$\dot{\widetilde{x}}_i(\tau) = \widetilde{x}_{i+1}(\tau) \quad i = 1 \dots n-1 \tag{4.18}$$

En utilisant (4.17)-(4.18), il est possible de réécrire l'erreur d'observation $\Delta y_1(\tau)$ (4.14c) et ses dérivées successives :

$$\Delta y_1(\tau) = x_1(\tau) - \widetilde{x}_1(\tau) \tag{4.19a}$$

$$\Delta y_{i+1}(\tau) = \frac{d^{\,(i)}\,\Delta y_1(\tau)}{d\tau^{\,(i)}} \qquad i = 1 \dots n-1 \tag{4.19b}$$

On en déduit :

$$\Delta y_i(\tau) = x_i(\tau) - \widetilde{x}_i(\tau) \quad i = 1 \dots n \tag{4.20a}$$

$$\Delta y_{i+1}(\tau) = \Delta \dot{y}_i(\tau) \quad i = 1 \dots n-1 \tag{4.20b}$$

$$\Delta \underline{y}(\tau)^T = \left[\Delta y_1(\tau) \quad \dots \quad \Delta y_n(\tau) \right] \tag{4.20c}$$

avec (4.20a) qui décrit la i ième dérivée de (4.19a), (4.20b) qui décrit sous une forme récurrente les dérivées successives, et (4.20c) la forme vectorielle déduite de (4.20a)-(4.20b).
Utilisons ces nouvelles notations pour décrire la dynamique de l'observateur. On insère (4.16) dans (4.14) pour un index i donné. On dérive temporellement la nouvelle expression à l'aide de (4.18) et (4.20a). L'expression $\dot{\widehat{x}}_i(\tau)$ ainsi obtenue peut être égalée à celle obtenue avec (4.15) pour un index $i+1$. Cette opération, menée successivement jusqu'à l'ordre $n-1$ donne :

$$\dot{\widehat{x}}_{n-1}(\tau) = \widetilde{x}_n(\tau) - \sum_{i=2}^{n-1} h_{i+1}\, \Delta y_i(\tau) \tag{4.21}$$

En égalant (4.21) à (4.14b) de rang $n-1$, il vient :

$$\dot{\widehat{x}}_{n-1}(\tau) = \breve{x}_n(\tau) + h_2\, \Delta y_1(\tau) \tag{4.22}$$

qui, inséré en (4.21) donne :

$$\breve{x}_n(\tau) = \widetilde{x}_n(\tau) - \sum_{i=2}^{n} h_i\, \Delta y_{i-1}(\tau) \tag{4.23}$$

La dépendance linéaire de $\breve{x}_n(\tau)$, $\widetilde{x}_n(\tau)$ et $\Delta \underline{y}(\tau)$ autorise la transformée de *Laplace* de (4.23) et son intégration successive, et du fait de (4.18), permet la détermination de la forme générale de $\breve{x}_i(s)$:

$$\breve{x}_i(s) = \widetilde{x}_i(s) - f_i(s) \quad i = 2 \dots n \tag{4.24a}$$

$$f_i(s) = \Delta y_1(s) \sum_{j=2}^{n} h_j\, s^k \quad k = i + j - n - 2 \tag{4.24b}$$

La transformée de *Laplace* inverse de (4.24a) s'écrit :

$$\breve{x}_i(\tau) = \widetilde{x}_i(\tau) - \mathcal{L}^{-1}\left\{ f_i(s) \right\} \tag{4.25}$$

Le second terme du membre de droite de (4.25) tend vers 0 si le vecteur $\Delta \underline{y}(\tau) \to \underline{0}$ pour $\tau \to \infty$. La dérivée de (4.23) peut être égalée à (4.14a) de rang $n-1$, et il vient :

$$\dot{\widetilde{x}}_n(\tau) = f(\tau) + \sum_{i=1}^{n} h_i\, \Delta y_i(\tau) \tag{4.26}$$

On cherche à présent à spécifier la dynamique des erreurs d'observations système-observateur. Pour y parvenir, on calcule la distance entre la composante n du vecteur $\dot{\underline{x}}(\tau)$ (4.3a) et $\dot{\widetilde{x}}_n(\tau)$ (4.26). Il vient :

$$\Delta \dot{y}_n(\tau) = \dot{x}_n(\tau) - \dot{\widetilde{x}}_n(\tau) \tag{4.27a}$$

$$= \Delta \widetilde{\Psi}(\tau) - I_0(\tau) - \sum_{i=1}^{n} h_i\, \Delta y_i(\tau) \tag{4.27b}$$

avec :

$$\Delta \widetilde{\Psi}(\tau) = \widetilde{\Psi}\left[\underline{x}(\tau), \underline{U}(\tau) \right] - \widetilde{\Psi}\left[\underline{\widetilde{x}}(\tau), \Delta \underline{y}(\tau), \underline{U}(\tau) \right] \tag{4.28}$$

La distance $\Delta\widetilde{\Psi}(\tau)$ (4.28), de type C^1, est obtenue à partir des fonctions non linéaires $\widetilde{\Psi}\left[\,\underline{x}(\tau),\underline{U}(\tau)\right]$ et $\widetilde{\Psi}\left[\underline{v}(\tau),\underline{U}(\tau)\right]$ (4.14f), elles aussi de type C^1.
Il est maintenant utile d'écrire (4.27b) sous une forme matricielle de dimension $n \times n$, en utilisant (4.20a), (4.27b) et (4.14d) :

$$\Delta\dot{\underline{y}}(\tau) = \underline{\widetilde{A}}\ \Delta\underline{y}(\tau) + \Delta\underline{f}(\tau) \tag{4.29a}$$

$$\Delta\underline{f}(\tau) = \underline{f}(\tau) - \underline{\widetilde{f}}(\tau) \tag{4.29b}$$

$$\underline{\widetilde{A}} = \begin{bmatrix} 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \dots & 0 & 0 & 1 \\ -h_1 & \dots & \dots & -h_n \end{bmatrix} \tag{4.29c}$$

Le terme $\underline{\widetilde{f}}(\tau)$ (4.14f) utilisé dans (4.29b) inclut $I_0(\tau)$ (4.14e). Ceci induit une dérivation temporelle de (4.27b) pour obtenir la représentation d'état complète de l'équation différentielle des erreurs d'observation. Pour y parvenir, il est nécessaire d'introduire les états et les distances supplémentaires :

$$\underline{u}_{n+1}(\tau) = \dot{\underline{u}}_n(\tau) \tag{4.30a}$$

$$x_{n+1}(\tau) = \dot{x}_n(\tau) \tag{4.30b}$$

$$\widetilde{x}_{n+1}(\tau) = \dot{\widetilde{x}}_n(\tau) \tag{4.30c}$$

$$\breve{x}_{n+1}(\tau) = \dot{\breve{x}}_n(\tau) \tag{4.30d}$$

$$\Delta y_{n+1}(\tau) = \Delta\dot{y}_n(\tau) \tag{4.30e}$$

qui trouvent leur place dans les matrices et les vecteurs augmentés :

$$\underline{U}_a(\tau)^T = \left[\ \underline{U}(\tau)^T \quad \underline{u}_{n+1}(\tau)\right] \tag{4.31a}$$

$$\underline{x}_a(\tau)^T = \left[\ \underline{x}(\tau)^T \quad x_{n+1}(\tau)\right] \tag{4.31b}$$

$$\underline{\widetilde{x}}_a(\tau)^T = \left[\ \underline{\widetilde{x}}(\tau)^T \quad \widetilde{x}_{n+1}(\tau)\right] \tag{4.31c}$$

$$v_a(\tau)^T = \left[\ \underline{v}(\tau)^T \quad \breve{x}_{n+1}(\tau)\right] \tag{4.31d}$$

$$\Delta\underline{y}_a(\tau)^T = \left[\ \Delta\underline{y}(\tau)^T \quad \Delta y_{n+1}(\tau)\right] \tag{4.31e}$$

$$= \underline{x}_a(\tau)^T - \underline{\widetilde{x}}_a(\tau)^T \tag{4.31f}$$

et de calculer séparément la dérivée temporelle de (4.28) :

$$\Delta\dot{\Psi}(t) = {\omega_o}^{n+1}\ \Delta\dot{\widetilde{\Psi}}(\tau) \tag{4.32a}$$

$$\Delta\dot{\widetilde{\Psi}}(\tau) = \dot{\widetilde{\Psi}}\left[\ \underline{x}_a(\tau),\underline{U}_a(\tau)\ \right] - \dot{\widetilde{\Psi}}\left[\ \underline{\widetilde{x}}_a(\tau),\Delta\underline{y}_a(\tau),\underline{U}_a(\tau)\ \right] \tag{4.32b}$$

La dérivée temporelle de (4.27b), en utilisant (4.14e), (4.20a), (4.30), (4.31) et (4.32) est écrite sous forme matricielle, et permet de décrire le système d"équation différentielle qui pilote l'erreur d'observation $\Delta y_1(\tau)$:

$$\Delta\dot{\underline{y}}_a(\tau) = \underline{A}_a\ \Delta\underline{y}_a(\tau) + \Delta\dot{\underline{\widetilde{\Psi}}}(\tau) \tag{4.33a}$$

$$\Delta\dot{\underline{\widetilde{\Psi}}}(\tau)^T = \left[\ \underline{0}^T \quad \Delta\dot{\widetilde{\Psi}}(\tau)\right] \tag{4.33b}$$

$$\underline{A}_a = \underline{G}_a + \underline{H}_a \tag{4.33c}$$

$$\underline{G}_a = \delta_{ij} \quad j = i+1,\ i = 1\dots n \tag{4.33d}$$

$$\underline{H}_a{}^T = \left[\ \underline{0} \quad \dots \quad \underline{0} \quad -\underline{h}_a\right] \tag{4.33e}$$

$$\underline{h}_a{}^T = \left[\ h_0 \quad \dots \quad h_n\right] \tag{4.33f}$$

La matrice $\underline{G}_a$ de (4.33d) est de dimension $(n+1) \times (n+1)$. Sa ligne $n+1$ est nulle. La structure des matrices (4.33c)-(4.33e) permet de déduire que l'observateur proposé est de gain unité. Le système (4.33) est structurellement proche de ce qui a été proposé dans (Schwaller et al. 2013), et le conditionnement du système proposé en (4.4) peut y être avantageusement utilisé.

Dans (4.32b) et (4.33b), on considère que $\dot{\widetilde{\Psi}}\,[\,\underline{x}_a(\tau), \underline{U}_a(\tau)\,]$ est une fonction *Lipschitz* en $\underline{x}_a(\tau)$ et uniformément bornée en $\underline{U}_a(\tau)$ dans un ensemble invariant, avec une constante de *Lipschitz* L, c'est à dire :

$$\left\| \Delta\dot{\tilde{\Psi}}(\tau) \right\| \leqslant L \parallel \underline{x}_a(\tau) - \underline{\widetilde{x}}_a(\tau) \parallel \tag{4.34a}$$

$$\leqslant L \left\| \Delta\underline{y}_a(\tau) \right\| \tag{4.34b}$$

Appliquer l'inéquation de *Lipschitz* à (4.32) permet de réduire le nombre de variables utilisables de $\Delta\underline{y}_a(\tau)$, et de caractériser l'erreur perturbatrice $\Delta\dot{\tilde{\Psi}}(\tau)$. Pour bien des systèmes, si les fonctions $\dot{\tilde{\Psi}}(\tau)$ ne sont pas globalement *Lipschitz*, elles peuvent l'être localement, où peuvent être transformées de façon adéquate pour le devenir.

4.3.3 Analyse en stabilité des observations d'état

L'analyse en stabilité des observations consiste à prouver le caractère globalement ou localement asymptotique de la dynamique d'évolution de l'erreur d'état. En d'autres termes, indépendamment des conditions initiales du système et de l'observateur, savoir si les estimations de l'observateur vont converger vers l'état du système physique. Celà conduit aux deux théorèmes suivants :

Théorème 1. *Considérons un système MISO décomposé selon* (4.2)*, pour lequel la structure de l'observateur* (4.14) *est utilisée. Si la fonction* $\dot{\tilde{\Psi}}\,[\,\underline{x}_a(\tau), \underline{U}_a(\tau)\,]$ *du système est localement Lipschitz en* $\underline{x}_a(\tau)$ *et uniformément bornée en* $\underline{U}_a(\tau)$ *dans un ensemble invariant, avec une constante de Lipschitz* L (4.34)*, alors l'observateur* (4.14) *sera localement stable si les conditions suivantes sont satisfaites :*

$$\underline{A}_a{}^T\underline{P} + \underline{P}\,\underline{A}_a + (n+1)\,L^2\,\underline{P}\,\underline{P} + \frac{\underline{I}}{n+1} \leqslant \quad 0 \tag{4.35}$$

Si la fonction $\dot{\tilde{\Psi}}\,[\,\underline{x}_a(\tau), \underline{U}_a(\tau)\,]$ *du système est globalement Lipschitz et que* (4.35) *est satisfaite, alors l'observateur* (4.14) *sera globalement et asymptotiquement stable.*

Démonstration. Le théorème 1 peut se démontrer en prouvant la stabilité de (4.33a) à l'aide d'une fonction de *Liapounoff* définie positive de type quadratique :

$$V_n(\tau) = \Delta\underline{y}_a(\tau)^T\,\underline{P}\,\Delta\underline{y}_a(\tau) \tag{4.36a}$$

$$\underline{P} > 0 \tag{4.36b}$$

$\underline{P}$ est une matrice symétrique définie positive, de dimension $(n+1)\times(n+1)$, satisfaisant les critère de *Sylvester* de positivité. La démonstration de convergence est liée à l'étude du signe de la dérivée temporelle de $V_n(\tau)$. Ceci s'obtient après la dérivation temporelle de (4.36a), et l'insertion dans le résultat obtenu de (4.33a) à la place du terme $\Delta\dot{\underline{y}}_a(\tau)$:

$$\dot{V}_n(\tau) = \Delta\underline{y}_a(\tau)^T\,\underline{Q}\,\Delta\underline{y}_a(\tau) + N(\tau) \tag{4.37a}$$

$$\underline{Q} = \underline{A}_a{}^T\underline{P} + \underline{P}\,\underline{A}_a \tag{4.37b}$$

$$N(\tau) = \Delta\underline{y}_a(\tau)^T\,(\,2\,\underline{P}\,)\,\Delta\dot{\underline{\tilde{\Psi}}}(\tau) \tag{4.37c}$$

Dans (4.37a), $N(\tau)$ décrit l'influence des fonctions non linéaires sur les distances d'états ; $\underline{Q}$ est une matrice symétrique de dimension $(n+1)\times(n+1)$. La vérification du signe de $\dot{V}_n(\tau)$ incite à majorer $N(\tau)$ à l'aide des inéquations de *Schwartz* et *Lipschitz* (4.34b) pour obtenir une forme quadratique qu'il sera possible d'associer au premier membre de droite de (4.37a) :

$$N(\tau) \leqslant \left\| \Delta\underline{y}_a(\tau)^T\,(\,2\,\underline{P}\,)\,\Delta\dot{\underline{\tilde{\Psi}}}(\tau) \right\| \tag{4.38a}$$

$$\leqslant \left\| \Delta\underline{y}_a(\tau)^T\,(\,2\,\underline{P}\,) \right\| \left\| \Delta\dot{\underline{\tilde{\Psi}}}(\tau) \right\| \tag{4.38b}$$

$$\leqslant \left\| \Delta\underline{y}_a(\tau)^T\,(\,2\,\underline{P}\,) \right\| L \left\| \Delta\underline{y}_a(\tau) \right\| \tag{4.38c}$$

Pour déterminer le signe de $\dot{V}_n(\tau)$, on applique l'inéquation suivante :

$$\left\| \underline{a}(\tau)^T\underline{b}(\tau) \right\| \leqslant \frac{\sigma\,\underline{a}(\tau)^T\underline{a}(\tau)}{2} + \frac{\underline{b}(\tau)^T\underline{b}(\tau)}{2\,\sigma} \tag{4.39a}$$

$$\sigma = \frac{n+1}{2} \tag{4.39b}$$

à (4.38c) pour obtenir la majoration souhaitée de $N(\tau)$:

$$N(\tau) \leqslant \Delta\underline{y}_a(\tau)^T \, \underline{R} \, \Delta\underline{y}_a(\tau) \tag{4.40a}$$

$$\underline{R} = (n+1) \, L^2 \, \underline{P} \, \underline{P} + \frac{\underline{I}}{n+1} \tag{4.40b}$$

L'inéquation (4.40a) appliquée à (4.37a) permet de déduire :

$$\dot{V}_n(\tau) \leqslant \Delta\underline{y}_a(\tau)^T \, \underline{M} \, \Delta\underline{y}_a(\tau) \tag{4.41a}$$

$$\underline{M} = \underline{Q} + \underline{R} \tag{4.41b}$$

En imposant $\underline{Q} + \underline{R} \leqslant 0$ on obtient la condition (4.35), qui permet d'affirmer la semi-négativité de $\dot{V}_n(\tau)$. On en déduit la stabilité globalement asymptotique de (4.33a) si (4.34) est globalement *Lipschitz*. Dans (4.25), il est supposé que la fonction $\mathcal{L}^{-1}\{f_i(s)\} \to 0$ pour $t \to \infty$. De ce fait, on peut dire que $\underline{v}(\tau) \to \underline{x}(\tau)$ pour $t \to \infty$. L'observateur est localement stable si (4.34) est localement *Lipschitz*.

■

Les conditions de stabilité (4.35) sont moins restrictives que celles proposées en (Schwaller et al. 2013). En effet, la pulsation ω_0 n'y figure plus. La stabilité ne dépend plus que de la composition de la matrice $\underline{P}$, de la valeur des gains h_i obtenus par une méthode quelconque de synthèse des paramètres et de la valeur de la constante de *Lipschitz* L. Au vu de (4.32), on relève que ω_o impacte directement $\dot{\tilde{\Psi}}(\tau)$, et de ce fait également $\left\|\dot{\tilde{\Psi}}(\tau)\right\|$. Appliquer l'hypothèse de *Lipschitz* (4.34), implique que la constante L est fortement dépendante de ω_o, du type de non linéarité rencontrée, et de l'ordre n du système physique. La constante L influence au carré les conditions (4.35). En utilisant ω_o pour maintenir L petit, il est possible de rendre les conditions de stabilité indépendantes de l'échelle de temps du système et de définir arbitrairement les gains h_i.

Théorème 2. *Si le théorème 1 peut être appliqué au système* (4.2) *que l'on souhaite observer, alors* (4.14) *sera exponentiellement convergent :*

$$V_n(\tau) \leqslant V_n(0) \, \exp\left(\frac{\mu}{\nu}\,\tau\right) \quad \frac{\mu}{\nu} < 0 \tag{4.42}$$

Démonstration. Compte tenu du théorème 1, on a :

$$V_n(\tau) \leqslant \nu \left\| \, \Delta\underline{y}_a(\tau) \, \right\|^2 \quad \nu > \phi_{n+1} \tag{4.43}$$

De l'inégalité (4.41a) on peut déduire :

$$\frac{\dot{V}_n(\tau)}{V_n(\tau)} \leqslant \frac{\mu \left\| \, \Delta\underline{y}_a(\tau) \, \right\|^2}{V_n(\tau)} \tag{4.44a}$$

$$\max(r_{ii} + q_{ii}) < \mu < 0 \quad i = 1 \ldots n+1 \tag{4.44b}$$

La majoration de (4.44a) avec (4.43) donne :

$$\frac{\dot{V}_n(\tau)}{V_n(\tau)} \leqslant \frac{\mu \left\| \, \Delta\underline{y}_a(\tau) \, \right\|^2}{\nu \left\| \, \Delta\underline{y}_a(\tau) \, \right\|^2} \tag{4.45}$$

qui se réduit à :

$$\frac{d\, V_n(\tau)}{V_n(\tau)} \leqslant \frac{\mu}{\nu} \, d\tau \tag{4.46}$$

On déduit ensuite (4.42) par intégration, qui indique la convergence exponentielle.

■

4.3.4 Synthèse des paramètres à l'aide de critères algébriques

On désire choisir une composition de $\underline{P}$ et réaliser une synthèse des gains $\underline{h}_a$ de (4.33f) garantissant la convergence de l'observateur. Pour y parvenir, on considère le terme $\Delta\underline{\tilde{\Psi}}(\tau)$ de (4.33a) comme l'entrée perturbatrice d'un système linéaire à paramètres constants de fonction de transfert $1/D(s)$. La transformée de *Laplace* de (4.33a) s'écrit :

$$\frac{\Delta y_1(s)}{s\ \Delta\widetilde{\Psi}(s)} = \frac{1}{D(s)} \tag{4.47a}$$

$$D(s) = s^{n+1} + h_n\ s^n + \ldots + h_1\ s + h_0 \tag{4.47b}$$

Une solution simple consiste à utiliser un polynôme à pôles multiples :

$$D(s) = (s+\nu)^{n+1} \tag{4.48a}$$

$$= \sum_{i=0}^{n+1} h_k\ s^k \quad k = n+1-i \tag{4.48b}$$

$$h_k = \nu^i\ \frac{(n+1)\,!}{i\,!\,k\,!} \tag{4.48c}$$

La valeur du pôle multiple $0 < \nu \leqslant 1$ autorise une certaine pondération des coefficients binomiaux (4.48c). Toutefois, c'est le choix de la pulsation ω_o qui reste prédominant pour déterminer la vitesse de convergence de l'observateur. Le nombre et la valeur des paramètres sont entièrement déterminés par l'ordre n. La matrice $\underline{A}_a$ (4.33c) est alors entièrement paramétrée.
Fixons à présent la matrice $\underline{P}$ à l'aide de :

$$\underline{P} = \begin{bmatrix} \lambda_1 & 0 & \ldots & \phi_1 \\ 0 & \ddots & \ldots & \vdots \\ \vdots & & \lambda_n & \phi_n \\ \phi_1 & \ldots & \phi_n & \phi_{n+1} \end{bmatrix} \tag{4.49a}$$

$$\phi_{n+1} = 1 + \frac{n-1}{3} \tag{4.49b}$$

$$\phi_i = \phi_{n+1}\ 2^{(i-n-1)} \quad i = 1 \ldots n+1 \tag{4.49c}$$

$$\lambda_i = \phi_{n+1}\ \phi_i, \quad i = 1 \ldots n \tag{4.49d}$$

Les paramètres ϕ_i de la dernière ligne et dernière colonne sont obtenus à l'aide d'une suite géométrique de raison 2, avec pour terme de base ϕ_{n+1} (4.49b). Les paramètres λ_i sont obtenus par homothétie des paramètres ϕ_i. Si l'on utilise les matrices $\underline{A}_a$ et $\underline{P}$ ainsi paramétrées dans le critère de stabilité (4.35), on peut rechercher la constante L_{max} limite pour $1 \leqslant n \leqslant 4$:

Ordre n	1	2	3	4
L_{max}	0,95	0,55	0,4	0,35

TABLE 4.1: Constantes de *Lipschitz* limites pour $\nu = 1$

Dans (4.35), la constante de *Lipschitz* L (4.34) est finalement le seul paramètre laissé libre. On contrôle sa valeur en fixant la constante ω_o qui permet le passage en temps non dimensionnel (4.4), (4.5). On veille ainsi à vérifier l'inéquation :

$$\left\| \Delta\dot{\tilde{\Psi}}(\tau) \right\| \ / \ \left\| \Delta\underline{y}_a(\tau) \right\| \leqslant L_{max} \tag{4.50}$$

pour garantir la stabilité de l'observation. Le choix d'utilisation des polynômes multiples permet en plus de garantir une réponse apériodique limite de la correction de l'erreur d'observation.

4.3.5 Détermination des paramètres de stabilité à l'aide des inégalités matricielles affines

Modifions à présent notre approche pour prouver la stabilité de (4.33). On fixe une pulsations ω_o pour l'observateur (4.14) pour contrôler la vitesse de convergence. On utilise les polynômes multiples (4.48) pour définir

les gains h_i. On cherche à présent une matrice $\underline{P}$ pleine permettant de respecter la positivité de la fonction de *Liapounoff* (4.36), et la semi-négativé de (4.37). On cherche enfin à maximiser la constante de *Lipschitz* de manière à déterminer la valeur L_{max} admissible. À partir de (4.41) on peut écrire :

$$\Delta\underline{y}_a(\tau)^T\ \underline{M}\ \Delta\underline{y}_a(\tau) \leqslant -2\ L\ \Delta\underline{y}_a(\tau)^T\ \underline{P}\ \Delta\underline{y}_a(\tau) \leqslant 0 \tag{4.51}$$

On peut ainsi majorer (4.41) avec :

$$\dot{V}_n(\tau) \leqslant \Delta\underline{y}_a(\tau)^T\ \underline{S}\ \Delta\underline{y}_a(\tau) \leqslant 0 \tag{4.52a}$$

$$\underline{S} = \underline{S}_{11} - \underline{S}_{12}{}^T\ \underline{S}_{22}{}^{-1}\ \underline{S}_{21} \tag{4.52b}$$

$$\underline{S}_{11} = \underline{S}_{11}{}^T = \underline{A}_a{}^T\underline{P} + \underline{P}\ \underline{A}_a \tag{4.52c}$$

$$\underline{S}_{12}{}^T = \underline{S}_{21} = \sqrt{n+1}\ L\ \underline{P} + \frac{\underline{I}}{\sqrt{n+1}} \tag{4.52d}$$

$$\underline{S}_{22}{}^{-1} = \underline{S}_{22} = -\underline{I} \tag{4.52e}$$

où $\underline{S}$ est le complément de *Schur* d'une matrice partionnée, dont (4.52c)-(4.52e) sont les constituants. Il est clair que si $\underline{S} \leqslant 0$, (4.33) sera au moins localement stable. On en déduit la matrice partionnée :

$$\underline{U} \leqslant 0 \tag{4.53a}$$

$$\underline{U} = \begin{bmatrix} \underline{S}_{11} & \underline{S}_{12} \\ \underline{S}_{21}{}^T & \underline{S}_{22} \end{bmatrix} \tag{4.53b}$$

Comme L est également une variable d'optimisation, il subsiste un terme bilinéaire $L\ \underline{P}$ dans (4.52d). Commençons par utiliser une transformation linéaire de façon à se ramener à une inégalité affine :

$$\underline{T}^T\ \underline{U}\ \underline{T} \leqslant 0 \tag{4.54a}$$

$$\underline{T} = \begin{bmatrix} \underline{P}^{-1} & \underline{0} \\ \underline{0} & \underline{I} \end{bmatrix} \tag{4.54b}$$

$$\underline{Y} = \underline{T}^T\ \underline{U}\ \underline{T} \tag{4.54c}$$

$$= \begin{bmatrix} \underline{P}^{-1}\ \underline{A}_a{}^T + \underline{A}_a\ \underline{P}^{-1} & \sqrt{n+1}\ L\ \underline{I} + \dfrac{\underline{P}^{-1}}{\sqrt{n+1}} \\ \sqrt{n+1}\ L\ \underline{I} + \dfrac{\underline{P}^{-1}}{\sqrt{n+1}} & -\underline{I} \end{bmatrix} \tag{4.54d}$$

On effectue un dernier changement de variable, de façon à se ramener à l'inégalité matricielle affine finale :

$$\underline{X} = \underline{P}^{-1} \tag{4.55a}$$

$$\underline{Y} = \begin{bmatrix} \underline{X}\ \underline{A}_a{}^T + \underline{A}_a\ \underline{X} & \sqrt{n+1}\ L\ \underline{I} + \dfrac{\underline{X}}{\sqrt{n+1}} \\ \sqrt{n+1}\ L\ \underline{I} + \dfrac{\underline{X}}{\sqrt{n+1}} & -\underline{I} \end{bmatrix} \tag{4.55b}$$

$$\underline{X} > 0, \quad L > 0 \quad \underline{Y} \leqslant 0 \tag{4.55c}$$

qui fournit des conditions suffisantes pour qu'un solveur puisse déterminer une matrice $\underline{P}$ symétrique définie positive. On satisfait alors (4.36) et (4.52a), et la stabilité de (4.33) est garantie. La maximisation de la constante de *Lipschitz* L_{max} permet de fixer l'inéquation (4.50), et de préciser les conditions locales de stabilité.

4.4 Observation du système originel *via* les fonctions de transformations inverses

4.4.1 Définition des observateurs du système originel

Une faiblesse bien connue des observateurs à grands gains est leur sensibilité aux bruits. L'observateur proposé en (4.14), du fait de sa structure, permet de contourner cet obstacle et de proposer des estimations d'état $\underline{v}(\tau)$ robustes décomposées sous forme canonique de régulation. En (4.6b), les fonctions de transformations inverses

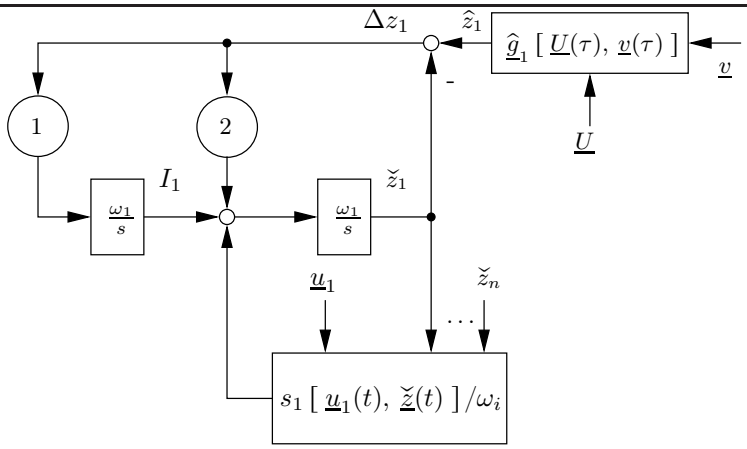


FIGURE 4.2: Observateur de la fonction inverse $\widehat{z}_1(\tau_i)$

$\underline{g}(\tau)$ autorisent la transformation de cette forme canonique à la représentation d'état (4.1) initiale. Cette façon de procéder ne permet malheureusement pas toujours d'obtenir des estimations d'état $\underline{z}(\tau)$ robustes. En effet, le bruit subsistant sur $\underline{v}(\tau)$ peut être considérablement amplifié selon la nature des fonctions $\underline{g}(\tau)$.
On se propose (Schwaller et al. 2017) de résoudre ce problème en jouant sur les propriétés de robustesse au bruit des observateurs (Bodizs et al. 2011). L'utilisation du vecteur estimé $\underline{v}(\tau)$ de l'observateur permet de définir :

$$\widehat{z}_i(t) = \widehat{g}_i\,[\,\underline{v}(\tau),\,\underline{U}(\tau)\,] \quad i = 1\ldots n \tag{4.56a}$$

$$\underline{\widehat{z}}(t)^T = \begin{bmatrix}\widehat{z}_1(t) & \ldots & \widehat{z}_n(t)\end{bmatrix} \tag{4.56b}$$

On obtient ainsi l'estimée $\underline{\widehat{z}}(t)$ de $\underline{z}(t)$ (4.1). Si les conditions (4.35) sont respectées, $\underline{\widehat{z}}(t) \to \underline{z}(t)$ quand $\Delta y_1(t) \to 0$. De même, $\dot{\underline{\widehat{z}}}(t) \to \dot{\underline{z}}(t)$ quand $\Delta y_1(t) \to 0$. On a alors :

$$\lim_{\Delta y_1(t) \to 0} \dot{\underline{\widehat{z}}}(t) = \dot{\underline{z}}(t) \tag{4.57a}$$

$$\dot{\underline{\widehat{z}}}(t)^T = \begin{bmatrix}\widehat{s}_1(t) & \ldots & \widehat{s}_n(t)\end{bmatrix} \tag{4.57b}$$

$$\widehat{s}_i(\tau) = s_i\,[\,\underline{\widehat{z}}(t),\,\underline{u}_1(t)\,] \tag{4.57c}$$

Les n estimées $\widehat{z}_i(t)$ peuvent servir d'entrées de références pour observer les n variables d'états $\breve{z}_i(t)$ qui tendent vers (4.56a). Leurs dérivées temporelles tendent vers $\dot{\underline{\widehat{z}}}(t)$, qui elles-mêmes tendent vers $\dot{\underline{z}}(t)$ (4.57). Définissons à présent n observateurs d'ordre un du même type que (4.14). Chacun est normé par une pulsation ω_i, possède sa propre constante de *Lipschitz*, et ses propres conditions de stabilité qu'il faut déterminer. La synthèse des gains h_i (4.33f) selon (4.48b) donne $\underline{h_a}^T = \begin{bmatrix}1 & 2\end{bmatrix}$ pour chacun d'eux. Les n observateurs s'écrivent :

$$\dot{\breve{z}}_i(\tau_i) = I_i(\tau_i) + 2\,\Delta z_i(\tau_i) + \breve{s}_i(\tau_i) \tag{4.58a}$$

$$\dot{I}_i(\tau_i) = \Delta z_i(\tau_i) \quad i = 1\ldots n \tag{4.58b}$$

$$\Delta z_i(\tau_i) = \widehat{z}_i(\tau_i) - \breve{z}_i(\tau_i) \tag{4.58c}$$

$$\breve{s}_i(\tau_i) = s_i\,[\,\underline{\breve{z}}(t),\,\underline{u}_1(t)\,]\,/\,\omega_i \tag{4.58d}$$

$$\tau_i = \omega_i\,t \tag{4.58e}$$

avec $\underline{\breve{z}}(t) = \begin{bmatrix}\breve{z}_1(t) & \ldots & \breve{z}_n(t)\end{bmatrix}$ le vecteur des estimations de $\underline{\widehat{z}}(\tau)$ utilisé en (4.58d) ; $\breve{s}_i(\tau)$ est la fonction non linéaire normée de $\dot{\breve{z}}_i(\tau)$. La FIGURE 4.2 page 36 illustre le diagramme fonctionnel de (4.56a) et (4.58) pour $\widehat{z}_1(\tau_i)$.

La procédure de calcul standard est la suivante :
- estimation de $\underline{v}(\tau)$ (4.14i) après traitement de (4.14) ;
- estimation de $\underline{\widehat{z}}(t)$ (4.56a) ;
- estimation des n distances d'états (4.58c) ;
- détermination des n fonctions non linéaires $\breve{s}_i(\tau_i)$ (4.58d) pour accéder aux n termes $\dot{\breve{z}}_i(\tau_i)$ (4.58a) et $\dot{I}_i(\tau_i)$ (4.58b) ;
- intégration des n équations (4.58a) pour obtenir $\underline{\breve{z}}(t)$.

La dérivation temporelle en temps non dimensionnel de (4.58c) en utilisant (4.57) permet d'obtenir pour chaque observateur l'équivalent de (4.27) :

$$\Delta\dot{z}_i(\tau_i) = \dot{\widehat{z}}_i(\tau_i) - \dot{\breve{z}}_i(\tau_i) \quad i = 1 \dots n \tag{4.59a}$$

$$= \Delta\widetilde{\Psi}_i(\tau_i) - I_i(\tau_i) - 2\ \Delta z_i(\tau_i) \tag{4.59b}$$

$$\Delta\widetilde{\Psi}_i(\tau_i) = \widehat{s}_i(\tau_i) - \breve{s}_i(\tau_i) \tag{4.59c}$$

$$\widehat{s}_i(\tau_i) = s_i\ [\ \underline{\widehat{z}}(t),\ \underline{u}_1(t)\ ]\ /\ \omega_i \tag{4.59d}$$

4.4.2 Dynamique des erreurs d'observations

On cherche à présent à caractériser la dynamique des erreurs d'observations en cherchant les n équations différentielles des distances $\Delta z_i(\tau_i)$ Du fait de la présence des intégrateurs $\dot{I}_i(\tau_i)$, une dérivation temporelle supplémentaires est nécessaire pour obtenir l'équation différentielle des distances $\Delta z_i(\tau_i)$. Pour y parvenir, il est nécessaire de définir les vecteurs augmentés suivants :

$$\underline{\Upsilon}(\tau_i) = \left[\ \underline{u}_1(\tau_i) \quad \underline{u}_2(\tau_i)\right] \tag{4.60a}$$

$$\underline{\widehat{z}}_i(\tau_i)^T = \left[\ \widehat{z}_i(\tau_i) \quad \dot{\widehat{z}}_i(\tau_i)\right] \quad i = 1 \dots n \tag{4.60b}$$

$$\underline{\breve{z}}_i(\tau_i)^T = \left[\ \breve{z}_i(\tau_i) \quad \dot{\breve{z}}_i(\tau_i)\right] \tag{4.60c}$$

$$\Delta\underline{z}_i(\tau_i)^T = \left[\ \Delta z_i(\tau_i) \quad \Delta\dot{z}_i(\tau_i)\right] \tag{4.60d}$$

$$\underline{\widehat{Z}}(\tau_i)^T = \left[\ \underline{\widehat{z}}_1(\tau_i) \quad \dots \quad \underline{\widehat{z}}_n(\tau_i)\right] \tag{4.60e}$$

$$\underline{\breve{Z}}(\tau_i)^T = \left[\ \underline{\breve{z}}_1(\tau_i) \quad \dots \quad \underline{\breve{z}}_n(\tau_i)\right] \tag{4.60f}$$

La dérivée temporelle de (4.59b) s'écrit :

$$\Delta\ddot{\breve{z}}_i(\tau_i) = \Delta\dot{\widetilde{\Psi}}_i(\tau_i) - \Delta z_i(\tau_i) - 2\ \Delta\dot{\breve{z}}_i(\tau_i) \quad i = 1 \dots n \tag{4.61a}$$

$$\Delta\dot{\widetilde{\Psi}}_i(\tau_i) = \dot{s}_i\left[\ \underline{\widehat{Z}}(\tau_i), \underline{\Upsilon}(\tau_i)\ \right] - \dot{s}_i\left[\ \underline{\breve{Z}}(\tau_i), \underline{\Upsilon}(\tau_i)\ \right] \tag{4.61b}$$

et donne l'expression scalaire des équations différentielles des erreurs d'observations. L'utilisation des notations (4.60) permet l'écriture matricielle de (4.61a) sous forme d'équations d'états :

$$\Delta\dot{\underline{z}}_i(\tau_i) = \underline{A}_i\ \Delta\underline{z}_i(\tau_i) + \Delta\dot{\underline{\widetilde{\Psi}}}_i(\tau_i) \quad i = 1 \dots n \tag{4.62a}$$

$$\underline{A}_i = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \tag{4.62b}$$

$$\Delta\dot{\underline{\widetilde{\Psi}}}_i(\tau_i) = \begin{bmatrix} 0 \\ \Delta\dot{\widetilde{\Psi}}_i(\tau_i) \end{bmatrix} \tag{4.62c}$$

En supposant que les fonctions non linéaires $\dot{s}_i$ sont au moins localement *Lipschitz* en $\widehat{\underline{z}}_i(\tau_i)$, et uniformément bornées en $\underline{\Upsilon}(\tau_i)$ dans un ensemble invariant, avec une constante de *Lipschitz* L_i, c'est à dire :

$$\left\| \Delta\dot{\widetilde{\Psi}}_i(\tau_i) \right\| \leqslant L_i\ \|\ \Delta\underline{z}_i(\tau_i)\ \|_F \quad i = 0 \dots n \tag{4.63}$$

Appliquer L'inégalité de *Lipschitz* à (4.62b) permet de réduire le nombre de variables utilisables de $\underline{\widehat{Z}}(\tau_i)$, $\underline{\widetilde{Z}}(\tau_i)$, et de caractériser l'erreur perturbatrice $\Delta\dot{\widetilde{\Psi}}_i(\tau_i)$. Pour beaucoup de systèmes, si les fonctions $\dot{s}_i$ ne sont pas globalement de type *Lipschitz*, elles peuvent l'être localement, où peuvent être transformées de façon adéquate pour le devenir.

4.4.3 Convergence des observations d'état

Cherchons à présent à analyser l'évolution globalement asymptotique des erreurs d'observation, et à caractériser les conditions limites de stabilités de chaque observateur (4.58).

Théorème 3. *Considérons un système MISO décomposé selon* (4.3)*, pour lequel les structures observatrices* (4.14) *et* (4.58) *sont utilisées, et liés entre eux par les fonctions de transformations inverses* (4.56a)*. Si les*

fonctions $\dot{s}_i \left[\underline{\widehat{Z}}(\tau_i), \underline{\Upsilon}(\tau_i) \right]$ *sont localement de type Lipschitz en* $\underline{\widehat{z}}_i(\tau_i)$ *et uniformément bornées en* $\underline{\Upsilon}(\tau_i)$ *dans un ensemble invariant, avec des constantes de Lipschitz* L_i (4.63)*, alors les observateurs* (4.58) *seront localement stables si les constantes* L_i *satisfont aux conditions suivantes :*

$$\underline{A}_i{}^T \underline{P}_i + \underline{P}_i \underline{A}_i + 2\ L_i{}^2\ \underline{P}_i\ \underline{P}_i + \frac{\underline{I}}{2} \leqslant \quad 0 \tag{4.64}$$

Si les fonctions $\dot{s}_i \left[\underline{\widehat{Z}}(\tau_i), \underline{\Upsilon}(\tau_i) \right]$ *sont globalement de type Lipschitz, et si les constantes* L_i *satisfont* (4.63)*, alors les observateurs* (4.58) *seront globalement et asymptotiquement stables.*

Démonstration. On démontre le théorème 3 en utilisant une fonction de *Liapounoff* adéquate, définie positive, telle la fonction quadratique suivante :

$$\breve{V}_n(\tau_i) = \sum_{i=1}^{n} v_i(\tau_i) \tag{4.65a}$$

$$v_i(\tau_i) = \Delta \underline{z}_i(\tau_i)^T\ \underline{P}_i\ \Delta \underline{z}_i(\tau_i) \tag{4.65b}$$

$$\underline{P}_i = \begin{bmatrix} \lambda_{i1} & \phi_{i1} \\ \phi_{i1} & \phi_{i2} \end{bmatrix} \quad \text{avec} \quad \underline{P}_i > 0 \tag{4.65c}$$

Les matrices $\underline{P}_i$ sont symétriques et définies positives, et du même type que celles définie en (4.49). La démonstration de convergence est liée à l'étude du signe de la dérivée de la fonction $\dot{\breve{V}}_n(\tau_i)$. Ceci s'obtient après une dérivation temporelle de (4.65a), et en insérant (4.62a) dans le résultat obtenu pour les termes $\Delta \dot{\underline{z}}_i(\tau_i)$:

$$\dot{\breve{V}}_n(\tau_i) = \sum_{i=1}^{n} \dot{v}_i(\tau_i) \tag{4.66a}$$

$$\dot{v}_i(\tau_i) = \Delta \underline{z}_i(\tau_i)^T\ \underline{Q}_i\ \Delta \underline{z}_i(\tau_i) + N_i(\tau_i) \quad i = 0 \ldots n \tag{4.66b}$$

$$\underline{Q}_i = \underline{A}_i{}^T \underline{P}_i + \underline{P}_i\ \underline{A}_i \tag{4.66c}$$

$$N_i(\tau_i) = \Delta \underline{z}_i(\tau_i)^T\ (\ 2\ \underline{P}_i\ )\ \Delta \dot{\underline{\Psi}}_i(\tau_i) \tag{4.66d}$$

Vérifier le signe du second membre implique la majoration de $N_i(\tau_i)$, en utilisant les inégalités de *Schwartz* et de *Lipschitz* (4.63) :

$$N_i(\tau_i) \leqslant \left\| \Delta \underline{z}_i(\tau_i)^T\ (\ 2\ \underline{P}_i\ )\ \Delta \dot{\underline{\Psi}}(\tau_i) \right\| \tag{4.67a}$$

$$\leqslant \left\| \Delta \underline{z}_i(\tau_i)^T\ (\ 2\ \underline{P}_i\ ) \right\| \left\| \Delta \dot{\underline{\Psi}}(\tau_i) \right\| \tag{4.67b}$$

$$\leqslant \left\| \Delta \underline{z}_i(\tau_i)^T\ (\ 2\ \underline{P}_i\ ) \right\|\ L_i\ \| \Delta \underline{z}_i(\tau_i) \| \tag{4.67c}$$

Pour déterminer le signe de $\dot{v}_i(\tau_i)$, on applique l'inégalité :

$$\left\| \underline{a}(\tau_i)^T \underline{b}(\tau_i) \right\| \leqslant \frac{1}{2}\ \underline{a}(\tau_i)^T \underline{a}(\tau_i) + \frac{1}{2}\ \underline{b}(\tau_i)^T \underline{b}(\tau_i) \tag{4.68a}$$

$$\underline{a}(\tau_i) = 2\ L_i\ \underline{P}_i{}^T\ \Delta \underline{z}_i(\tau_i) \tag{4.68b}$$

$$\underline{b}(\tau_i) = \Delta \underline{z}_i(\tau_i) \tag{4.68c}$$

à (4.67c) pour obtenir la majoration souhaitée de $N_i(\tau_i)$:

$$N_i(\tau_i) \leqslant \Delta \underline{z}_i(\tau_i)^T\ \underline{R}_i\ \Delta \underline{z}_i(\tau_i) \quad i = 1 \ldots n \tag{4.69a}$$

$$\underline{R}_i = 2\ L_i{}^2\ \underline{P}_i\ \underline{P}_i + \frac{1}{2}\ \underline{I} \tag{4.69b}$$

À l'aide de l'inégalité (4.69a), on déduit de (4.66b) :

$$\dot{v}_i(\tau_i) \leqslant \Delta \underline{z}_i(\tau_i)^T\ \underline{M}_i\ \Delta \underline{z}_i(\tau_i) \tag{4.70a}$$

$$\underline{M}_i = \underline{Q}_i + \underline{R}_i \tag{4.70b}$$

En imposant $\underline{Q}_i + \underline{R}_i \leqslant 0$, on obtient la condition (4.64), qui permet d'affirmer la semi-négativité de $\dot{v}_i(\tau_i)$. La fonction $\breve{V}_n(\tau_i)$ (4.65a) étant la somme des fonctions $\dot{v}_i(\tau_i)$ semi-négatives, $\dot{\breve{V}}_n(\tau_i)$ sera elle-même semi-négative : (4.62a) est globalement et asymptotiquement stable ; (4.62a) est localement stable si (4.63) est localement *Lipschitz* ■

Si l'on utilise le théorème 2, il est aisé de démontrer que les observateurs (4.58) sont exponentiellement convergents.

4.5 Simulations

Deux exemples viennent illustrer la section § 4.3. Le premier, § 4.5.1 porte sur l'observation d'un oscillateur chaotique de (Lorenz 1963) d'ordre 3, modèle de convection de masses d'airs, utilisé en météorologie. Son principal intérêt est de souligner la capacité du nouvel observateur à fournir un vecteur $\underline{v}(\tau)$ sous formes de dérivées successives robustes au bruit.
Le deuxième, § 4.5.2 page 43 porte sur l'observation d'un système de gestion des boues activées (Nagy-Kiss et al. 2010), utilisé pour le traitement des eaux usées d'une station d'épuration. Il s'agit dans ce cas de décrire un système multi-entrées comportant énormément de paramètres, avec des variables d'états possédant des échelles de temps très différentes. Ce deuxième exemple est une bonne illustration de l'intérêt que l'on peut trouver à l'observation d'un système non transformé (4.1) *via* les fonctions inverses.

4.5.1 L'attracteur étrange de *Lorenz*

Le système de *Lorenz* est décrit par la représentation d'état :

$$\dot{z}_x(t) = \sigma\ z_y(t) - \sigma\ z_x(t) \tag{4.71a}$$
$$\dot{z}_y(t) = -\ z_y(t) + \rho\ z_x(t) - z_x(t)\ z_z(t) \tag{4.71b}$$
$$\dot{z}_z(t) = -\ \beta\ z_z(t) + z_x(t)\ z_y(t) \tag{4.71c}$$
$$y(t) = z_x(t) \tag{4.71d}$$
$$\underline{z}(t) = \left[z_1(t),\ z_2(t), z_3(t)\right] \tag{4.71e}$$

pour lequel on choisit les conditions initiales suivantes :

$$\beta = 1, \quad \sigma = 4, \quad \rho = 20 \tag{4.72a}$$
$$z_x(0) = z_y(0) = z_z(0) = 10 \tag{4.72b}$$

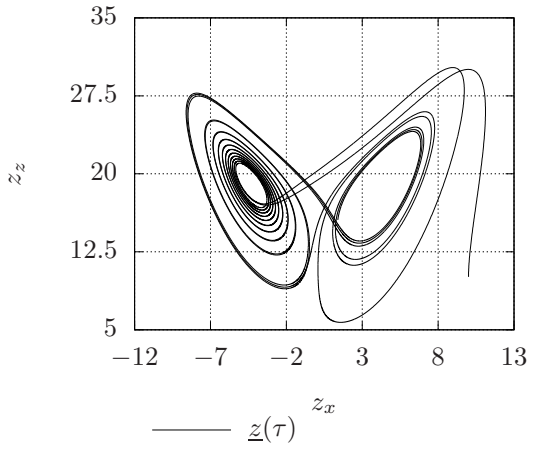
Sa représentation d'état sous forme canonique de régulation est la suivante :

$$\dot{x}_3(t) = \Psi\left[\underline{x}(t)\right] \tag{4.73a}$$
$$\begin{aligned}\Psi\left[\underline{x}(t)\right] = &-(1+\sigma+\beta)\ x_3(t) - \beta\ (1+\sigma)\ x_2(t) + \beta\ \sigma\ (\rho-1)\ x_1(t) \\ &- x_1(t)^2 x_2(t) - \sigma\ x_1(t)^3 + \frac{x_2(t)\ x_3(t) + (1+\sigma)\ x_2(t)^2}{x_1(t)}\end{aligned} \tag{4.73b}$$
$$\dot{x}_2(t) = x_3(t) \tag{4.73c}$$
$$\dot{x}_1(t) = x_2(t) \tag{4.73d}$$
$$y(t) = x_1(t), \quad n = 3 \tag{4.73e}$$

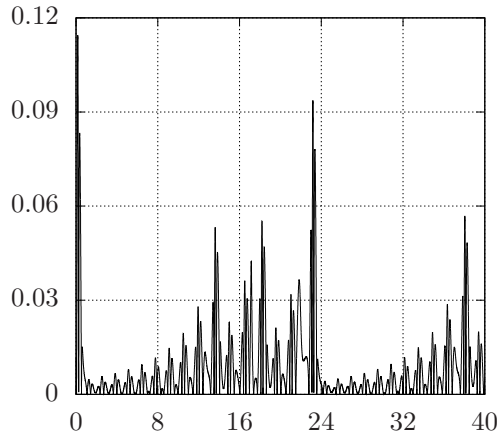
avec (4.73b) la fonction non linéaire Ψ du système. Cette dernière peut être ou pas dépendante de $\underline{U}$. La seule contrainte à remplir est l'existence d'une distance d'état $\Delta\dot{\tilde{\Psi}}$ (4.34) de type *Lipschitz*. Le système de transformation permettant de lier (4.73) à (4.71) est donné par :

$$z_x(t) = x_1(t) \tag{4.74a}$$
$$z_y(t) = x_1(t) + \left(\frac{1}{\sigma}\right) x_2(t) \tag{4.74b}$$
$$z_z(t) = \rho - 1 - \left(1 + \frac{1}{\sigma}\right)\frac{x_2(t)}{x_1(t)} - \left(\frac{1}{\sigma}\right)\frac{x_3(t)}{x_1(t)} \tag{4.74c}$$

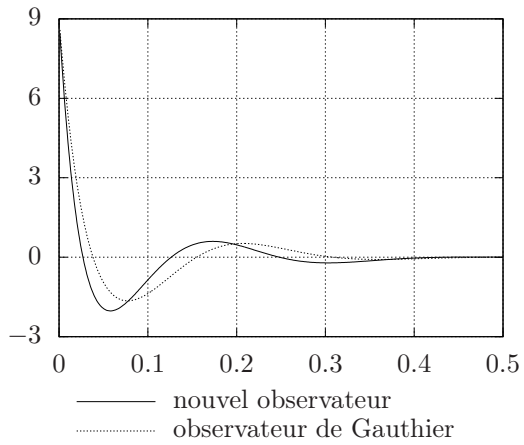
La FIGURE 4.3(a) illustre le comportement bien connu de l'attracteur dans le plan $z_x\ z_z$. Paramétré de la sorte, il possède une fréquence propre de l'ordre de 1 Hz. Il est intéressant de noter que $x_1(t)$ passe par 0 durant de très brèves transitions, ce qui rend possible l'observation avec la fonction $\tilde{\Psi}$ (4.73b).



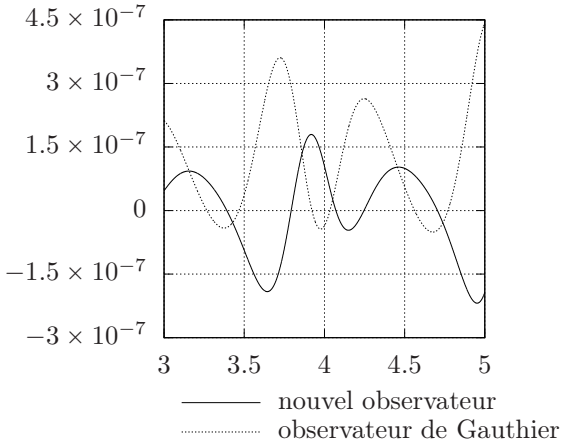
(a) trajectoires d'état



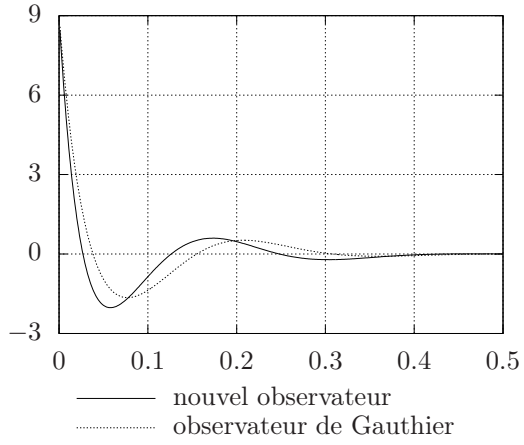
(b) $|\dot{\tilde{\Psi}}(\tau)|/||\underline{x}_a(\tau)||$



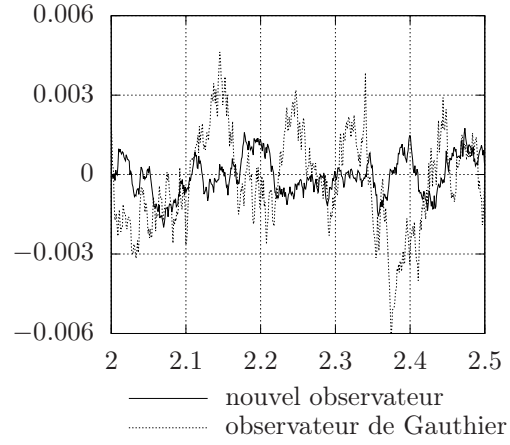
(c) $\Delta y(t)$, convergence asymptotique



(d) $\Delta y(t)$, convergence stabilisée

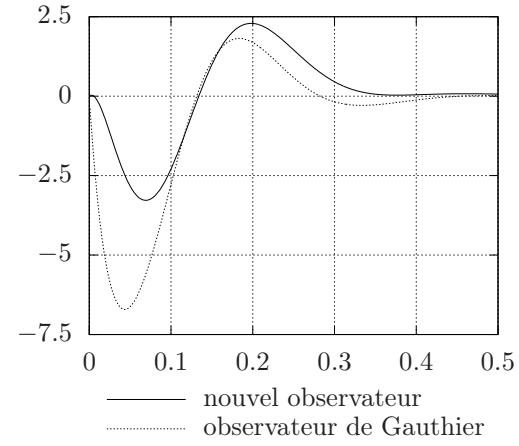


(e) $\Delta y(\tau)$, convergence asymptotique

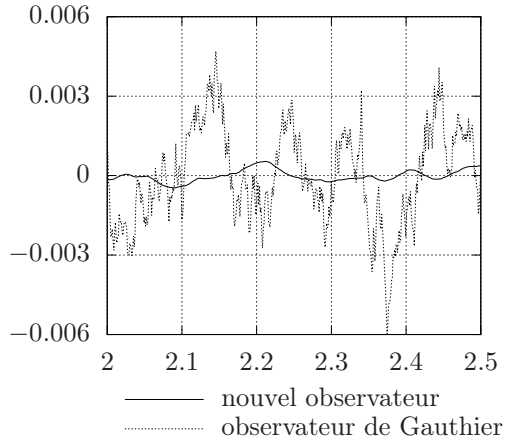


(f) $\Delta y(\tau)$, convergence stabilisée

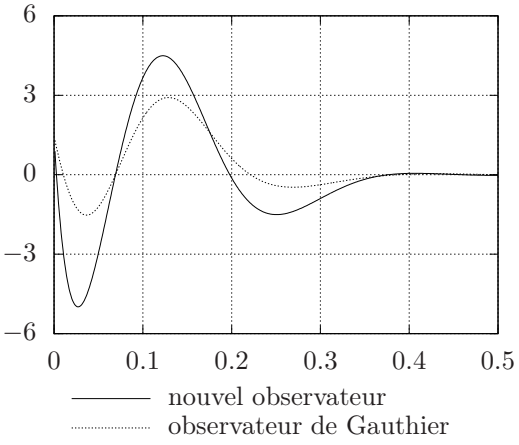
FIGURE 4.3: Observation d'un attracteur de *Lorenz, partie 1*



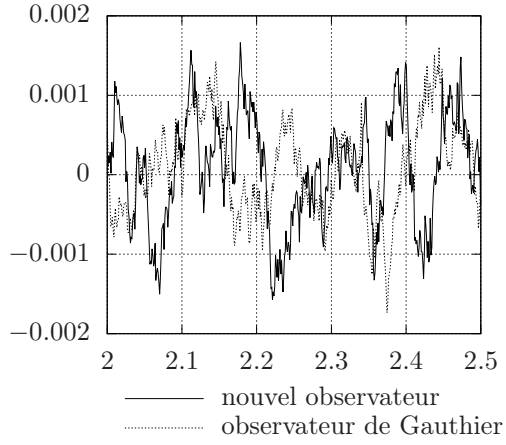
(a) $\Delta\widehat{x}_2(\tau)$, $\Delta\check{x}_2(\tau)$, convergence asymptotique



(b) $\Delta\widehat{x}_2(\tau)$, $\Delta\check{x}_2(\tau)$, convergence stabilisée



(c) $\Delta\widehat{x}_3(\tau)$, $\Delta\check{x}_3(\tau)$, convergence asymptotique



(d) $\Delta\widehat{x}_3(\tau)$, $\Delta\check{x}_3(\tau)$, convergence stabilisée

FIGURE 4.4: Observation d'un attracteur de *Lorenz, partie 2*

La représentation normée de (4.73) s'écrit :

$$\dot{x}_3(\tau) = \widetilde{\Psi}\left[\underline{x}(\tau)\right] \tag{4.75a}$$

$$\widetilde{\Psi}\left[\underline{x}(\tau)\right] = -\frac{1+\sigma+\beta}{\omega_o}\, x_3(\tau) - \frac{\beta\,(1+\sigma)}{{\omega_o}^2}\, x_2(\tau) + \frac{\beta\,\sigma\,(\rho-1)}{{\omega_o}^3}\, x_1(\tau) - \frac{x_1(\tau)^2\, x_2(\tau)}{{\omega_o}^2} - \frac{\sigma\, x_1(\tau)^3}{{\omega_o}^3} + \frac{x_2(\tau)\, x_3(\tau) + \dfrac{1+\sigma}{\omega_o}\, x_2(\tau)^2}{x_1(\tau)} \tag{4.75b}$$

$$\dot{x}_2(\tau) = x_3(\tau) \tag{4.75c}$$

$$\dot{x}_1(\tau) = x_2(\tau) \tag{4.75d}$$

$$y(\tau) = x_1(\tau), \quad f_o = 1/T_o = 2,5\ Hz \tag{4.75e}$$

Il est évident que les observateurs proposés précédemment (Schwaller et al. 2008), (Schwaller et al. 2013) ne peuvent pas être utilisés dans ce cadre, du fait de la fonction $\widetilde{\Psi}$ qui ne peut être décomposée comme suit :

$$\widetilde{\Psi}\left[\underline{x}(\tau)\right] = f_1\left(x_1(\tau), x_2(\tau), \underline{u}(\tau)\right) + \dot{f}_2\left(x_1(\tau), x_2(\tau)\right)$$

Le système de transformation normé liant (4.75) à l'espace d'état (4.71) s'écrit :

$$z_x(\tau) = x_1(\tau) \tag{4.76a}$$

$$z_y(\tau) = x_1(\tau) + \left(\frac{\omega_o}{\sigma}\right) x_2(\tau) \tag{4.76b}$$

$$z_z(\tau) = \rho - 1 - \left(\omega_o + \frac{\omega_o}{\sigma}\right) \frac{x_2(\tau)}{x_1(\tau)} - \left(\frac{{\omega_o}^2}{\sigma}\right) \frac{x_3(\tau)}{x_1(\tau)} \tag{4.76c}$$

En isolant $x_2(\tau)$ et $x_3(\tau)$ respectivement dans (4.76b) et (4.76c), il est possible de reconstruire $\underline{x}_a(\tau)$ connaissant $z_x(\tau)$, $z_x(\tau)$, $z_z(\tau)$ (4.74). En calculant la dérivée numérique de $\widetilde{\Psi}(\tau)$ (4.75b), on peut ensuite estimer une constante de *Lipschitz* maximisée, dans notre cas $L = 0,11$ (FIGURE 4.3(b)) sur les 40 s du test (abscisses en secondes pour les FIGURE 4.3 et 4.4). Or ce rapport est bien en deçà de la valeur $L_{max} = 0,4$ (tableau 4.1 page 34) obtenue pour des systèmes d'ordre 3. La pulsation normative ω_o est bien choisie, l'observateur (4.14) sera stable. Ce dernier, illustré par la FIGURE 4.1 page 29 s'écrit comme suit sous forme scalaire :

$$\dot{\widehat{x}}_1(\tau) = \widehat{x}_2(\tau) + h_3\ \Delta y_1(\tau) \tag{4.77a}$$

$$\dot{\widehat{x}}_2(\tau) = \breve{x}_3(\tau) + h_2\ \Delta y_1(\tau) \tag{4.77b}$$

$$\dot{\breve{x}}_2(\tau) = \breve{x}_3(\tau) \tag{4.77c}$$

$$\dot{\breve{x}}_3(\tau) = I_0(\tau) + h_1\ \Delta y_1(\tau) + \widetilde{\Psi}\left[\underline{\breve{x}}(\tau),\ \widehat{x}_1(\tau)\right] \tag{4.77d}$$

$$\dot{I}_0(\tau) = h_0\ \Delta y_1(\tau) \tag{4.77e}$$

$$\Delta y_1(\tau) = y(\tau) - \widehat{x}_1(\tau) \tag{4.77f}$$

avec la fonction $\widetilde{\Psi}\left[\underline{\breve{x}}(\tau), \widehat{x}_1(\tau)\right]$:

$$\begin{aligned}\widetilde{\Psi}\left[\underline{\breve{x}}(\tau),\ \widehat{x}_1(\tau)\right] = \quad & -\frac{1+\sigma+\beta}{\omega_o}\ \breve{x}_3(\tau) - \frac{\beta\ (1+\sigma)}{{\omega_o}^2}\ \breve{x}_2(\tau) + \frac{\beta\ \sigma\ (\rho-1)}{{\omega_o}^3}\ \widehat{x}_1(\tau) \\ & -\frac{\widehat{x}_1(\tau)^2\ \breve{x}_2(\tau)}{{\omega_o}^2} - \frac{\sigma\ \widehat{x}_1(\tau)^3}{{\omega_o}^3} + \frac{\breve{x}_2(\tau)\ \breve{x}_3(\tau) + \dfrac{1+\sigma}{\omega_o}\ \breve{x}_2(\tau)^2}{\widehat{x}_1(\tau)}\end{aligned} \tag{4.78}$$

Le système de transformation liant l'observateur à l'espace d'état originel est le suivant :

$$\widehat{z}_x(\tau) = \widehat{x}_1(\tau) \tag{4.79a}$$

$$\widehat{z}_y(\tau) = \widehat{x}_1(\tau) + \left(\frac{\omega_o}{\sigma}\right) \breve{x}_2(\tau) \tag{4.79b}$$

$$\widehat{z}_z(\tau) = \rho - 1 - \left(\omega_o + \frac{\omega_o}{\sigma}\right) \frac{\widehat{x}_2(\tau)}{\widehat{x}_1(\tau)} - \left(\frac{{\omega_o}^2}{\sigma}\right) \frac{\breve{x}_3(\tau)}{\widehat{x}_1(\tau)} \tag{4.79c}$$

En utilisant (4.48) pour $n = 3$, $\nu = 1$, on obtient les gains :

$$\underline{h_a}^T = \begin{bmatrix} 1 & 4 & 6 & 4 \end{bmatrix} \tag{4.80}$$

Dans ce cas, les conditions de stabilité (4.35) et (4.36b) seront respectées. Les conditions initiales de l'observateur sont fixées à $\underline{\breve{x}}(0) = \underline{0}$, $\widehat{x}_1(0) = 1$, $\widehat{x}_2(0) = 0$. Les FIGURES 4.3(c) et (d) illustrent la convergence de l'observateur en phase de stabilisation comparées à celle de l'observateur de *Gautier*. Les réponses sont très similaires. Le test suivant est mené en situation bruitée, en superposant à $x_1(\tau)$ un bruit blanc à bande passante limitée d'amplitude 0.2. Pour l'observateur (4.14), on visualise les distances :

$$\Delta\breve{x}_i(\tau) = x_i(\tau) - \breve{x}_i(\tau) \quad i = 2 \ldots n \tag{4.81}$$

De même, pour l'observateur de *Gauthier*, on visualise :

$$\Delta\widehat{x}_i(\tau) = x_i(\tau) - \widehat{x}_i(\tau) \quad i = 2 \ldots n \tag{4.82}$$

Les FIGURES 4.3(e), 4.3(f), 4.4(a), 4.4(b), 4.4(c), 4.4(d) illustrent les résultats obtenus à partir des mesures bruitées $y(\tau)$, avec pour le système et l'observateur les mêmes conditions initiales que celles du test sans bruit de mesure. Les distances d'état obtenues pour l'observateur (4.14) sont systématiquement plus faibles que celles de l'observateur de *Gauthier*, excepté pour les variables d'état d'ordre n, qui sont comparables. L'observateur (4.14) se révèle comme plus robuste au bruit, et produit les meilleures estimations des variables $z_y(\tau)$, $z_z(\tau)$ après la transformation (4.79).

4.5.2 Modèle de boues activées de type ASM1

Le modèle simplifié de traitement des boues activées ASM1 utilisé par (Nagy-Kiss et al. 2010), structurellement du même type que (4.1) s'écrit :

$$\dot{z}_i(t) = s_i(t) \quad i = 1 \dots 3 \tag{4.83a}$$
$$y_1(t) = z_1(t) \quad y_2(t) = z_2(t) \tag{4.83b}$$

avec :

$$s_1(t) = k_1\ \ell_1(t) + k_2\ z_3(t) - k_3\ \ell_2(t)\ z_3(t) \tag{4.84a}$$
$$s_2(t) = k_4\ \ell_3(t) - k_1\ \ell_4(t) - k_5\ \ell_2(t)\ z_3(t) \tag{4.84b}$$
$$s_3(t) = k_9\ \ell_5(t) - k_6\ \ell_6(t) - k_7\ z_3(t) + k_8\ \ell_2(t)\ z_3(t) \tag{4.84c}$$

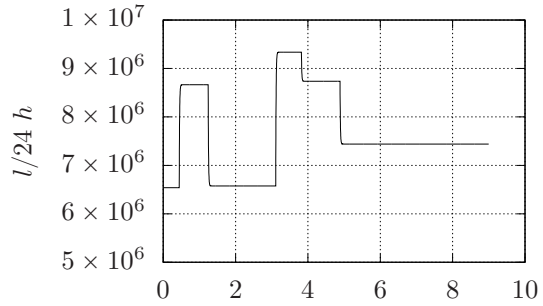
et les fonctions non linéaires :

$$\ell_1(t) = u_{11}(t)\ (\ u_{31}(t) - z_1(t)\ ) \tag{4.85a}$$
$$\ell_2(t) = \frac{z_1(t)\ z_2(t)}{(\ k_{10} + z_1(t)\ )(\ k_{11} + z_2(t)\ )} \tag{4.85b}$$
$$\ell_3(t) = u_{21}(t)\ [\ k_{12} - z_2(t)\ ] \tag{4.85c}$$
$$\ell_4(t) = u_{11}(t)\ z_2(t) \tag{4.85d}$$
$$\ell_5(t) = u_{11}(t)\ u_{41}(t) \tag{4.85e}$$
$$\ell_6(t) = u_{11}(t)\ z_3(t) \tag{4.85f}$$
$$\underline{u}_1(t) = \begin{bmatrix} u_{11}(t) & \dots & u_{41}(t) \end{bmatrix} \tag{4.85g}$$

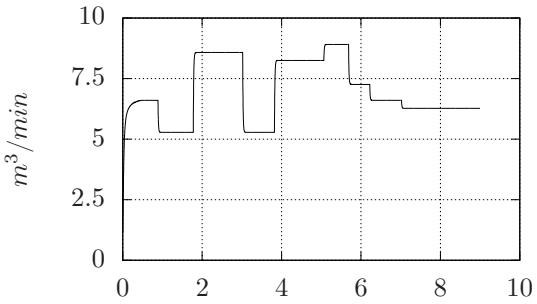
Les constantes sont données par :

$$\begin{array}{lll} k_1 = 5 \cdot 10^{-11} & k_2 = 1,08 \cdot 10^{-5} & k_3 = 2.872 \cdot 10^{-4} \\ k_4 = 3.5\ 10^{-4} & k_5 = 9 \cdot 10^{-5} & k_6 = 1.316 \cdot 10^{-12} \\ k_7 = 4.8 \cdot 10^{-6} & k_8 = 7.47 \cdot 10^{-5} & k_9 = 8 \cdot 10^{-11} \\ k_{10} = 20 & k_{11} = 0.2 & k_{12} = 10 \end{array} \tag{4.86}$$

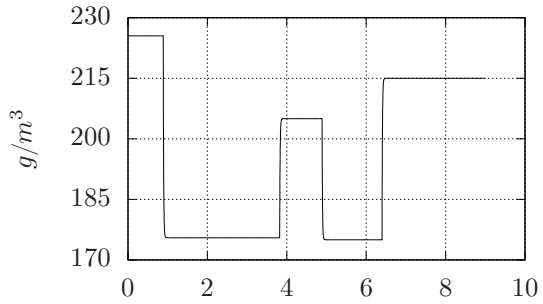
$\underline{u}_1(t)$ (4.85g) représente les entrées du système (FIGURES 4.5 (a),(b),(c),(d) page 44), respectivement le débit d'eau, le débit d'air injecté, l'injection de substrat carboné soluble recyclé, la concentration particulaire de biomasse recyclée hétérotrophe. Toutes les abscisses des figures sont graduées en heures.
Les variables $z_1(t)$, $z_2(t)$ $z_3(t)$ représentent les états du réacteur (FIGURES 4.5 (e),(f),(g)), respectivement la concentration en substrat carboné rapidement biodégradable, la concentration en oxygène dissous, la concentration particulaire de biomasse, avec (4.86) ses paramètres, tous connus, et $z_1(0) = 4.1$, $z_2(0) = 3.0$, $z_3(0) = 867$ ses conditions initiales. Les grandeurs $y_1(t)$, $y_2(t)$ (4.83b) représentent les sorties mesurées du système.



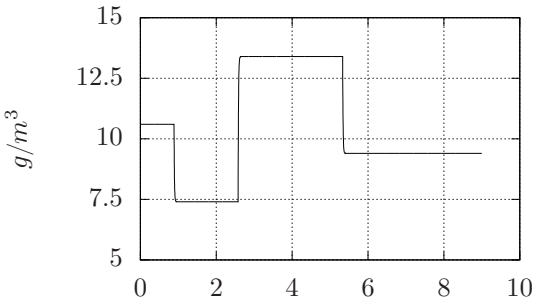
(a) $u_{11}(t)$, débit d'eau



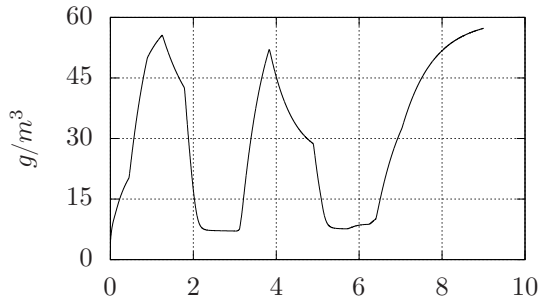
(b) $u_{21}(t)$, débit d'air injecté



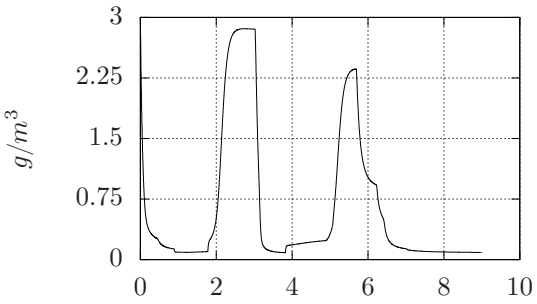
(c) $u_{31}(t)$, injection de substrat carboné



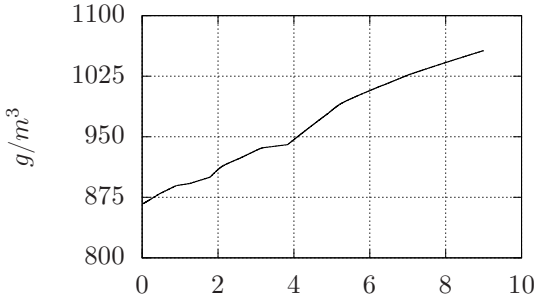
(d) $u_{41}(t)$, réinjection de biomasse



(e) $z_1(t)$, concentration en substrat carboné biodégradable



(f) $z_2(t)$, concentration en oxygène dissous



(g) $z_3(t)$, concentration en biomasse

FIGURE 4.5: Variables d'entrées et variables d'état du bioréacteur

Modèle transformé sous une forme canonique de régulation

Transformons des deux équations différentielles d'ordre 1 (4.84b) et (4.84c) en une seule équation différentielle d'ordre deux. Avec cette dernière, on détermine une fonction $\Psi\,[\,\underline{x}(t),\,\underline{U}(t)\,]$ et un système que l'on peut exprimer sous une forme canonique de régulation. Cette application de la procédure générale de transformation (Fliess 1990) permet le passage des systèmes (4.1) à (4.2) et autorise l'utilisation d'un observateur similaire à celui proposé en (4.14), associé à la transformation inverse (4.49) et aux observateurs (4.58).

Dans notre exemple, on cherche à observer $y_2(t)$ et ses dérivées successives, afin d'en déduire une estimation de la variable d'état non mesurable $z_3(t)$. De $s_2(t)$ (4.84b) on déduit $z_3(t)$:

$$z_3(t) = \frac{k_4\ \ell_3(t) - k_1\ \ell_4(t) - \dot{z}_2(t)}{k_5\ \ell_2(t)} \tag{4.87}$$

La dérivée temporelle de (4.87) donne l'expression de $\dot{z}_3(t)$, qui peut être égalée à $s_3(t)$ (4.83b). On en déduit :

$$\dot{x}_1(t) = x_2(t) \tag{4.88a}$$
$$\dot{x}_2(t) = \Psi\,[\,\underline{x}(t),\,\underline{U}(t)\,] \tag{4.88b}$$
$$\Psi\,[\,\underline{x}(t),\,\underline{U}(t)\,] = k_4\ \dot{\ell}_3(t) - k_1\ \dot{\ell}_4(t) - k_5\ \ell_7(t) \tag{4.88c}$$
$$\ell_2(t) = \frac{z_1(t)\ x_1(t)}{(\ k_{10} + z_1(t)\ )\,(\ k_{11} + x_1(t)\ )} = n(t)/d(t) \tag{4.88d}$$
$$\ell_3(t) = u_{21}(t)\ \ [\ k_{12} - x_1(t)\ ] \tag{4.88e}$$
$$\ell_4(t) = u_{11}(t)\ x_1(t) \tag{4.88f}$$
$$\ell_6(t) = u_{11}(t)\ g_3(t) \tag{4.88g}$$
$$\ell_7(t) = g_3(t)\ \dot{\ell}_2(t) + s_3(t)\ \ell_2(t) \tag{4.88h}$$
$$z_2(t) = x_1(t) \tag{4.88i}$$

Le système (4.88) décrit une équation différentielle d'ordre deux, structurellement arrangée sous une forme canonique de régulation du même type que celle décrite en (4.2). Les fonctions dérivées $\dot{\ell}_3(t)$, $\dot{\ell}_4(t)$ de (4.88c), et $\dot{\ell}_2(t)$, $s_3(t)$ de (4.88h) sont définies par :

$$\dot{\ell}_2(t) = \dot{n}(t)\ d(t) - \dot{d}(t)\ n(t)/d(t)^2 \tag{4.89a}$$
$$\dot{n}(t) = s_1(t)\ x_1(t) + z_1(t)\ x_2(t) \tag{4.89b}$$
$$\dot{d}(t) = s_1(t)\ (k_{11} + x_1(t)) + x_2(t)\ (k_{10} + z_1(t)) \tag{4.89c}$$
$$s_1(t) = k_1\ \ell_1(t) + k_2\ g_3(t) - k_3\ \ell_2(t)\ g_3(t) \tag{4.89d}$$
$$\dot{\ell}_3(t) = u_{22}(t)\,(\ k_{12} - x_1(t)\ ) - u_{21}(t)\ x_2(t) \tag{4.89e}$$
$$\dot{\ell}_4(t) = u_{12}(t)\ x_1(t) + u_{11}(t)\ x_2(t) \tag{4.89f}$$
$$s_3(t) = k_9\ \ell_5(t) - k_6\ \ell_6(t) - k_7\ g_3(t) + k_8\ \ell_2(t)\ g_3(t) \tag{4.89g}$$

L'équation $\dot{z}_1(t) = s_1(t)$, définie en (4.83a) est conservée, et l'intégration de $s_1(t)$ produit $z_1(t)$, qui est la sortie mesurée définie en (4.83b). Le vecteur des fonctions de transformations inverses (4.6b) doit permettre, dans notre exemple, de déterminer $z_3(t)$. Il s'écrit :

$$z_1(t) = g_1(t) \tag{4.90a}$$
$$z_2(t) = g_2(t) \tag{4.90b}$$
$$z_3(t) = g_3(t) = \frac{k_4\ \ell_3(t) - k_1\ \ell_4(t) - x_2(t)}{k_5\ \ell_2(t)} \tag{4.90c}$$

et permet de lier $\underline{x}(t)$ à $\underline{z}(t)$, qui est une fonction non linéaire de $\underline{U}(t)$ et $\underline{x}(t)$ à travers $\ell_2(t)$, $\ell_3(t)$ et $\ell_4(t)$.

Changement d'échelle de $\widetilde{\Psi}\,[\underline{x}(t),\,\underline{U}(t)]$, observation de $z_2(t)$ sous forme canonique de régulation et détermination du système de transformation inverse

Á l'aide de (4.5) on norme temporellement (4.88b), et on définit à présent $\widetilde{\Psi}(\tau)$ à l'aide des variables d'entrées-sorties :

$$\widetilde{\Psi}(\tau) = \frac{\widetilde{\Psi}\,[\,\underline{v}_a(t),\,\underline{U}(t)\,]}{{\omega_o}^2} \tag{4.91a}$$

$$\underline{v}_a(t)^T = \left[\,\underline{v}(t) \quad \breve{z}_1(t)\right] \tag{4.91b}$$

$$\underline{v}(t)^T = \left[\,\widehat{x}_1(t) \quad \breve{x}_2(t)\right] \tag{4.91c}$$

$$\breve{x}_2(t) = \omega_o\,\breve{x}_2(\tau) \quad \widehat{x}_1(t) = \widehat{x}_1(\tau) \tag{4.91d}$$

$$\underline{U}(t) = \left[\,\underline{u}_1(t) \quad \underline{u}_2(t)\right] \tag{4.91e}$$

La pulsation normative choisie pour (4.91) est $\omega_o = 3.927\ 10^{-2}\ rd/s$. Dans (4.91b), on définit $\underline{v}_a(t)$ comme le vecteur $\underline{v}(t)$ (4.14i) augmenté de la variable $\breve{z}_1(t)$, elle-même résultante de l'observation de la variable mesurée $y_1(t)$.
On note que $\underline{v}(t)$ contient deux termes d'ordre deux du fait de (4.88). L'équation (4.91d) autorise la conversion de variables d'états en temps adimensionnel en variables d'état temporelles. En tenant compte de (4.5) et des définitions (4.91), on écrit la fonction $\widetilde{\Psi}(\tau)$ en échelle de temps normée comme suit :

$$\widetilde{\Psi}(\tau) = \frac{k_4\,\dot{\widehat{\ell}}_3(t) - k_1\,\dot{\widehat{\ell}}_4(t) - k_5\,\widehat{\ell}_7(t)}{{\omega_o}^2} \tag{4.92a}$$

$$\widehat{\ell}_1(t) = u_{11}(t)\,(\,u_{31}(t) - \breve{z}_1(t)\,) \tag{4.92b}$$

$$\widehat{\ell}_2(t) = \widehat{n}(t)/\widehat{d}(t) \tag{4.92c}$$

$$\widehat{\ell}_3(t) = u_{21}(t)\,\left[\,k_{12} - \widehat{x}_1(t)\,\right] \tag{4.92d}$$

$$\widehat{\ell}_4(t) = u_{11}(t)\,\widehat{x}_1(t) \tag{4.92e}$$

$$\widehat{\ell}_5(t) = u_{11}(t)\,u_{41}(t) \tag{4.92f}$$

$$\widehat{\ell}_6(t) = u_{11}(t)\,\widehat{z}_3(t) \tag{4.92g}$$

$$\widehat{\ell}_7(t) = \widehat{z}_3(t)\,\dot{\widehat{\ell}}_2(t) + \dot{\widehat{z}}_3(t)\,\widehat{\ell}_2(t) \tag{4.92h}$$

$$\widehat{n}(t) = \breve{z}_1(t)\,\widehat{x}_1(t) \tag{4.92i}$$

$$\widehat{d}(t) = (\,k_{10} + \breve{z}_1(t)\,)\,(\,k_{11} + \widehat{x}_1(t)\,) \tag{4.92j}$$

Les fonctions dérivées $\dot{\widehat{\ell}}_3(t)$, $\dot{\widehat{\ell}}_4(t)$ de (4.92a) et $\dot{\widehat{\ell}}_2(t)$, $\dot{\widehat{z}}_3(t)$ de (4.92h) sont définies par :

$$\dot{\widehat{\ell}}_2(t) = \frac{\dot{\widehat{n}}(t)\,\widehat{d}(t) - \dot{\widehat{d}}(t)\,\widehat{n}(t)}{\widehat{d}(t)^2} \tag{4.93a}$$

$$\dot{\widehat{n}}(t) = \dot{\widehat{z}}_1(t)\,\widehat{x}_1(t) + \breve{z}_1(t)\,\breve{x}_2(t) \tag{4.93b}$$

$$\dot{\widehat{d}}(t) = \dot{\widehat{z}}_1(t)\,(\,k_{11} + \widehat{x}_1(t)\,) + \breve{x}_2(t)\,(\,k_{10} + \breve{z}_1(t)\,) \tag{4.93c}$$

$$\dot{\widehat{\ell}}_3(t) = u_{22}(t)\,(\,k_{12} - \widehat{x}_1(t)\,) - u_{21}(t)\,\breve{x}_2(t) \tag{4.93d}$$

$$\dot{\widehat{\ell}}_4(t) = u_{12}(t)\,\widehat{x}_1(t) + u_{11}(t)\,\breve{x}_2(t) \tag{4.93e}$$

$$\dot{\widehat{z}}_1(t) = k_1\,\widehat{\ell}_1(t) + k_2\,\widehat{z}_3(t) - k_3\,\widehat{\ell}_2(t)\,\widehat{z}_3(t) \tag{4.93f}$$

$$\dot{\widehat{z}}_3(t) = k_9\,\widehat{\ell}_5(t) - k_6\,\widehat{\ell}_6(t) - k_7\,\widehat{z}_3(t) + k_8\,\widehat{\ell}_2(t)\,\widehat{z}_3(t) \tag{4.93g}$$

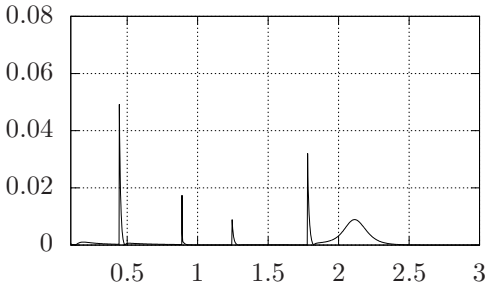
L'observateur sous forme canonique de régulation du système (4.88) s'écrit :

$$\dot{\widehat{x}}_1(\tau) = \breve{x}_2(\tau) + h_2\,\Delta y_1(\tau) \tag{4.94a}$$

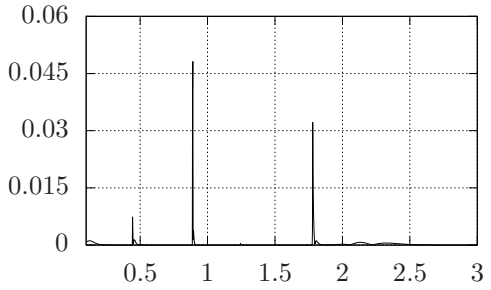
$$\dot{\breve{x}}_2(\tau) = I_0(\tau) + h_1\,\Delta y_1(\tau) + \widetilde{\Psi}(\tau) \tag{4.94b}$$

$$\dot{I}_0(\tau) = h_0\,\Delta y_1(\tau) \tag{4.94c}$$

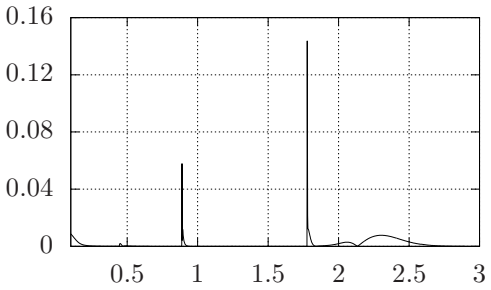
$$\Delta y_1(\tau) = y_2(\tau) - \widehat{x}_1(\tau) \tag{4.94d}$$



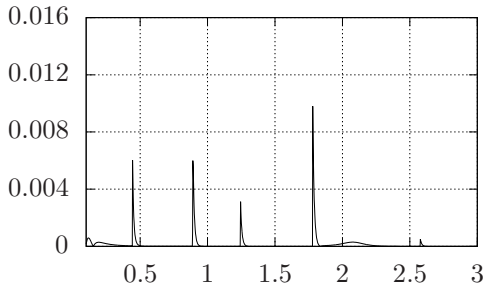
(a) Évolution de $\| \Delta\dot{\tilde{\Psi}}_1(\tau_1) \| / \| \Delta\underline{z}_1(\tau_1) \|$



(b) Évolution de $\| \Delta\dot{\tilde{\Psi}}(\tau) \| / \| \Delta\underline{y}_a(\tau) \|$

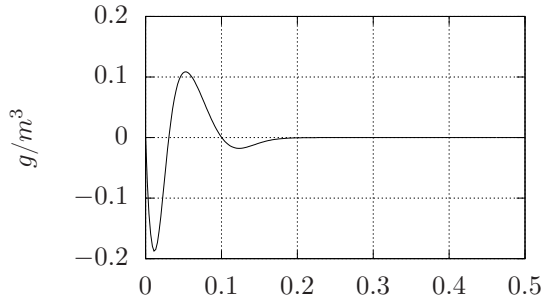


(c) Évolution de $\| \Delta\dot{\tilde{\Psi}}_2(\tau_2) \| / \| \Delta\underline{z}_2(\tau_2) \|$

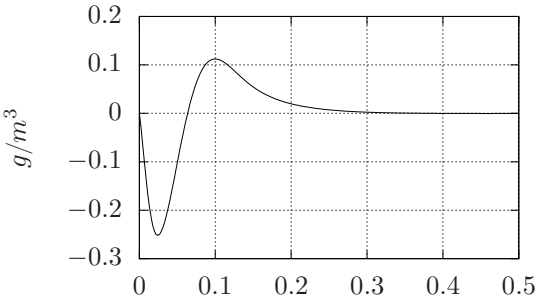


(d) Évolution de $\| \Delta\dot{\tilde{\Psi}}_3(\tau_3) \| / \| \Delta\underline{z}_3(\tau_3) |$

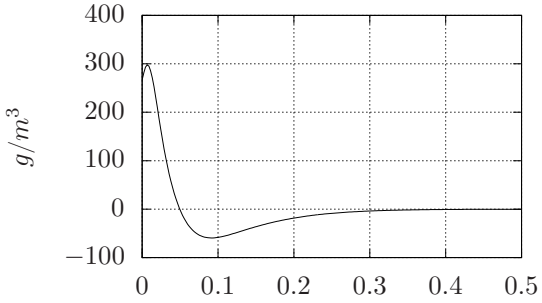
FIGURE 4.6: Recherche des constantes de *Lipschitz*



(a) Différence $z_1(t) - \check{z}_1(t)$ de substrat carboné

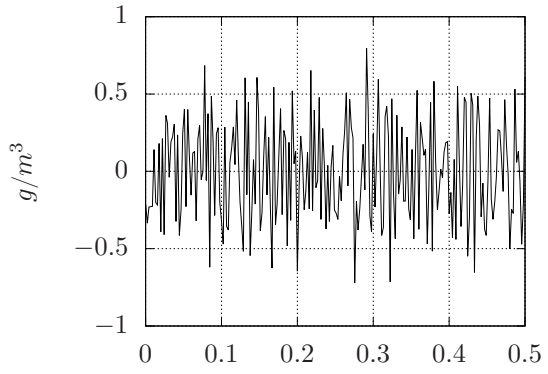


(b) Différence $z_2(t) - \check{z}_2(t)$ d'oxygène dissous

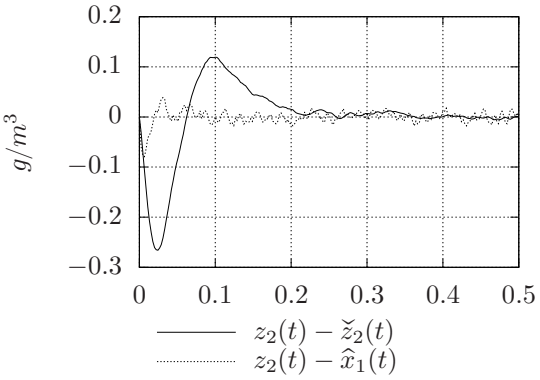


(c) Différence $z_3(t) - \check{z}_3(t)$ de biomasse

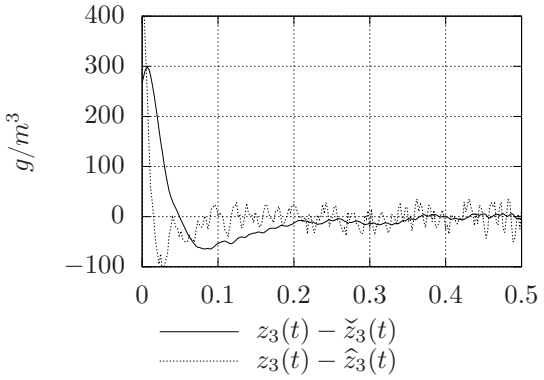
FIGURE 4.7: Distances d'état sans bruit de mesure



(a) Différence $z_1(t) - \breve{z}_1(t)$ de substrat carboné



(b) Différence $z_2(t) - \breve{z}_2(t)$ d'oxygène dissous



(c) Différence $z_3(t) - \breve{z}_3(t)$ de biomasse

FIGURE 4.8: Distances d'état avec bruit de mesure

avec $y_2(\tau)$ (4.83b) utilisée pour former l'erreur d'observation $\Delta y_1(\tau)$. L'estimée $\widehat{z}_3(t)$ de $z_3(t)$ utilisée en (4.92h), (4.93f) et (4.93g), de même que l'estimée $\widehat{z}_2(t)$ de $z_2(t)$ sont définies à l'aide des fonctions de transformations inverses suivantes :

$$\widehat{z}_1(t) = y_1(t) \tag{4.95a}$$

$$\widehat{z}_2(t) = \widehat{x}_1(t) \tag{4.95b}$$

$$\widehat{z}_3(t) = \frac{k_4\, \widehat{\ell}_3(t) - k_1\, \widehat{\ell}_4(t) - \breve{x}_2(t)}{k_5\, \widehat{\ell}_2(t)} \tag{4.95c}$$

$$\underline{\widehat{z}}(t) = \begin{bmatrix} \widehat{z}_1(t) & \widehat{z}_2(t) & \widehat{z}_3(t) \end{bmatrix} \tag{4.95d}$$

Observation du système de transformation inverse

Le système de transformation inverse (4.95) sert à former l'erreur d'observation (4.14c) de trois observateurs similaires à ceux définis en (4.58), dans le but d'estimer $\underline{\widehat{z}}(t)$. Ces variables et les pulsations normatives qui leur sont associées sont définies par :

$$\underline{\breve{z}}(t) = \begin{bmatrix} \breve{z}_1(t) & \breve{z}_2(t) & \breve{z}_3(t) \end{bmatrix} \tag{4.96a}$$

$$\breve{s}_i(\tau) = \frac{s_i\left[\underline{u}_1(t), \underline{\breve{z}}(t)\right]}{\omega_i} \quad i = 1 \ldots 3 \tag{4.96b}$$

$$\omega_1 = \omega_o \quad \omega_2 = \omega_3 = \omega_o/5 \tag{4.96c}$$

L'observateur de $\widehat{z}_1(t)$ s'écrit :

$$\dot{\breve{z}}_1(\tau_1) = I_1(\tau_1) + 2\ \Delta z_1(\tau_1) + \breve{s}_1(\tau_1) \tag{4.97a}$$

$$\breve{s}_1(\tau_1) = \Big[\ k_1\ \widehat{\ell}_1(t) + k_2\ \breve{z}_3(t) - k_3\ \breve{\ell}_2(t)\ \breve{z}_3(t)\ \Big]/\omega_1 \tag{4.97b}$$

$$\breve{\ell}_2(t) = \frac{\breve{z}_1(t)\ \breve{z}_2(t)}{(\ k_{10} + \breve{z}_1(t)\ )\ (\ k_{11} + \breve{z}_2(t)\ )} \tag{4.97c}$$

$$\dot{I}_1(\tau_1) = \Delta z_1(\tau_1) \tag{4.97d}$$

$$\Delta z_1(\tau_1) = \widehat{z}_1(t) - \breve{z}_1(\tau_1) \tag{4.97e}$$

Celui de $\widehat{z}_2(t)$:

$$\dot{\breve{z}}_2(\tau_2) = I_2(\tau_2) + 2\ \Delta z_2(\tau_2) + \breve{s}_2(\tau_2) \tag{4.98a}$$

$$\breve{s}_2(\tau_2) = \Big[\ k_4\ \breve{\ell}_3(t) - k_1\ \breve{\ell}_4(t) - k_5\ \breve{\ell}_2(t)\ \breve{z}_3(t)\ \Big]/\omega_2 \tag{4.98b}$$

$$\breve{\ell}_3(t) = u_{21}(t)\ [\ k_{12} - \breve{z}_2(t)\ ] \tag{4.98c}$$

$$\breve{\ell}_4(t) = u_{11}(t)\ \breve{z}_2(t) \tag{4.98d}$$

$$\dot{I}_2(\tau_2) = \Delta z_2(\tau_2) \tag{4.98e}$$

$$\Delta z_2(\tau_2) = \widehat{z}_2(t) - \breve{z}_2(\tau_2) \tag{4.98f}$$

Celui de $\widehat{z}_3(t)$:

$$\dot{\breve{z}}_3(\tau_3) = I_3(\tau_3) + 2\ \Delta z_3(\tau_3) + \breve{s}_3(\tau_3) \tag{4.99a}$$

$$\breve{s}_3(\tau_3) = \begin{bmatrix} k_9\ \widehat{\ell}_5(t) - k_6\ u_{11}(t)\ \breve{z}_3(t) \\ -k_7\ \breve{z}_3(t) + k_8\ \breve{\ell}_2(t)\ \breve{z}_3(t) \end{bmatrix}/\omega_3 \tag{4.99b}$$

$$\dot{I}_3(\tau_3) = \Delta z_3(\tau_3) \tag{4.99c}$$

$$\Delta z_3(\tau_3) = \widehat{z}_3(t) - \breve{z}_3(\tau_3) \tag{4.99d}$$

Le rôle de l'observateur (4.97) est de contrebalancer les effets du bruit superposé à $z_1(t)$, qui a par moments un gros impact sur les estimées $\widehat{z}_3(t)$ (4.95c), à cause du terme $\widehat{\ell}_2(t)$ présent dans le dénominateur.
Les facteurs d'échelle ω_2 et ω_3 de (4.98b) et (4.99b), équations utilisées dans (4.97b), sont fortement réduits par rapport à ω_o. Ceci a pour effet de réduire le bruit superposé aux estimées $\breve{z}_2(t)$ et $\breve{z}_3(t)$.

On cherche à présent à déterminer les constantes de *Lipschitz* qui vont permettre de définir les conditions de stabilité de chaque observateur. Commençons par la recherche de L pour (4.88).
En utilisant les conditions initiales $\underline{z}(0)$ on peut à l'aide de (4.90) reconstruire le vecteur $\underline{x}(0)$, et ensuite déterminer $\widetilde{\Psi}\left[\underline{x}(0), \underline{U}(t)\right]$. On calcule ensuite les distances d'état $\Delta\underline{\dot{y}}(\tau)$. La dérivation numérique de $\widetilde{\Psi}\left[\underline{x}(t), \underline{U}(t)\right]/{\omega_o}^2 - \widetilde{\Psi}\left[\underline{x}(0), \underline{U}(t)\right]/{\omega_o}^2$ permet la détermination de $\underline{y}_a(\tau)$ et le calcul d'un rapport $L \geqslant \| \Delta\dot{\widetilde{\Psi}}(\tau) \| / \| \Delta\underline{y}_a(\tau) \|$ au cours du temps. La FIGURE 4.6(b) page 47 illustre cette procédure et permet de choisir une constante $L = 0.06$ par exemple, qui sera inférieure au $L_{max} = 0,55$ du TABLEAU 4.1 page 34.
Sur l'axe des abscisses seules les trois premières heures de l'enregistrement sont représentées, qui correspondent à la région où la convergence des observateurs est attendue. En utilisant $\nu = 1$ et $n = 2$ dans (4.48c), on fixe les gains :

$$\underline{h}_a = \begin{bmatrix} 1 & 3 & 3 \end{bmatrix} \tag{4.100}$$

On respecte alors les conditions (4.35) et (4.36b), l'observateur (4.94) sera donc stable.

On cherche à présent à déterminer les constantes L_1, L_2, L_3 des observateurs des fonctions de transformations inverses. On utilise une procédure similaire à ce qui vient d'être évoqué plus haut pour estimer les $\Delta\underline{z}_i(\tau_i)$, $\Delta\dot{\widetilde{\Psi}}_i(\tau_i)$ et leurs modules respectifs. Les FIGURES 4.6(a),(c) et (d) permettent de choisir des constantes de *Lipschitz* $L_1 = 0.08$, $L_2 = L_3 = 0.016$ (4.89) toutes inférieures au $L_{max} = 0,55$ obtenu pour $n = 2$ (TABLEAU 4.1).
On respecte alors les conditions (4.64) et (4.65c), les trois observateurs (4.97)-(4.99) seront stables.

Simulations et résultats obtenus

L'objectif de la simulation est d'observer la stabilisation générale des observateurs pour une différence initiale de concentration en biomasse.

Les conditions initiales de (4.94) sont fixées à $I_0(0) = 0$, $\breve{x}_2(0) = -0.164$, $\widehat{x}_1(0) = z_2(0) = 3$
Celles de (4.97) à $\breve{z}_1(0) = z_1(0) = 4.1$.
Celles de (4.97) à $\breve{z}_2(0) = 3$.
Celles de (4.99) à $\breve{z}_3(0) = 600$.
Les FIGURES 4.7(a)-4.7(c) illustrent la réduction exponentiellement convergente des distances d'états $z_i(t) - \breve{z}_i(t)$ pour des sorties mesurées $y_1(t)$ et $y_2(t)$ non bruitées. Le même test est mené en superposant deux bruits blancs à bandes passantes limitées sur les mêmes sorties. Ces bruits non corrélés ont une amplitude d'1% de la pleine échelle des variables. La FIGURE 4.8(c) page 48 permet de vérifier si la dynamique de convergence de $\breve{z}_3(t)$ est conservée. La pulsation normative ω_o utilisée pour (4.94) et (4.97) permet la réduction du bruit sur les estimées de 10% par rapport aux variables mesurées et à contenir celui restant présent sur $\widehat{z}_3(t)$. Ce réglage permet par contre d'avoir une vitesse de convergence de $\breve{z}_1(t)$ et $\widehat{x}_1(t)$ du même ordre de grandeur que les variations rapides qui peuvent être observées avec $z_1 t)$ et $z_2(t)$.
Sur les FIGURES 4.8(a)-(c), l'effet filtrant sur les estimées des observateurs (4.97) et (4.99) est mis en évidence : division par 5 du bruit sur $y_2(t)$ pour $\breve{z}_2(t)$ et fluctuation de 2% du bruit superposé à $\breve{z}_3(t)$ par rapport à la pleine échelle.

4.6 Bilan des développements menés sur les observateurs

Le nouvel observateur MISO proposé est structurellement construit pour fournir des estimations d'état sous une forme canonique de régulation autant que sous la forme des équations d'états non linéaires originelles des systèmes. Le champ d'application couvert par cette approche couvre l'intégralité des systèmes décomposables par une transformation de *Fliess* (Fliess 1990) et uniformément observables au sens de *Hermann et Gauthier* (Hermann & Krener 1977, Gauthier & Bornard 1981).
Ses conditions de stabilité sont rigoureusement analysées et garanties si les non linéarités des systèmes observés sont au moins localement *Lipschitz*, ce qui est bien souvent le cas des applications du physicien ou de l'ingénieur. Dans ce cas, on démontre le caractère exponentiellement convergent des estimées.

Pour résoudre le problème de la synthèse des paramètres, un polynôme de *Newton* couplé à une méthode de conditionnement des fonctions non linéaires est proposé, qui permet une approche algébrique simple et efficace, et de se passer des méthodes d'optimisation classiquement utilisées. La stratégie PI de l'observateur permet de modéliser les fluctuations de mesures non stationnaires exogènes au système observé.

La structure sous forme canonique de régulation offre une grande robustesse au bruit pour la détermination des dérivées successives de la variable de sortie, et possède l'avantage non négligeable d'être directement exploitable pour réaliser le contrôle du système, sans aucune transformation additionnelle.

Le rajout des observations basés sur les fonctions de transformation inverse permet le contrôle du bruit également sur les estimées des variables d'états du système originel. Les paramètres normatifs ω_i servant à l'écriture des observateurs des fonctions inverses en temps adimensionnel permet également le contrôle du bruit sur les estimées du vecteur $\underline{z}(t)$ du système (4.1). Cette technique, efficace pour les observateurs sous forme canonique de régulation, peut également s'envisager pour des observateurs à fort gain comme ceux de (Gauthier et al. 1992).

Chapitre 5

Développement du langage « oS »

5.1 A quoi est destiné le langage « oS »

Les développements théoriques exposés au chapitre 4 ont été vérifiés et simulés à l'aide d'un outil de calcul développé par mes soins : il s'agit d'un langage de programmation destiné à la modélisation de systèmes dynamiques, « oS »(objects simulation), dont le formalisme mathématique est dérivé des équations et des machines d'état.

L'idée générale qui a guidé ce travail est de disposer d'un outil proche de la notion de diagramme fonctionnel au moment de l'écriture d'un programme, en oubliant les classiques variables et tests des langages de programmation comme « C », « C++ » ou « Pascal ». Les diagrammes fonctionnels d'état suggèrent naturellement l'idée de flux d'information, et de leur traitement et continu. Disposer d'un outil permettant de s'adapter à ce type d'analyse permet de se focaliser sur la description des propriétés des systèmes dynamiques ou discrets.

Le langage « oS » est un langage sans véritables variables, où l'on décrit son(ses) modèle(s) sous forme d'objets, reliés entre eux par des liens symbolisant des flux de données. Chaque objet est une entité qui effectue un traitement sur les données qui lui parviennent, et met le résultat de ses traitements à disposition d'autres objets par l'intermédiaire de liens.
Les objets sont constitués eux-mêmes d'éléments ou opérateurs, dont la définition et l'interconnection est entièrement synthétisée sous forme de classes. Ces classes constituent en quelque sorte un plan descriptif, matérialisé ensuite par l'objet.
Raisonner ainsi en terme d'objets formant des entités propres, que l'on décrit par un assemblage d'éléments ayant chacun leurs propres paramètres et interconnexions permet de s'affranchir de lourdes manipulations de variables et de mécanismes d'allocation mémoires, qui deviennent transparents pour l'utilisateur. A charge pour ce dernier de déclarer à bon escient l'ordonnancement de ses objets, et les propriétés qui en découlent. A ce titre, on peut dire que le langage « oS » est un langage orienté objet très déclaratif.

Le compilateur « os2c » qui analyse le langage « oS » va produire en sortie un code source « C » ANSI. Celui-ci est massivement constitué d'appels aux fonctions d'une bibliothèque appelée « libsim.a »Le langage « oS » produit ainsi un vrai code « C » compilable sous forme d'exécutable sur tout type de machine, et d'une grande rapidité d'exécution.
La bibliothèque « libsim.a » est le cœur de l'outil de simulation. Le langage « oS » vient l'encapsuler pour gérer automatiquement tout ce qu'il faudrait faire en « C » pour obtenir le même résultat.

5.2 Principe d'une simulation de systèmes dynamiques

On décompose toujours notre système à simuler en plusieurs entrées $u_i, i \in [1, p]$ et plusieurs sorties $v_j, j \in [1, q]$, ces dernières liées par des fonctions d'entrées g_i et f_j. L'ensemble répond ensuite aux équations d'état :

$$\left.\begin{array}{r} \dot{x_1} = f_1(x_1 \dots x_n; u_1 \dots u_p) \\ \vdots \\ \dot{x_n} = f_n(x_1 \dots x_n; u_1 \dots u_p) \\ v_1 = g_1(x_1 \dots x_n; u_1 \dots u_p) \\ \vdots \\ v_q = g_q(x_1 \dots x_n; u_1 \dots u_p) \end{array}\right\} \quad (5.1)$$

Les dérivées $\dot{x_i}(t)$ peuvent être approximée et discrétisées à partir des formules de différences finies :

$$\left[\frac{d\, x_i(t)}{dt} \right]_{t=k\Delta t} \approx \frac{x_i(k \cdot \Delta t) - x_i\left((k-1) \cdot \Delta t\right)}{\Delta t} \tag{5.2}$$

Cette approximation est utilisée tel quel pour le calcul de dérivées, et peut-être utilisée également pour l'approximation du processus d'intégration :

$$x_i(k \cdot \Delta t) = \dot{x_i}\left(k \cdot \Delta t\right) \cdot \Delta t + x_i\left((k-1) \cdot \Delta t\right) \tag{5.3}$$

Ces équations sont illustrées par la figure suivante :

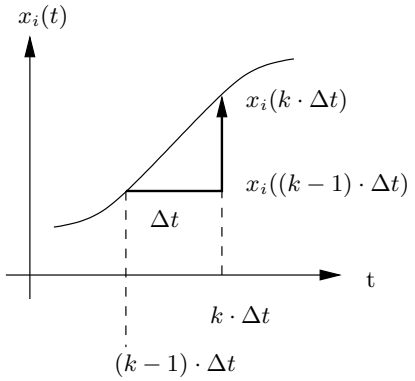


FIGURE 5.1: L'approximation de Cauchy

A l'aide de (5.2) on peut réécrire les équations différentielles d'état, à l'aide des conventions d'écriture suivante :

$$\left.\begin{array}{l} u_1(k \cdot \Delta t) = u_1(k) \\ \vdots \\ u_p(k \cdot \Delta t) = u_p(k) \\ x_1(k \cdot \Delta t) = x_1(k) \\ \vdots \\ x_n(k \cdot \Delta t) = x_n(k) \end{array}\right\} \tag{5.4}$$

Le système d'équation (5.1) continu se réécrit alors en échantillonné :

$$\left.\begin{array}{lcl} \dfrac{x_1(k) - x_1(k-1)}{\Delta t} & = & f_1(x_1(k-1) \ldots x_n(k-1); u_1(k-1) \ldots u_p(k-1)) \\ & \vdots & \\ \dfrac{x_{n,k} - x_n(k-1)}{\Delta t} & = & f_n(x_1(k-1) \ldots x_n(k-1); u_1(k-1) \ldots u_p(k-1)) \\ v_1(k) & = & g_1(x_1(k) \ldots x_n(k); u_1(k) \ldots u_p(k)) \\ & \vdots & \\ v_q(k) & = & g_q(x_1(k) \ldots x_n(k); u_1(k) \ldots u_p(k)) \end{array}\right\} \tag{5.5}$$

De l'écriture de (5.5) on tire une forme généralisée applicable à tous les type d'opérateurs de la bibliothèque libsim :

$$\left.\begin{array}{lcl} x_i(k) & = & x_i(k-1) + \Delta t \cdot f_i(x_1(k-1) \ldots x_n(k-1); u_1(k-1) \ldots u_p(k-1)) \\ & & i \in [1, n] \\ v_j(k-1) & = & g_j(x_1(k-1) \ldots x_n(k-1); u_1(k-1) \ldots u_p(k-1)) \\ & & j \in [1, q] \end{array}\right\} \tag{5.6}$$

le Δt définit le pas de calcul, et pour la suite des explications concernant les opérateurs de libsim, il sera noté T_e où temps inter-échantillons.

Choix de T_e : en pratique, T_e est positionné petit au départ, puis progressivement augmenté. Quand les résultats des simulations commencent à se dégrader, c'est qu'on tombe sur le phénomène du repliement de spectre. On a alors trouvé la limite sous laquelle ne pas aller. De manière usuelle, $10 \cdot T_{\min}$ avec $T_{\min}$ comme plus petite constante de temps du système étudié est un bon compromis ...

Ordre de calcul : une itération de calcul va consister dans un premier temps à calculer tous les $x_j(k)$ (équ. (5.6)), et ensuite seulement tous les $f_i(x_1(k)\dots x_n(k); u_1(k)\dots u_p(k))$ et $g_j(x_1(k)\dots x_n(k); u_1(k)\dots u_p(k))$ pour l'itération suivante.

5.3 Principe de simulation d'une machine d'état séquentielle

Le langage « oS » permet également le traitement de flux de données en exploitant les règles de l'algèbre de *Boole*, et la logique séquentielle sous forme de machines d'état. On note $s_{[1,r]}$ les sorties, $u_{[1,q]}$ les entrées, $e_{[1,n]}$ les états séquentiels. Les indices r et q représentent respectivement le nombre de sorties et d'entrées, et n le nombre d'états séquentiels.

L'algèbre de *Boole* fait intervenir les opérateurs logique « et » « ou » et « non » sur des entrées. En « oS » une entrée étant par nature analogique, son état est interprété dans les équations binaires comme nulle (donc fausse), ou différente de 0 (donc vraie). L'inversion d'une entrée (opération logique « non ») sera donc vraie (=1) si l'entrée est nulle, et fausse (=0) si l'entrée est vraie (non nulle).

Un « et » logique utilise l'opérateur produit, et son traitement est identique, que l'on soit en mode « analogique » ou « numérique » du point de vue des données.

Un « ou » logique utilise l'opérateur somme, et sont traitement est également identique, que l'on soit en mode « analogique » ou « numérique ».

Ce mode de fonctionnement mixte mêlant « l'analogique » et le « numérique » présente l'avantage d'une unification des processus de traitement, et permet facilement de combiner des processus à évènement discrets (les machines d'état) et les processus continus dynamiques (les équations d'état).

Les équations combinatoires peuvent toujours être définies sous forme de sommes de produits des entrées, ou de produits de sommes. Elles sont synthétisées à partir des opérateurs définis précédemment.

La logique séquentielle fait intervenir le temps dans les équations combinatoires. En électronique numérique, les circuits classiquement dévolus à ce type de tâche sont des bascules (RS, D, T, JK, ...) synchrones cadencées par une horloge.

En « oS » le rôle de l'horloge synchrone est tenu par les itérations successives de calculs. La fonction de mémorisation d'une bascule (par exemple la bascule D) est tenue par l'opérateur de transformée en « Z ». Ce mode de fonctionnement quelque peu baroque permet d'unifier le traitement du flux de données, et de pouvoir mixer les processus à évènement discrets (les machines d'état), les processus continus dynamiques (les équations d'état), et les filtres numériques (à base de transformées en « Z »). Une machine d'états numérique sera donc structurée comme suit :

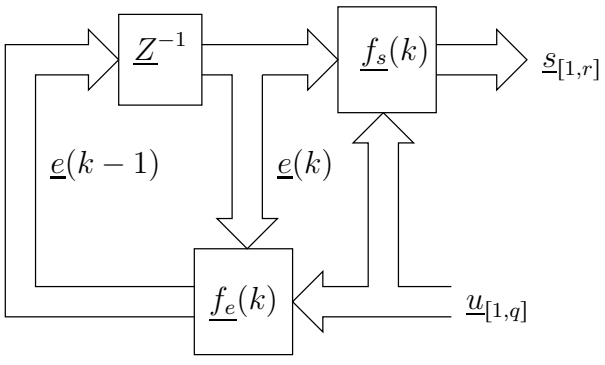


FIGURE 5.2: Structure d'une machine d'état en langage « oS »

avec les notations suivantes :

- $\underline{e}(k-1) = \underline{f_e}\,(\underline{e}(k-1), \underline{u}(k-1))$ les équations d'état ;
- $\underline{s}(k) = \underline{f_s}\,(\underline{e}(k), \underline{u}(k))$ les équations de sorties ;
- $\underline{Z}^{-1}$ les n fonctions mémoires de la machine d'état.

5.4 Quelques règles générales du langage « oS »

5.4.1 Les mots clefs

Si « oS » permet de s'abstraire des classiques variables des langages procéduraux, il n'en demeure pas moins vrai qu'il reste nécessaire de donner un nom aux objets que l'on veut décrire. Ceci signifie qu'il faut éviter d'utiliser un certain nombre de mots clefs utilisés par le compilateur « os2c »

```
input, include, object, link, main, type, set, compute, rewind, class, for, extern,
VALUE, FILE_IN, FILE_OUT, CONST, ADDER, INTEGRATOR, DERIVATIVE, ZM1, LINK_IN,
POWER, MULTIPLIER, DIVIDER, SQRT, CUSTOM, RANDOM, ABSOLUTE, LOG2, LOG10,
DTIME, L1P, L2PH, L3PH, SATH, DZH, SIN, ASIN, COS, ACOS, NOT, STOP,
TANG, ATANG, SINH, ASINH, COSH, ACOSH, TANH, ATANH, MAX, MIN, PROCESS_IN, PROCESS_OUT.
```

5.4.2 Les noms de variables

En « oS » il est tenu compte des minuscules et des majuscules. La longueur des mots n'a aucune importance. Un nom d'objet doit commencer par une lettre. Pour la suite du nom, les lettres majuscules ou minuscules, les chiffres, les caractères opérateurs

```
& ! ~ < > = _ . \ ^ + - * / ( )
```

sont autorisés. Tout le reste est à éviter, car utilisé où utilisable à l'avenir dans les fondements syntaxiques du langage « oS ».

5.4.3 Les chaînes de caractères

En « oS » il est parfois souhaitable, entre autre pour la gestion des boucles (§ 5.9 page 61) et des éléments vectorisés de définir une chaîne de caractère. Une chaîne de caractère peut se constituer comme un nom de variable, ou commencer par un caractère opérateur (§ 5.4.2), et se continuer soit par des lettres majuscules, minuscules, des chiffres. En « oS » une chaîne de caractère est un outil permettant de référencer soit des noms de fichiers de données, soit d'indicer des noms d'éléments (§ 5.8.2 page 59).

5.4.4 Les paramètres numériques

Le langage « oS » gère les valeurs numériques entières ou réelles. Il est possible de référencer une valeur par un nom symbolique. Cette opération d'assignation est effectuée dans un fichier d'en-tête (§ 5.5 page 55), ou directement au début du programme source « oS ». La définition syntaxique et grammaticale de l'opération d'assignation symbolique en langage « oS » est proche de celle du langage « C »

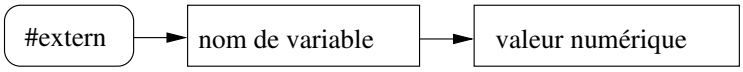


FIGURE 5.3: Assignation symbole-valeur numérique

Le mot clef « #extern » précède le symbole textuel, qui lui-même précède la variable numérique entière ou réelle redéfinie. Un exemple de définitions symboliques :

```
#extern PLUS   1     // entier: le plus d'un additionneur
#extern MOINS -1     // entier: le moins d'un additionneur
#extern Te     0.05  // reel: l'intervalle d'echantillonnage
#extern CTRL   0     // entier: controle de fin de calcul sur fin de fichier
#extern CHAMP0 0     // entier: champ 0 d'un fichier d'entrée
#extern CHAMP1 1     // entier: champ 1 d'un fichier d'entrée
```

Le compilateur « os2c » référence les définitions symboliques, et transtype en interne les variables entières en variables réelles. Au niveau du code « C » généré par « os2c » les noms symboliques utilisés dans le source « oS » sont remplacés par une recherche dans une liste de paramètres.
Pour toute la suite du document, quand on parlera de paramètre entier ou réel, on sous entendra la valeur numérique où son nom symbolique assigné dans un fichier d'en-tête.
Une grosse différence existe entre les définitions de niveau « oS » et de niveau « C ». En « oS » on peut également définir de façon symbolique des tableaux de paramètres. Ceci prend par exemple la forme suivante :

```
#extern table[0][0] 1
#extern table[0][1] 2
#extern table[1][0] 3
#extern table[1][1] 4
```

Cette possibilité n'existe pas en langage « C » et pourra être utilement mise à profit plus tard, quand on voudra exploiter les possibilités d'indiçage dans des boucles (§ 5.9 page 61).
Les définitions symboliques de niveau « os2c » se traduisent après compilation d'un fichier source « oS » par la création en sortie d'un fichier table de symbole « symbolic.tab » en plus du code « C » et du code documentaire « .tex ». Cette table des symboles est utilisable comme fichier de paramètres au moment du lancement d'un fichier binaire exécutable s'il a été lié avec la bibliothèque « libsim » et qu'il fait appel aux notions de boucles et aux définitions symboliques des tableaux de paramètres. Ce fichier de paramètres prend alors la forme suivante :

```
table[0][0] 1.000000e+00
table[0][1] 2.000000e+00
table[1][0] 3.000000e+00
table[1][1] 4.000000e+00
```

Chaque définition textuelle est associée sur la même ligne à sa valeur numérique réelle. Entre les deux, on trouve un caractère tabulation comme séparateur, et une fin de ligne après le nombre. L'utilisation dans un fichier binaire d'une table des symboles est un moyen simple et pratique de dynamiquement paramétrer un programme sans avoir à le recompiler. On peut ainsi modifier le comportement de ce dernier, sans modifier son contenu. Une table des symboles peut être construite à la main ou par tout autre programme.

5.4.5 Les commentaires, blancs et tabulations

La syntaxe des commentaires suit la même syntaxe que pour le langage « C ». Le compilateur offre également la possibilité d'écrire des commentaires documentaires. Il s'agit d'une possibilité de documentation de type code source LaTeXcomprise dans le fichier source « oS ». Ceci permet d'écrire un source « oS » et la documentation correspondante en une seule fois. La portion de source LaTeXcommence par /* et se termine par */. Elle peut être codée sur plusieurs lignes ou pages, et est extraite par le préprocesseur du compilateur « os2c ». Le résultat obtenu est ensuite compilable par LaTeXséparément. Une portion de code source documentaire s'écrit :

```
/*
Ceci est un commentaire compilable par LaTeX
ecrit sur plusieurs lignes ...
*/
```

5.5 Les fichiers « d'en-tête »

Il y a deux méthode pour incorporer des fichiers de définitions externes dans un fichier source « oS »
- les inclusions de définitions traitées au niveau du compilateur « C »
- Les inclusions de fichiers traitées spécifiquement par le compilateur « os2c ».

5.5.1 L'inclusion de niveau « C »

L'inclusion « d'en-têtes » de niveau « C » est définie hors des objets ou du corps de programme « oS ». Il s'agit d'inclure dans le code « C » généré par le compilateur « os2c » tous les en-têtes « *.h » de définitions symboliques utiles à la compilation « C ».
Il peut s'agir d'en-têtes de fonctions déjà compilés par « os2c » pour d'autres fichiers sources « oS », éventuellement des bouts de codes sources « C » définissant les fonctions utiles à l'application finale.
Dans ce type d'inclusion, il est à noter que « os2c » n'effectue aucune analyse du contenu des fichiers inclus. Cette inclusion de niveau « C » répond à la syntaxe et à la grammaire suivante :

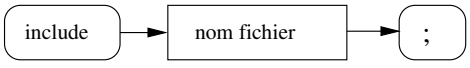


FIGURE 5.4: Inclusion de niveau « C »

Le nom de fichier à inclure est précédé du mot clef « include » et suivi par un ';'. Exemple type d'inclusions de niveau « C »

```
include header.h;              // un fichier de definitions
include sources_fonctions.c;   // des sources C personnalises
```

5.5.2 L'inclusion de niveau « oS »

Dans un fichier source en langage « oS » on peut inclure d'autres fichiers dont le contenu est inclus dans l'analyse de « os2c ». Ceci implique que les fichiers inclus doivent répondre à une syntaxe et une grammaire de langage « oS ».
Il est ainsi possible d'inclure des fichiers de définitions symboliques « *.oh » ou même un autre bout de code source « oS » contenant des définitions de classes et sous classes (§ 5.8 page 58). Ce principe d'inclusion est identique à un « coupé-collé » de fichier source. Cette inclusion de niveau « oS » répond à la syntaxe et à la grammaire suivante :

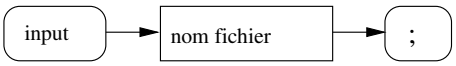


FIGURE 5.5: Inclusion de niveau « oS »

Le nom de fichier inclus est précédé du mot clef « input » et suivi d'un point virgule.

5.6 L'écriture de code « C » en ligne

En tout point d'un source « oS » il est possible d'inclure des portions de source « C ». Ceci est utile par exemple pour appeler des fonctions personnalisables d'éléments, déclarer des variables, allouer des ressources mémoires, ou même coder des fonctions entières de traitement. Deux possibilités existent : l'inclusion mono ou multi-ligne.

5.6.1 Inclure une ligne de code « C »

Pour écrire une très courte portion de code « C » on peut utiliser l'inclusion mono-ligne : une portion de code « C » terminée par une fin de ligne est précédée du caractère % :

```
% printf( "ceci est une portion de code C en ligne\n" );
```

la partie suivant le caractère % est recopiée textuellement dans le code « C » généré par « os2c » y compris le caractère de fin de ligne. La ligne suivante est considérée comme étant à nouveau du « oS ».

5.6.2 Basculer en mode de programmation « C »

Il peut être parfois utile d'écrire des portions importantes de code « C » dans un source « oS » en particulier pour les fonctions personnalisables d'éléments non standards, déclarer des variables, allouer des ressources mémoires. Dans de tels cas de figure, il est intéressant de pouvoir « basculer » du mode de programmation « oS » au « C » et vice-versa. Un tel basculement est obtenu quand le préprocesseur de « os2c » rencontre les caractères %%. Par exemple :

```
%% // basculement en mode de programmation "C"

 printf( "ceci est une portion de code C\n" );
 printf( "multiligne\n\n" );

%% // basculement en mode de programmation "oS"
```

Le code « C » compris entre %% est intégralement recopié dans le code « C » généré par « os2c » y compris les caractères de fin de ligne.

5.7 Structure de « libsim » quelques notions générales

Le but de cette section est de donner au lecteur une vue globale du fonctionnement de la bibliothèque libsim. Cette bibliothèque est le cœur de toute application développée en langage « oS ». Son mode de fonctionnement conditionne quelque peu la structure syntaxique et grammaticale du langage « oS ». Les quelques notions qui suivent permettent de faire le lien entre le code « C » généré par le compilateur « os2c » et le langage « oS ».

5.7.1 Qu'est-ce qu'un élément de traitement

Un élément de traitement ou de calcul est une petite entité qu'on spécialise en lui attribuant un type (additionneur, diviseur...). On le paramètre spécifiquement en fonction de ce type, et on peut le lier avec d'autre éléments.
Cette entité possède toujours un champ résultat qui contient en permanence le résultat du dernier calcul effectué, ou une valeur initiale qui lui a été affecté.
Le champ résultat d'un élément peut être lié à une ou plusieurs entrées d'autres éléments.
Une entrée d'un élément ne peut être reliée qu'une seule fois à un seul champ résultat d'un autre élément.
La fonction de calcul attribuée à un élément prend en compte comme entrées les champs résultats d'autres éléments, ainsi que d'éventuels paramètres attribués à l'élément.
Un objet est une ensemble de plusieurs éléments en dépendances. Des fonctions de lien inter éléments permettent toutes les combinaisons entre opérateurs mathématiques à l'intérieur d'un objet.
Le noyau de simulation libsim permet la gestion d'un nombre quelconque d'éléments à l'intérieur d'un objet.
Un élément est désigné par un nom unique seulement à l'intérieur de son objet.
L'entrée d'un élément ne peut être liée qu'une seule fois, et ne doit jamais demeurer inutilisée.

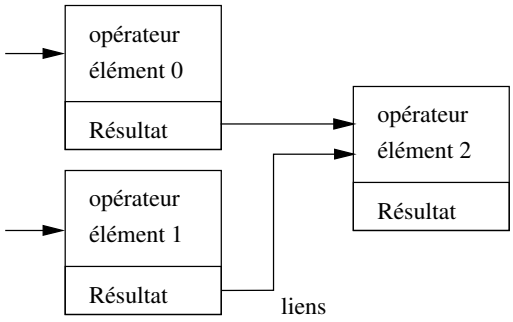


FIGURE 5.6: Chaînage d'éléments

5.7.2 Qu'est-ce qu'un objet

Une fois compilé, un objet est une association d'éléments mutuellement chaînés, et déclarés par une fonction « C » faisant appel aux ressources de la bibliothèque « libsim ». Le nom de l'objet est de fait le nom de la fonction « C » déclarative de l'objet. Un objet forme donc une entité mathématique complexe (plusieurs opérateurs en dépendances) formant une équation par exemple.

On peut définir un nombre quelconque d'objets. Des fonctions de liens inter objets permettent des associations quelconques (objets reliés en série, en parallèle ...), et permettent le transit du flux de données.
Un objet contient N éléments, qui vont être typés à l'intérieur de l'objet. Chaque élément de cet objet est référencé par son nom intra objet.
Un élément d'un objet peut avoir des homonymes existant dans d'autres objets :

- nom-objet-0 éléments e0, e1, e2...
- nom-objet-1 éléments e0, e1, e2...
- ...
- nom-objet-N éléments e0, e1, e2...

Une application du langage de simulation sera donc une succession d'objets déclarés et décrits les uns derrière les autres, suivis d'un corps de programme tout à fait à la fin (un peu comme en langage Pascal).
L'ordre de déclaration et d'initialisation des objets ainsi que l'ordre de déclaration et de typage des éléments à l'intérieur d'un objet dans le source « oS » détermine ultérieurement l'ordre d'exécution des calculs.

Un objet est en principe vu « de l'extérieur » comme une boîte noire comportant des points d'entrée prévus pour être connectés à des éléments constitutifs d'un autre objet. Le champ résultat de n'importe quel élément constitutif d'un objet peut être lié au point d'entrée d'un autre objet. On appelle ceci une liaison inter objets. Il n'y a qu'à travers ces canaux de communication qu'une liaison entre plusieurs objets est possible. Ceci autorise l'association d'objets spécialisés, *via* ces liens. Le développement d'une application peut ainsi être grandement simplifié, car segmenté en entités indépendante.

5.7.3 Le corps de programme

Le langage « oS » est un langage déclaratif en ce sens que la plus grosse partie du code consiste à déclarer des objets, des éléments, et les liens qui les unissent l'ensemble. Toute cette phase n'est exécutée qu'une fois, à l'initialisation d'une application. Ensuite, on sort du mode déclaratif pour entrer en mode exécutif. Ce mode n'est plus consommateur en code exécutable, mais en ressource CPU. En effet, dans ce mode, on saute dans l'unique fonction de calcul de la libsim, et l'on n'en ressort qu'une fois le travail effectué : calcul d'un nombre fixé d'itérations, fin de fichiers en mode lecture, stop asynchrone sur évènement externe ...
Un corps de programme en langage « oS » est unique, constitue le point d'entrée du programme, et remplit trois fonctions principales :

— déclarer les objets ;
— définir une liste de calcul ;
— lancer et gérer les passes successives de calculs.

5.8 Les classes

Une classe en langage « oS » est un ensemble de définitions d'éléments, et des liens qui les unissent. Il s'agit essentiellement d'une liste de déclarations générique, utilisable à loisir plus tard comme constituant d'un objet. Une classe en soi n'a pas d'existence tangible au niveau du code « C ». Sa matérialisation se fait au moment de la déclaration d'un objet. Une classe peut en quelque sorte être comparée à une « liste de courses » avant d'aller faire son marché !
L'intérêt principal du concept de classe vient de la possibilité d'emboîter les classes (notion de sous classes et d'héritage), de pouvoir les dupliquer à l'envie (notion d'héritage vectorisé), et d'avoir la possibilité à tout moment de paramétrer spécifiquement des éléments (notion de surcharge paramétrique).
Les classes en langage « oS » sont le gage d'une réutilisation possible d'associations standards d'éléments et de liens, sans avoir à réécrire constamment de longues portions de code source. C'est la porte ouverte à l'écriture de bibliothèques standards d'opérations sur des flux de données, qu'il est possible d'incorporer dans des objets quand le besoin de l'utilisateur s'en fait sentir.

5.8.1 La classe élémentaire

Une classe, au sens le plus simple possible en « oS », est définie par son nom, la déclaration d'un certain nombre d'éléments, et des liens qui les unissent. Le canevas syntaxique et grammatical est défini par la FIGURE 5.7 page 58 :

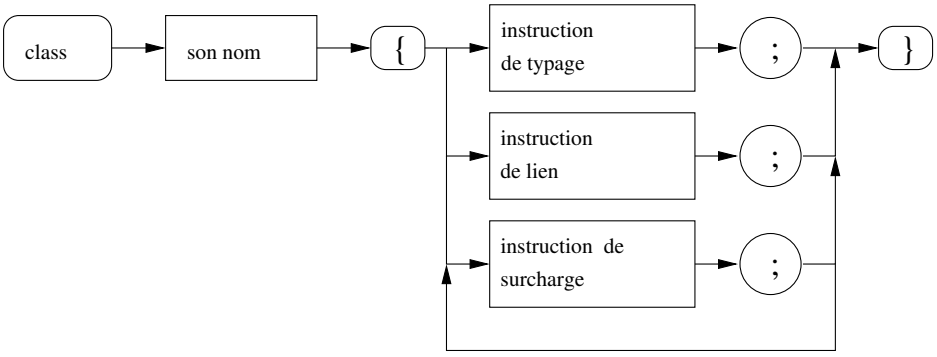


FIGURE 5.7: Grammaire et syntaxe d'une classe simple

Exemple type de déclaration d'une classe élémentaire :

```
classe nom_classe
{    // accolade ouvrante, debut de la liste des instructions
     // les instructions de typage et de lien sont positionnees ici
}    // fin de la liste des instructions
```

trois instructions constituent principalement la liste en accolades :
- typer (déclarer et définir) un élément de traitement dans la classe ;
- lier deux éléments de la classe ;
- reparamétrer (surcharger) un élément déjà typé.

Une instruction est constituée généralement d'un mot clef permettant de la différentier des autres instructions, d'une suite d'arguments et de paramètres, et se termine par un point virgule.

5.8.2 Grammaire standard d'un typage d'élément

Un élément se type et se paramètre à l'aide d'une seule instruction, illustrée par la figure 5.8 page 59 :

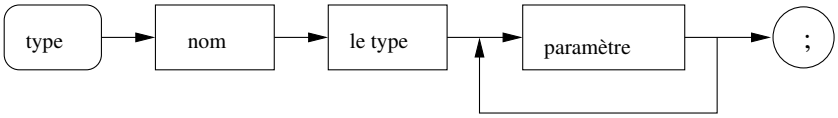


FIGURE 5.8: Grammaire et syntaxe d'un typage d'élément

Une telle instruction commence par le mot clef « type », suivi du nom de l'élément que l'on souhaite typer. Viennent ensuite si nécessaire, des paramètres numériques, ou chaînes de caractères. L'instruction se termine par un point virgule.
Il est à noter pour l'ensemble des éléments typés, que les champs résultats sont initialisés à 0 au moment de leur matérialisation ultérieure par un objet, et que l'ordre d'apparition dans la liste détermine l'ordre de passage au moment d'une itération de calcul.

Les éléments niladiques

On range dans cette catégorie tous les éléments typés couvrant des opérateurs sans opérande, mais délivrant une ou plusieurs informations de sorties. On trouve dans cette catégorie les types suivants :
- FILE_IN lecture d'une ligne d'opérande(s) dans un fichier ASCII source
- PROCESS_IN encapsuler un processus qui lira en entrée un flux d'une ligne d'opérande(s)
- LINK_IN point d'entrée d'un objet
- RANDOM générateur de bruit blanc à bande passante limitée
- STOP gestion de fin de calcul asynchrone sur évènement externe
- VALUE champ résultat prenant une valeur constante

Les éléments monadiques

On range dans cette catégorie tous les éléments typés couvrant des opérateurs à deux opérandes d'entrées, et délivrant une information de sortie. On trouve dans cette catégorie les types :
- ABSOLUTE effectuer la valeur absolue d'une valeur d'entrée
- ACOS effectuer l'arc cosinus d'une valeur d'entrée
- ACOSH effectuer l'arc cosinus hyperbolique d'une valeur d'entrée
- ASIN effectuer l'arc sinus d'une valeur d'entrée
- ASINH effectuer l'arc sinus hyperbolique d'une valeur d'entrée
- ATANG effectuer l'arc tangente d'une valeur d'entrée
- ATANH effectuer l'arc tangente hyperbolique d'une valeur d'entrée
- CONST multiplier une valeur d'entrée par une constante
- COS effectuer le cosinus d'une valeur d'entrée
- COSH effectuer le cosinus hyperbolique d'une valeur d'entrée
- DERIVATIVE dériver temporellement le signal d'entrée à l'aide d'une différence finie
- DTIME appliquer un décalage temporel fixe sur une entrée
- DZH appliquer une fonction zone morte avec ou sans hystérésis sur une entrée
- INTEGRATOR intégrer temporellement le signal d'entrée
- LOG2 calculer le logarithme népérien d'une entrée

- LOG10 calculer le logarithme décimal d'une entrée
- L1P appliquer un seuillage homothétique sur une entrée
- L2PH appliquer un seuillage 2 points avec ou sans hystérésis à une entrée
- L3PH appliquer un seuillage 3 points avec ou sans hystérésis à une entrée
- NOT effectuer la négation booléenne d'une valeur d'entrée
- SATH appliquer une limitation avec ou sans hystérésis à une entrée
- SIN calculer le sinus d'une valeur d'entrée
- SINH calculer le sinus hyperbolique d'une valeur d'entrée
- SQRT calculer la racine carrée d'une valeur d'entrée
- TANG calculer la tangente d'une valeur d'entrée
- TANH calculer la tangente hyperbolique d'une valeur d'entrée
- ZM1 ligne à retard unitaire (une itération de calcul, de valeur temporelle T_e) sur une entrée

Les éléments diadiques

On range dans cette catégorie tous les éléments typés couvrant des opérateurs à deux opérandes d'entrées, et délivrant une information de sortie. On trouve dans cette catégorie les types :

- DIVIDER diviser 2 valeurs provenant de deux valeurs d'entrées
- WATCHDOG gérer une temporisation à l'aide de deux entrées de contrôle
- POWER calcul de la puissance $S = x^y$, x et y étant deux entrées distinctes de l'élément

Les éléments multi-adiques

On range dans cette catégorie tous les éléments typés couvrant des opérateurs à n opérandes d'entrées, et délivrant une information de sortie. On trouve dans cette catégorie les types :

- ADDER Additionner ou soustraire selon le paramétrage n valeurs d'entrées, et positionner le champ résultat avec : $S = \sum_{i=1}^{n} E_i K_i$ avec $K = \pm 1$, n n'étant pas limité
- CUSTOM ce type spécial est prévu pour interfacer du code « C » personnalisé permettant de résoudre des opérateurs non couverts par les types standards du code « oS ».
- FILE_OUT lecture simultanée des n liens d'entrées, et rangement dans une ligne texte d'un fichier de sortie
- PROCESS_OUT lecture simultanée des n liens d'entrées, et rangement dans une ligne texte fournie à un processus parallèle encapsulé par « os2c »
- MAX déterminer le maximum de n valeurs d'entrées, et positionner le champ résultat avec : $S = max(E_i) \quad i = 1 \dots n$
- MIN déterminer le minimum de n valeurs d'entrées, et positionner le champ résultat avec : $S = min(E_i) \quad i = 1 \dots n.$
- MULTIPLIER multiplier ensemble n entrées, et positionner le champ résultat avec : $S = \prod_{i=1}^{n} E_i \quad i = 1 \dots n.$

5.8.3 Grammaire standard d'un lien intra-classe

Une instruction de ce type permet de lier le champ résultat d'un élément (considéré comme la source du lien), à l'entrée d'un élément destination. Un tel lien se paramètre à l'aide d'une seule instruction :

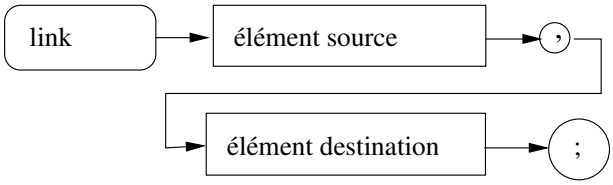


FIGURE 5.9: Grammaire standard d'un lien intra-classe

Exemple d'utilisation de cette instruction :

```
link e0, e1; // lien intra-classe entre le champ resultat de e0
             // et une entree de e1
```

La grammaire précédente permet de lier tous les types d'éléments, sauf les éléments de type « FILE_IN », « FILE_IN » et « CUSTOM » qui sont conçus pour gérer des champs résultats multiples. Chacun de ces champs de sortie possédant un numéro unique, l'instruction « link » permet d'en tenir compte. La grammaire et la syntaxe devient :

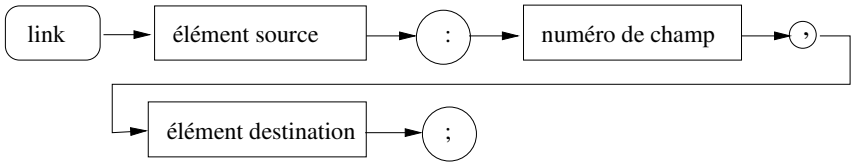


FIGURE 5.10: Grammaire standard d'un lien intra-classe à champ multiple

Exemple d'utilisation dans ce cas de figure :

```
link e0:2, e1; // lien intra-classe entre le champ resultat 2 de l'élément e0
               // et une entree de e1
```

5.8.4 Grammaire standard d'une surcharge paramétrique

La surcharge paramétrique est un outil permettant à l'utilisateur de modifier les valeurs de paramètres de certains éléments déjà déclarés par une instruction « type ». Les éléments en question peuvent êtres de type : « CONST », « VALUE », « DTIME », « RANDOM », « L1P », « L2PH », « L3PH », « SATH », « DZH », « INTEGRATOR ». Les éléments d'un autre type ne se prêtent pas à une surcharge. La grammaire d'une surcharge est rigoureusement identique à celle de l'instruction « type » ; seul le mot clef est différent.

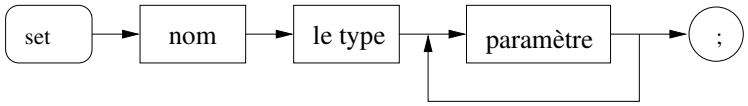


FIGURE 5.11: Grammaire standard d'une surcharge paramétrique

Syntaxe « oS » exacte :

```
 set e3         // le nom de l'element
     INTEGRATOR // le type
     0.0        // 0.0 borne inferieure
     0.0        // 0.0 borne superieure
     0.5;       // 0.0 valeur initiale, remplacee par 0.5
```

5.9 Les boucles « for » dans une classe

5.9.1 Structure d'une boucle « for »

Une instruction de typage ne permet de déclarer qu'un élément à la fois. De même, une instruction de lien de permet pas d'établir plus d'un lien en même temps, et une surcharge paramétrique ne modifie qu'un seul élément à la fois. Pourtant, il est souvent utile de vectoriser la création d'un certain nombre d'éléments du même type. De même, il peut être utile de pouvoir lier « en bloc » un vecteur d'éléments à un fichier par exemple, pour pouvoir multiplexer les données au moment du stockage. Une initialisation « en bloc » d'un vecteur d'éléments peut par exemple être utile pour positionner des conditions initiales. Dans de telles situations, la notion de boucle est alors la bienvenue.

L'idée consiste à pouvoir déclarer, surcharger ou lier des éléments « en boucle ». Cette façon de procéder est très économe en écriture et en code « C » généré quand elle est utilisée à bon escient. Pour bien comprendre, partons d'un exemple :

```
#define BORNE_DEBUT 0
#define BORNE_FIN   2
```

```
for j,BORNE_DEBUT,BORNE_FIN {
   // le corps de boucle
   %printf( "compteur %d\n", j );
}
```

Les deux premières lignes définissent deux paramètres numériques (§ 5.4.4 page 54) pour indiquer respectivement la borne départ et fin de comptage de la boucle.

Le mot clef « for » est la convention de déclaration de début d'une structure grammaticale et lexicale de boucle. Suit un nom de variable. Une telle variable peut être considérée comme muette au sens mathématique du terme. Elle est muette, car elle n'a d'existence que pendant la phase déclarative à l'intérieur d'une classe ou d'un objet. Cette variable n'est rien d'autre qu'un indice de comptage. Au niveau d'analyse du langage « oS », cette variable est considérée comme une chaîne de caractère (§ 5.4.3 page 54). **De ce fait, elle ne doit pas comporter de blancs ou de tabulations**. Le traitement proprement dit de cette chaîne de caractère est laissé au soin du compilateur « C », qui va transcrire la variable chaîne en variable entière utilisable. Le langage « oS » répertorie en interne les variables muettes d'un objet dès qu'il en est fait mention, sans que l'utilisateur ait à s'en soucier.

Après la variable muette, vient une variable numérique d'initialisation de comptage, suivi d'une limite supérieure de fin de comptage. Deux virgules séparent la variable muette, et les deux paramètres de comptage. Une accolade ouvrante et une accolade fermante marquent le corps de la boucle. Leur présence est obligatoire.

Pour les boucles, on utilise en « oS » une méthodologie de comptage similaire à celle du « C » : on tourne dans la boucle tant que la variable de comptage est inférieure à la borne de fin. La variable de comptage est augmentée de un à chaque tour. En « oS », une boucle est de nature pré-testée : dans notre exemple, si « BORNE_DEBUT » et « BORNE_FIN » ont la même valeur, l'intérieur du corps de boucle n'est pas exécuté.

La traduction en langage « C » de l'exemple précédent donnerait :

```
int j;

for ( j=0; j < 2 ; j++ ) {
   printf( "compteur %d\n", j );
}
```

En « oS », il est possible d'imbriquer autant de boucles qu'on le souhaite. L'exemple suivant illustre deux boucles imbriquées :

```
#define BORNE_DEBUT 0
#define BORNE_FIN   2

for j,BORNE_DEBUT,BORNE_FIN {
   for k,BORNE_DEBUT,BORNE_FIN {
      // le corps de boucle
      %printf( "compteur %d %d\n", j, k );
   }
}
```

Le nom de variable muette 'i' est prohibé. C'est un grand classique des programmeurs, utilisé de façon systématique au niveau de la génération d'un objet.

5.9.2 Indiçage dans une boucle « for »

Le nom d'un élément déclaré à l'aide d'un instruction « type » est unique. Il est donc indispensable pour une instruction « type » que l'on souhaite insérer dans le corps d'une boucle « for » de posséder un mécanisme d'indiçage des noms. C'est ici que l'utilisation des variables muettes assimilées à des chaînes de caractères prend tout son sens.
Le langage « oS » détecte un indiçage quand une variable numérique ou une chaîne de caractère est encadrée entre un crochet ouvrant ou fermant. nom[0], nom[ZERO], nom[j] sont considérés comme étant des variables indicées :

- nom[0] est un nom d'élément déclaré tel quel au niveau du code « C » final.
- nom[ZERO], avec « ZERO » assimilé à une redéfinition symbolique de niveau « oS »
  `#define ZERO 0 // déclaration au debut du source "oS"`
  Dans ce cas, ZERO est traduit par « os2c » en variable entière 0, et nom[0] est le nom d'élément généré au niveau du code « C » final.

— Pour nom[ZERO], avec « ZERO » assimilé à une redéfinition symbolique de niveau « C », ou dans un cas de figure comme nom[j], alors « os2c » doit recourir à la reconstitution d'un nom d'élément indicé, effectué en trois temps. « os2c » considère tout ce qui se trouve entre crochets comme une chaîne de caractères (dans l'exemple, « ZERO » ou « j »). Cette dernière est alors recopiée comme argument tel quel dans le code « C » généré. A la compilation « C » de ce code, la chaîne est interprétée comme un traitement numérique sur des entiers. A l'exécution du code binaire, le calcul d'indices entiers est effectué. Le résultat est converti en ASCII entre crochet ouvrant et fermant, et fusionné avec la chaîne de caractère précédent les crochets (dans notre exemple « nom »).

On comprend ainsi l'intérêt d'assimiler les symboles opérateurs « C »

```
& ! ~ < > = _ . \ ^ + - * / ( )
```

comme des constituants possibles de chaînes de caractères. « oS » les considère comme des caractères, par contre, le compilateur « C » les traitera comme des opérateurs à son niveau. De ce fait, il est conseillé pour l'utilisateur d'user de parenthèses sans modérations pour décrire ses indices ...

Le langage « oS » n'est pas limitatif sur le nombre d'indices entre crochets. L'utilisateur peut les agréger autant qu'il le souhaite, et utiliser ce mécanisme dans une boucle « for » pour les instructions « type », « link » ou « set ».

Illustrons tout ceci par l'exemple suivant :

```
#define TLIGNES 2
#define TCOLS   2

type fich-out         // le nom de l'element fichier
     FILE_OUT         // de type fichier de sortie
     matrice.dat      // le nom de fichier
     (TLIGNES*TCOLS); // nombre de champs de sorties

for j,0,TLIGNES {
    for k,0,TCOLS {
       type in-p[j][k] LINK_IN;
       link in-p[j][k], fich-out;
    }
}
```

On suppose que les définitions symboliques TLIGNES et TCOL sont définies par ailleurs au niveau « C » par un fichier de définition. Dans notre exemple, on ouvre un fichier multiplexé de (TLIGNES*TCOLS) champs de sorties. Dans le corps de la boucle « for », on déclare une matrice de TLIGNES et TCOLS éléments points d'entrées, qui sont liés et multiplexés dans le fichier « matrice.dat ». A l'exécution du binaire, les éléments déclarés à « libsim » auront pour nom :

— in-p[0][0]
— in-p[0][1]
— in-p[1][0]
— in-p[1][1]

et auront cet ordre d'apparition dans le fichier de sortie multiplexé. Il est à noter qu'ensuite, on peut manipuler un nom indicé spécifique (avec variables numériques) partout à l'intérieur d'une classe ou d'un objet.

5.9.3 Les arguments indicés

Certains éléments de traitements offrent la possibilité à l'utilisateur non seulement d'indicer leur nom, mais également certains paramètres chaînes de caractères. Il s'agit plus précisément des noms de fichiers et des champs paramètres des éléments « FILE_IN » et « FILE_OUT », les n° de champ de « CUSTOM » et « FILE_IN » en cas de liens, et les paramètres entiers ou réels des éléments de calculs : « CUSTOM », « VALUE », « CONST », « INTEGRATOR », « MULTIPLIER », « RANDOM », « DTIME », « L1P », « L2PH », « L3PH », « SATH », « DZH », « MAX », « MIN ». Ces éléments peuvent faire l'objet d'un indiçage sous forme de vecteurs, de matrices pour peu que les définitions symboliques de niveau « oS » aient été prévues (§ 5.4.4 page 54)... Prenons l'exemple suivant :

```
#define TLIGNES 2
#define TCOLS   2
```

```
for j,0,TLIGNES {
    type fich-out[j]   // le nom indice de chaque element fichier
         FILE_OUT      // de type fichier de sortie
         ligne.dat[j]  // le nom de fichier
         TCOLS;        // le nombre de champs de sorties
}
```

Dans cet exemple, on déclare deux éléments de type fichier de sortie « fich-out[0] » et « fich-out[1] », permettant d'ouvrir respectivement deux fichiers « ligne.dat[0] » et « ligne.dat[1] » comportant chacun « TCOLS » champs de sorties par ligne texte.

5.10 Les blocs « if » dans une classe

En complément des boucles « for », « os2c » possède la notion de bloc conditionnel. Ceci peut s'avérer particulièrement utile si l'on souhaite activer une liste d'action conditionnelle à l'intérieur d'une boucle « for » par exemple.

5.10.1 Structure d'un bloc « if »

Une liste d'action conditionnelle s'écrira typiquement :

```
#define TLIGNES 2
#define TCOLS   2

for j,0,TLIGNES {
  for k,0,TCOLS {
    if (j==k) {
      type cste[j][k]  // le nom indice de chaque constante
           CONST
           (j*k);      // la valeur de la constante
    }
  }
}
```

Entre le mot clef « if » et l'accolade ouvrante « { », on retrouve une condition de test, recopiée telle qu'elle au niveau du code « C » généré. Une accolade fermante « } » vient terminer la liste des actions entre accolades. L'exemple précédent consiste donc à déclarer deux éléments constantes « cste[0][0] » et « cste[1][1] », respectivement de valeur 0 et 1. La traduction « C » du bloc testé s'écrit donc :

```
    if (j==k) { // debut de bloc
    }           // fin de bloc
```

5.10.2 Structure d'un bloc « else »

On peut également associer à un bloc « if », un bloc « else » associé à une liste d'actions qui seront activées seulement si la liste d'actions associée au bloc « if » n'est pas activée. Le bloc « else » est le complémentaire du bloc « if ». Reprenons l'exemple § 5.10.1 page 64, en rajoutant le bloc « else » complémentaire :

```
#define TLIGNES 2
#define TCOLS   2

for j,0,TLIGNES {
  for k,0,TCOLS {
    if (j==k) {
      type cste[j][k]  // le nom indice de chaque constante
           CONST
           (j*k);      // la valeur de la constante
    }
    else {
```

```
      type cste[j][k] CONST (j+k);
    }
  }
}
```

Après le mot clef « else », on trouve les accolades ouvrantes et fermantes encadrant la liste des actions complémentaires du bloc « if ». La traduction « C » du bloc « else » s'écrit donc :

```
    else { // debut de bloc
    }      // fin de bloc
```

5.10.3 Structure d'un bloc « else if »

On peut également associer à un bloc « if », un bloc « else if » associé à une liste d'actions qui seront activées seulement si la liste d'actions associée au bloc « if » n'est pas activée, et que le test après les deux mots clefs successifs « else if » est valide. Reprenons l'exemple § 5.10.1 page 64, en rajoutant un nouveau bloc conditionnel « else if » en plus du bloc « if » initial :

```
#define TLIGNES 2
#define TCOLS   2

for j,0,TLIGNES {
  for k,0,TCOLS {
    if (j==k) {
      type cste[j][k]  // le nom indice de chaque constante
           CONST
           (j*k);      // la valeur de la constante
    }
    else if (j==0) {
      type cste[j][k] CONST 0;
    }
    else {
      type cste[j][k] CONST (j+k);
    }
  }
}
```

Après les mots clefs « else if » et l'accolade ouvrante « { », on retrouve une condition de test, recopiée telle qu'elle au niveau du code « C » généré. Entre les accolades ouvrantes et fermantes, on trouve ensuite la liste des actions (ici type cste[j][k] CONST 0 ;). La traduction « C » du bloc « else if » s'écrit donc :

```
    else if () { // debut de bloc
    }            // fin de bloc
```

5.11 Le mécanisme d'héritage de classe

La définition d'une classe élémentaire (§ 5.8.1 page 58) est un processus de définition standard d'une entité qui a pour vocation de pouvoir être mise en bibliothèques, et être réutilisée. Sa réutilisation suppose un mécanisme autorisant son incorporation dans un ensemble qui peut être plus important, et contenir plusieurs classes similaires ou différentes. Ce mécanisme a pour nom l'héritage de classe.

5.11.1 L'heritage simple

Le mécanisme d'héritage simple consiste à « assembler » une ou plusieurs classes distinctes dans une nouvelle classe éventuellement plus grosse. Illustrons ce propos par l'exemple suivant :

```
class cl1 { // définition de la classe elementaire cl1
  type entree-cl1 LINK_IN;
  type cste-cl1   CONST 2.0;

  link entree-cl1, cste-cl1;
}
```

La classe élémentaire « cl1 » est une entité multipliant une valeur d'entrée par 2.

```
class cl2 { // définition de la classe elementaire cl2
  type entree-cl2 LINK_IN;
  type cste-cl2   CONST 3.0;

  link entree-cl2, cste-cl2;
}
```

La classe élémentaire « cl2 » est une entité multipliant une valeur d'entrée par 3. L'incorporation de ces deux classes dans une classe plus importante se fera le plus simplement possible comme suit :

```
class capsule cl1 cl2;
```

Après le mot clef « class », suit le nom de la nouvelle classe, lui même suivi de la liste des noms de classes élémentaires à incorporer. Chaque nom est séparé par un blanc ou une tabulation, la liste est terminée par un point virgule.

Dans cet exemple, la nouvelle classe « capsule » hérite des classes « cl1 » et « cl2 ». On dit également que « cl1 » et « cl2 » sont des sous classes de la classe « capsule ». L'héritage de « cl1 » et « cl2 » se traduit au niveau du nom des éléments constitutifs de la nouvelle classe « capsule ». les noms d'éléments des classes « cl1 » et « cl2 » sont rebaptisés en précédant leur ancien nom du nom de la nouvelle classe suivi d'un point séparateur :

- capsule.entree-cl1
- capsule.cste-cl1
- capsule.entree-cl2
- capsule.cste-cl2

Les liens déjà définis dans les classes élémentaires sont préservés.

Dans l'exemple, après héritage, l'élément « capsule.entree-cl1 » est lié à « capsule.cste-cl1 », et l'élément « capsule.entree-cl2 » est lié à « capsule.cste-cl2 ». La classe nouvelle classe capsule a donc hérité des liens définis dans les classes « cl1 » et « cl2 ».

L'agrégation de plusieurs classes ne résout pas le problème de l'interaction des sous classes, ni la nécessité qui peut se faire sentir de surcharger certains paramètres. Considérons l'exemple précédent. On souhaite en plus de l'agrégation des deux sous classes lier l'élément « capsule.cste-cl1 » à l'entrée « capsule.entree-cl2 », et surcharger le paramètre « capsule.cste-cl2 » pour que ce dernier prenne la valeur 5. On écrira donc :

```
class capsule cl1 cl2 {
   link capsule.cste-cl1, capsule.entree-cl2; // lien
   set  capsule.cste-cl2  CONST 5.0;          // surcharge
}
```

En lieu et place du point virgule, apparaissent une accolade ouvrante et fermante, entre lesquels une liste d'instructions similaires à celles qui ont été étudiées aux § 5.8.1 et 5.9 peuvent prendre place.
Le canevas grammatical et syntaxique d'un héritage de classe simple est donc le suivant :

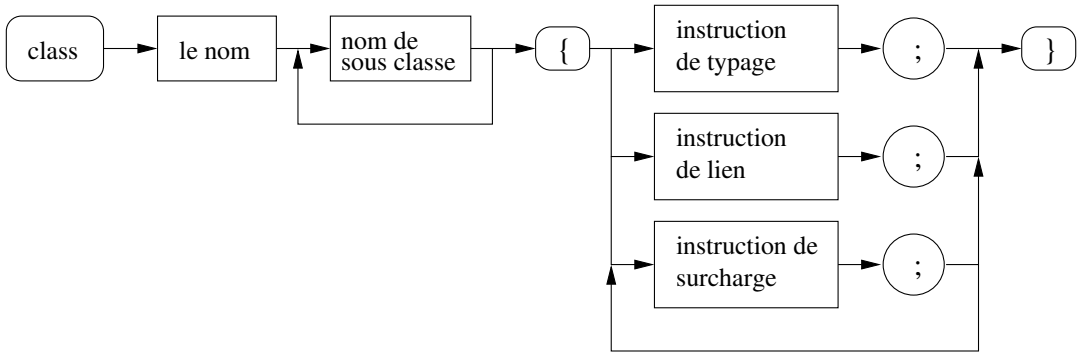


FIGURE 5.12: Grammaire et syntaxe d'un héritage de classe simple

Le langage « oS » ne comporte aucune limite quand à la taille des agrégations ou la profondeur des héritages. De ce fait, il est possible d'emboîter des classes est des sous classes comme des poupées russes, et de constituer ainsi des bibliothèques de classes. Ces dernières peuvent ensuite s'inclure dans un fichier source « oS » à l'aide d'une directive d'inclusion (§ 5.5.2 page 56).

5.11.2 L'héritage vectorisé

Le mécanisme d'héritage vectorisé est une extension de l'héritage simple. Son existence tient au fait qu'il peut être intéressant de disposer d'un outil de duplication souple d'une même classe, un nombre quelconque de fois. En effet, définir « manuellement » N classes identiques structurellement, et de nom chaque fois différent peut être fastidieux. L'héritage vectorisé existe pour contourner ce problème, et autoriser la création de vecteurs, de matrices d'une classe élémentaire.
L'idée est grammaticalement ou syntaxiquement très simple : on associe au nom d'une classe un nombre de duplications entre crochets, dans le même esprit que pour les boucles « for ». Reprenons l'exemple précédent, et cherchons à obtenir le même type de fonctionnalité en utilisant ce nouveau mécanisme :

```
#define TAILLE_VECTEUR 2

class capsule cl1[TAILLE_VECTEUR] {
   link capsule.cste-cl1[0], capsule.entree-cl1[1]; // lien
   set  capsule.cste-cl1[1]  CONST 5.0;             // surcharge
}
```

Pour le compilateur « os2c », ceci revient en interne à construire automatiquement pour la classe capsule la liste d'instructions suivantes :

```
  type capsule.entree-cl1[0]  LINK_IN;               // début du mecanisme
  type capsule.cste-cl1[0]    CONST 2.0;             // de vectorisation
  link capsule.entree-cl1[0], capsule.cste-cl1[0];
  type capsule.entree-cl1[1]  LINK_IN;
  type capsule.cste-cl1[1]    CONST 2.0;
  link capsule.entree-cl1[1], capsule.cste-cl1[1];   // fin du mecanisme
                                                     // de vectorisation

  link capsule.cste-cl1[0],   capsule.entree-cl1[1]; // lien
  set  capsule.cste-cl1[1]    CONST 5.0;             // surcharge
```

Quelques remarques s'imposent :

- la taille du vecteur spécifiée entre crochets au moment de la déclaration de classe doit l'être soit sous forme d'entier, soit par une redéfinition symbolique de niveau « oS » (§ 5.4.4 page 54). Il s'agit donc d'un paramètre numérique que « os2c » doit pouvoir interpréter, afin de pouvoir reconstituer la liste des instructions de la nouvelle classe.
- L'assignation des noms d'élément se fait automatiquement par « os2c » selon la même logique de progression que pour les boucles « for » (§ 5.9 page 61).
- L'ordre déclaratif dans la classe élémentaire est respecté. L'ordre d'exécution au moment du calcul suivra la même logique de progression que l'ordre déclaratif.
- La liste des instructions spécifique à la nouvelle classe définis entre accolade ouvrante et fermante sont prises en compte et déclarées physiquement après l'héritage vectorisé.
- Au niveau du code « C » généré, on retrouve la longue liste reconstituée en interne ...

Le langage « oS » n'impose aucune limite dans le nombre de vecteurs successifs que l'utilisateur va définir. On peut par exemple imaginer une déclaration de ce type :

```
class capsule cl1[TAILLE_VECTEUR][TAILLE_VECTEUR][TAILLE_VECTEUR];
```

Dans ce cas, pour l'élément « entree-cl1 » de la sous classe « cl1 », l'indiçage automatique suivra la progression suivante :

```
capsule.entree-cl1[0][0][0], capsule.entree-cl1[0][0][1],
capsule.entree-cl1[0][1][0], capsule.entree-cl1[0][1][1],
capsule.entree-cl1[1][0][0], capsule.entree-cl1[1][0][1],
capsule.entree-cl1[1][1][0], capsule.entree-cl1[1][1][1]
```

5.11.3 Grammaire standard d'un héritage

En résumé, le schéma grammatical et syntaxique d'un héritage de classe, simple ou vectorisé est représenté par la FIGURE 5.13 page 68 :
On peut noter que le mécanisme d'héritage décrit ici est générique, et utilisable au niveau d'une classe, mais également au niveau des objets.

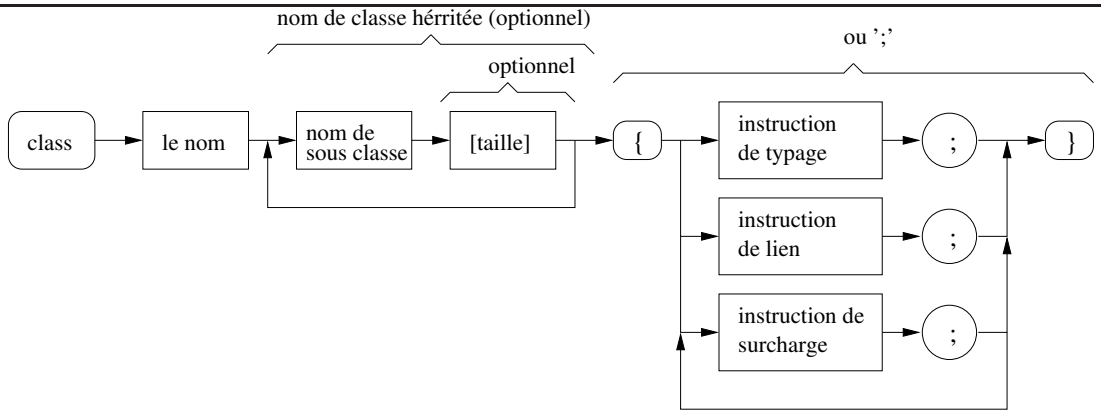


FIGURE 5.13: Grammaire et syntaxe d'un héritage avec toutes ses variantes

5.12 Les objets

Si les classes sont des entités uniquement déclaratives et « virtuelles » au niveau du code « C » généré, il n'en est pas de même pour les objets. Le langage « oS » comporte deux types d'objets :

— Les objets standards, d'un format très similaire aux classes, qui les composent ;
— Le « main », ou corps de programme, point d'entrée unique de l'application binaire.

Le compilateur « os2c » s'appuie sur la déclaration d'un objet pour déclencher la compilation de ce dernier. C'est à ce niveau que les classes se matérialisent en code « C ».

5.12.1 Objets standards

différences entre classes et objets standards

Un objet standard est grammaticalement et lexicalement très similaire à une classe. La notion d'héritage y est intégralement prise en compte, et fonctionne de la même façon.
Philosophiquement, on trouve cependant une différence de taille : les éléments de traitement ne peuvent pas être déclarés au sein d'un objet. Ceci est l'apanage des classes, exclusivement. Un objet est donc surtout un assemblage de classes, et son rôle est principalement :

— de lier les classes pour assurer le flux de données entre « morceaux » (classes) de l'objet ;
— de lier deux objets standards entre eux pour assurer le flux de données entre entités fonctionnelles indépendantes ;
— de surcharger les éléments de calculs de l'objet afin de conditionner les valeurs initiales d'une simulation.

On peut noter que le nom de l'objet standard sera celui utilisé par « os2c » pour construire la fonction « C » contenant la liste des appels aux fonctions « libsim ». Cette suite d'appels traduit en langage « C » la liste des instructions « oS ». La liste traduite est constituée dans l'ordre d'apparition des éléments et des liens déclarés dans les classes. La matérialisation des classes est effectuée dans l'ordre de leur apparition dans la liste correspondante. La liste des instructions entre accolades suivant celle des classes est traitée en dernier. Le canevas grammatical et syntaxique d'un objet standard est donné par la FIGURE 5.14 page 68 :

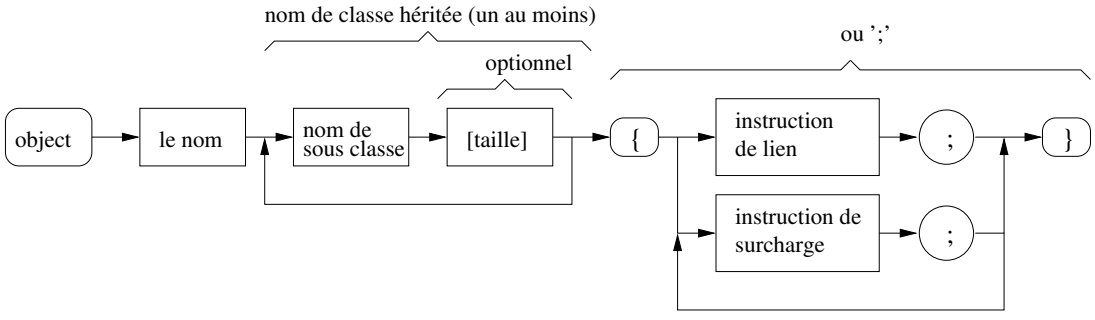


FIGURE 5.14: Grammaire et syntaxe d'un objet avec toutes ses variantes

Les liens entre éléments d'une classe sont de nature grammaticale et syntaxique similaire à ceux existant à l'intérieur d'une classe (§ 5.8.3 page 60). Il en est de même des surcharges paramétriques (§ 5.8.4 page 61).

Les liens entre objets

Il s'agit de relier le champ résultat d'un élément dans un objet source à un élément de type LINK_IN d'un objet destination. Exemple d'une déclaration d'instruction de lien inter objets, avec un élément source à champ de sortie standard :

```
link objet_source      element_source,      // element de type quelconque
     objet_destination element_destination; // element destination de type LINK_IN
```

Si l'élément de l'objet source est de type « FILE_IN » ou « CUSTOM », le lien s'écrit :

```
link objet_source      element_source:0,    // element_source, champ 0
     objet_destination element_destination; // element destination de type LINK_IN
```

Le canevas grammatical et syntaxique d'un lien inter objets est le suivant :

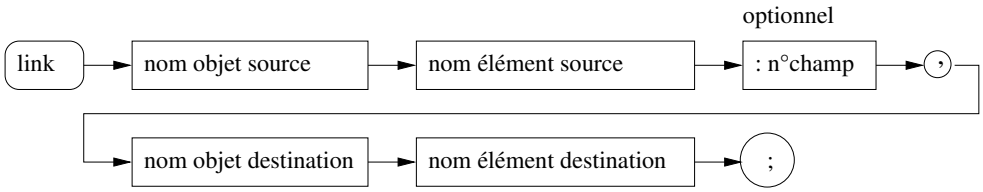


FIGURE 5.15: Grammaire et syntaxe d'une instruction de lien inter objets

5.12.2 L'objet corps de programme

L'objet corps de programme est particulier en ce sens qu'il est unique. C'est le point d'entrée du programme « oS ». Il répertorie les objets standards qui caractérisent l'application, et contrôle le déroulement des itérations de calcul.
Exemple de déclaration la plus simple possible d'un objet corps de programme :

```
main 1.0, // 1.0 = intervalle en secondes entre deux échantillons
     1000 // nombre d'itérations de calculs
{         // la liste des instructions du "main" s'écrit entre les accolades
}
```

Après le mot clef « main », suivent deux paramètres (qui peuvent faire l'objet d'une redéfinition symbolique) :
- l'intervalle d'échantillonnage séparant deux itérations de calculs. Ce paramètre réel est utilisé ensuite par le noyau « libsim » pour le calcul des dérivées et intégrales numériques.
- Le paramètre de contrôle des modalités de fin d'exécution d'une session de calcul. Il y a trois façons différentes de procéder :
  - indiquer un paramètre entier long > à 0, indiquant le nombre d'itérations à effectuer ;
  - indiquer un paramètre entier = 0. Ceci signifie au noyau de calcul que la fin de calcul produira à la première fin de fichier rencontrée. Si aucun élément de type fichier d'entrée n'a été déclarer, la boucle de calcul est sans fin.
  - indiquer un paramètre entier = -1. La fin de calcul survient par évènement externe asynchrone. Ce type d'évènement est signalé par un ou plusieurs éléments de type « STOP ».

Après le paramétrage de contrôle de la simulation, suivent une accolade ouvrante et fermante obligatoires, entre lesquelles on trouvera une liste d'instructions spécifiques au corps de programme. La grammaire et la syntaxe standard de ce dernier sont illustrés par la FIGURE 5.16 page 70 :
Dans cette figure, il est à noter le caractère particulier de l'inclusion « C » multi-ligne, qui couplé à l'instruction de calcul prend un sens particulier qui va être détaillé dans les points suivants.

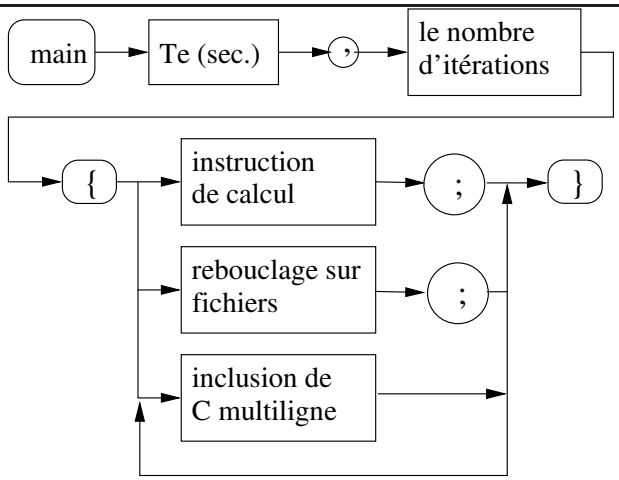


FIGURE 5.16: Grammaire et syntaxe du corps de programme

Instruction de lancement des calculs

L'instruction « compute » a un double rôle dans un corps de programme : définir une liste de calcul, et faire basculer l'application du mode déclaratif au mode calculatoire.
On sous entend par liste de calcul l'ensemble des objets standards que le noyau de simulation doit prendre en compte. L'ordre de déclaration des objets détermine l'ordre de passage dans la liste de calcul au cours du traitement d'une itération : les premiers déclarés sont les premiers calculés.
Il y a deux moyens de gérer les objets de calculs standard :

— **le mode mono module implicite** ; on sous entend qu'il n'y a qu'un fichier source « oS » dans lequel sont déclarés les objets standards ainsi que le corps de programme. C'est la façon la plus simple de procéder. L'ordre de déclaration des objets devient implicitement l'ordre de calcul. Ceci n'exclut pas la possibilité d'utiliser des fichiers de définitions symboliques de niveau « oS » et des fichiers de définitions de classes hors de ce fichier principal « oS » de déclaration général. La constitution de la liste de calcul, ainsi que le basculement en mode calculatoire se fait ensuite ainsi :

```
compute ;
```

— **Le mode multi module explicite** ; on déclare explicitement l'ensemble des objets standards qui doivent être référencés dans la liste de calcul, et qui ne se trouvent pas forcément dans le fichier où se trouve l'objet corps de programme. Associé avec les fichiers de définitions symboliques de niveau « C » et l'éditeur de lien du compilateur « C », ce mode offre la possibilité de gérer de grosses applications. La constitution de la liste de calcul, ainsi que le basculement en mode calculatoire se fait ensuite ainsi :

```
compute objet_1 objet_2 objet_3 ;
```

Dans cet exemple, on référence de façon explicite « objet_1 », « objet_2 », « objet_3 » dans la liste de calcul.
Ce mode de fonctionnement n'exclut pas la possibilité de déclarer un objet standard dans le même fichier « oS » principal que le corps de programme. Il est nécessaire dans ce cas de déclarer aussi son appartenance à la liste de calcul entre le mot clef « compute » et le point virgule.
Pour des usages plus pointus comme le temps réel, il peut être utile d'exclure un objet standard de la liste de calcul principal. Dans ce cas, on déclare l'objet standard dans le fichier « oS » contenant le corps de programme, mais on ne le fait pas émarger dans la liste de calcul.

L'instruction est définie grammaticalement et syntaxiquement par la FIGURE 5.17 page 70 :

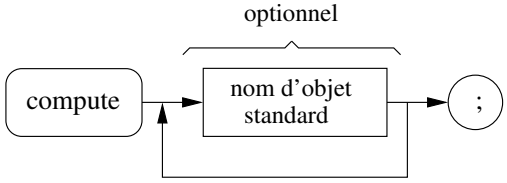


FIGURE 5.17: Grammaire et syntaxe de l'instruction de calcul

Inclusion de code « C » dans une itération de calcul

Il est possible d'adjoindre du code « C » en ligne exécuté après chaque itération de calcul. Ceci permet ainsi de constituer une boucle de calcul spécifique, écrite en code « C » sur plusieurs lignes :

```
compute application  %%
  printf("boucle fichier %d iteration %ld\n", i, _cptboucle);
 %%
```

En lieu et place du point virgule on trouve donc les caractères clefs %% pour instaurer un basculement en mode « C » multi-ligne. Suit ensuite le code « C » spécifique, l'ensemble se termine à nouveau par %%. On termine ainsi l'ajout de code spécifique dans une boucle de calcul.
Il est bon de préciser que l'exécution du code « C » spécifique vient après l'exécution de l'itération de calcul par le noyau « libsim ». Ce rajout optionnel est défini grammaticalement et syntaxiquement par la FIGURE 5.18 page 71 :

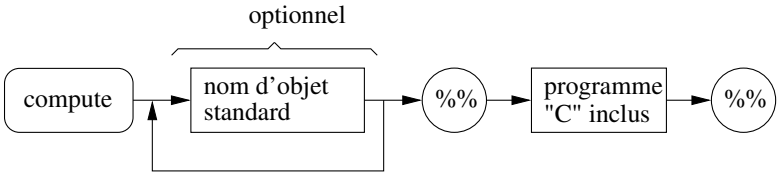


FIGURE 5.18: Grammaire et syntaxe du rajout « C » spécifique

Instruction de rebouclage des fichiers d'entrées

Souvent, il est intéressant de définir un stimulus d'entrée en terme de signal périodique, dont on fixe une période pour un ou plusieurs fichier(s) d'entrée(s). Ensuite, il est possible de préciser au noyau de calcul de « libsim » à chaque fin de fichier(s) d'entrée(s) rencontrée, d'avoir à « reboucler » sur ces mêmes fichiers un nombre paramétrable de fois.
Le noyau de simulation gère automatiquement la fermeture et la réouverture des fichier de type FILE_IN, et garde ouvert les fichiers de type FILE_OUT pendant toute la durée de la simulation. Cette instruction est définie syntaxiquement par la FIGURE 5.19 page 71. Elle est agrégée syntaxiquement à l'instruction « compute » vue plus haut.

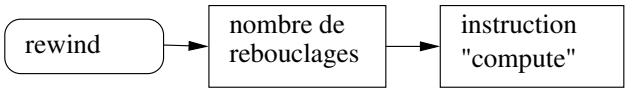


FIGURE 5.19: Définition syntaxique de l'instruction de rebouclage des FILE_IN

Exemple type de déclaration de l'instruction de rebouclage :

```
rewind 3 compute; // reboucler 3 fois sur le(s) fichiers de type FILE_IN
                  // en enchainant les calculs sans interruption.
```

5.13 Bilan des développements « os2c »

L'outil de programmation qui vient d'être exposé a servi au test et à la simulation des travaux évoqués au chapitre 4 (Schwaller et al. 2008, 2009, Schwaller 2011, Schwaller et al. 2013, 2016, 2017). Il se démarque de « matlab, simulink » ou « octave » par le fait qu'il est nativement de nature compilée, et non interprété. Une fois compilé, les ressources systèmes mises en œuvre sont de nature très différentes. « Matlab et Octave » sont à la base spécialisés dans les traitements matriciels, alors qu'« os2c » est plus adapté à la définition de modèles orientés objets et au traitement de flux continus de données.
En ce sens, il se rapproche plus de « Modélica », qui est un outil structurellement conçu pour décrire des modèles mathématiques sous forme d'objets. « Modélica » possède une syntaxe proche de celle de « Java & Matlab ». Toutefois, son utilisation sur des environnement embarquables est difficilement envisageable du fait

des ressources système qu'il nécessite. On le retrouve dans des environnements de CAO comme ceux de « Dymola par Dassault systèmes / Dynasim » où « MapleSim par MapleSoft ».

Le compilateur « os2c » analyse le langage « oS » en une seule passe de compilation et produit en sortie un code source « C » ANSI compilable sous forme d'exécutable sur tout type de machine.
La partie calculatoire, gérée par un noyau de calcul compris dans la bibliothèque de traitement « libsim » est clairement séparée de la partie algorithmique. L'utilisateur n'a pas à se soucier des calculs, qui peuvent être gérés sous forme de réels à virgule fixe ou flottante. Une simple édition de lien permet de choisir la bibliothèque adaptée au type de processeur voulu.
Cette caractéristique influe favorablement sur la portabilité et surtout la rapidité d'exécution.

Cet outil a été largement éprouvé, sur des systèmes d'exploitations et des processeurs divers (80C51, processeurs Intel divers, processeurs ARM, processeurs alpha). Son efficacité, est spectaculaire. A titre de comparaison, le modèle ASM1 (4.81)-(4.84) met 30 minutes à être simulé sur un pc icore3 avec « octave » et la bibliothèque « libode54 »(version en logiciel libre de Matlab). Le même modèle ainsi que l'observateur complet (4.81)-(4.99) met 1/2 secondes sur le même processeur ... Le descriptif du code source « oS » est consultable au chapitre A.

La chaîne de compilation « os2c » peut être utilisée en natif sur toutes les plate-forme où une bibliothèque « libsim » est implantable, et prend très peu de ressources système. On évite alors les « cross-compilations », et celà permet alors d'envisager des générateurs de codes paramétrables, et réagissant à l'exécution d'autres systèmes.
À titre d'exemple, le préprocesseur et le « parser » d'« os2c » fait 140 Ko sur un environnement Linux.
La « cross-compilations »est également envisageable avec « os2c », même si elle n'est pas souhaitable.

Le langage « oS » est conçu pour le traitement par flux de systèmes numériques ou d'équations d'états. En ce sens, on peut dire qu'il est de la même famille que Matlab-Simulink ou open Modélica.
Toutefois, la conception orientée objet est poussée plus loin, du fait de la notion d'héritage multiple vectorisé. La manipulation de matrices multidimensionnelles de degré n d'objets est alors utilisable. Associée aux liens indicés dans des boucles et au paramétrage dynamique, on dispose alors d'un outil puissant pour adapter dynamiquement le code et le fonctionnement des applications compilées. Il n'est alors pas nécessaire de recompiler une simulation avant chaque test.

Le gestion de processus multitâche permettant l'entrée et la sortie de flux synchronisés est également un point fort du langage « oS » car il permet d'interfacer simplement des applications externes indépendantes (sockets réseau par ex.), et de profiter des développements d'autres projets indépendants d'« os2c ».

Avec le langage « oS » on maîtrise parfaitement l'ordre de calcul des composants, ce qui peut être déterminant quand on travaille avec des modèles de systèmes chaotiques ou fortement non linéaires.

Avec « os2c », on dispose de la possibilité d'enfouir les algorithmes compilés en exécutables paramétrables dans une chaîne d'analyse indépendante de la plate forme matérielle. Le logiciel autorise des traitements multitâche matériellement réparti en réseau.

Chapitre 6

Conclusion, perspectives, bilan synthétique global

6.1 Conclusion

Mes thématiques sont ciblées sur la modélisation, l'observation de systèmes dynamiques et l'implantation des algorithmes développés dans des chaînes automatisées de traitements. Ce type de thématique de recherche n'a pas vocation à rester cantonné au domaine de l'automatique et des mathématiques appliquées. En effet, le but poursuivi est de développer des outils de modélisation, observation et identification paramétrique à mêmes d'être mis à profits dans le domaine des sciences pour l'Ingénieur, dans le domaine du génie biologique et médical, où même dans le domaine des sciences économiques ...

6.2 Perspectives ouvertes par les recherches

Les perspectives ouvertes par les travaux menés à l'INSA sont à vocation fondamentale et transverse. Du point de vue fondamental, on peut évoquer plusieurs pistes de réflexion.

- Exploration des systèmes multivariables à sorties et entrées multiples (MIMO).
- Identification en continu et en temps réel des paramètres déterminant les lois de comportement des modèles non linéaires en plus des états. La structure proposée par le nouvel observateur se prête à ce type de démarche. Une telle approche peut être une réponse au problème des incertitudes paramétriques.
- Couplage de l'observateur à des systèmes d'équations aux dérivées partielles, pour caractériser dans le temps et dans l'espace la dynamique des systèmes.
- Adaptation de l'observateur à la contrainte des prises de mesures asynchrones.
- Couplage de l'observateur aux algorithmes de commande non linéaire.
- Couplage de l'observateur aux outils de diagnostic de pannes et défauts de capteurs.

Les problématiques fondamentales qui viennent d'être évoquées, ont pour vocation à s'appliquer aux systèmes dynamiques que l'on trouve dans les sciences de l'ingénieur. De façon non exhaustive on peut évoquer quelques exemples :

- la modélisation de systèmes biomécaniques et de systèmes cognitifs ;
- la caractérisation de fluides complexes ;
- la modélisation de procédés thermiques ;
- la modélisation dans le domaine de l'eau et l'environnement ;
- la modélisation de réacteurs chimiques ou biologiques ;
- la modélisation de systèmes en robotique ;
- la modélisation de véhicules ou d'aéronefs ;
- la cryptographie et le diagnostic de procédés ;
- la commande adaptative ...

Ces recherches qui visent à développer des outils d'estimations d'états couplés aux chaînes de mesures peuvent donc trouver un point d'ancrage dans de nombreux projets de nature disciplinaires différentes.

La base logicielle étudiée au chapitre 5 doit encore être valorisée sous forme de publication, et il est prévu de la diffuser sous une licence GNU, de façon à rester de domaine public, et à fédérer ensuite une communauté de développeurs, car le développement d'outils de calculs ne se termine jamais ...

6.3 Bilan synthétique

6.3.1 Résumé des responsabilités scientifiques

Entre 1992 et 1993 :	chargé de la mise en place de l'atelier d'exploration fonctionnel de l'IMFS dans le service ORL du CHU Hautepierre de Strasbourg.
Entre 1995 et 1993 :	chargé à 100 % de l'encadrement de thèse de Mr.*F. Drouhin.*
Entre 1999 et 2003 :	titulaire d'une prime d'encadrement doctoral et de recherche, et codirecteur de thèse de *S. Faucheux*
En 2000 :	codirection de l'équipe « Systèmes Bio Mécaniques et Cognitifs » de l'IMFS. Élu comme membre du conseil d'UMR.
Printemps 2002, mai 2004 :	Élu membre suppléant de la commission de spécialistes 61/63 de l'ULP.
En juin 2002 :	codirection de l'équipe « Écoulement de Fluides Chargés et Assainissement » (EFCA) de l'IMFS. Membre du conseil scientifique de l'IMFS.
En août 2002 :	responsable de l'action spécifique conjointe CNRS-Ministère de la Recherche intitulée « Phénomènes de transport solide et de mélange : liaisons entre échelles microscopiques et macroscopiques » (Bulletin du SPI, n° 34, août 2002). La notification du projet a été effectuée le 17/11/2003, l'arrivée du courrier au bureau de la Recherche de l'ULP a lieu le 7/1/2004.
En janvier 2003 :	codirection du projet RITEAU (Réseau de recherche et d'Innovation Technologique EAU et technologies de l'environnement) « MES-Flux »La notification du projet a eu lieu le 1/10/2003, l'arrivée du courrier au bureau de la Recherche de l'ULP a eu lieu le 10/10/2003.
En juillet 2004 :	gestion de la relocalisation d'une partie de l'UTR EFCA à l'IUT Louis Pasteur, comme antenne de l'IMFS, et codirection de l'équipe pour le quadriennal 2004/2008.
Septembre 2006, mars 2007 :	Obtention d'une délégation CNRS.
Depuis juin 2015 :	reviewer dans *Int. J. of Applied Mathematics and Computer Science.*

6.3.2 Liste des travaux et publications

Actuellement, ma liste de publication comporte :

- 12 publications internationales ; les revues ciblées étaient :
  - — Journal of Mathematics Research (2017)
  - — International Journal of Applied Mathematics and Computer Science (2013 et 2016)
  - — Biomedical Signal Processing and Control (2007)
  - — Office européen des brevets (2004)
  - — IEEE Transaction on Nuclear Science (2000, 1998, 1997)
  - — Nuclear Instruments And Methods In Physics Research (1999 (2 fois), 2000)
  - — Technical report 029 du Cern, 1998
- 6 publications nationales ; les revues ciblées étaient :
  - — Instrumentation - aspects fondamentaux (2004)
  - — Thèse de doctorat de *S. Faucheux* ULP (2003) co-encadrant
  - — Institut National de la Propriété Industrielle (2001)
  - — Thèse de doctorat de *F. Drouhin* UHA (1998) co-encadrant
  - — Thèse de doctorat de *B. Schwaller* ULP (1994)
  - — La revue d'ONO (1993)
- 8 conférences internationales ; les conférences ciblées (toutes avec actes) étaient :
  - — IEEE Mediterranean Conference on Control and Automation (2008)
  - — Bàràny Society Meeting (2004)
  - — International Symposium on Ultrasonic Doppler Method for Fluid Mechanics and Fluid Engineering (2004)
  - — Conférence Internationale sur l'Automatisation Industrielle (2003)
  - — IRCOBI conference (2002)
  - — Japan-France seminar on Intelligent Materials and Structures (1998)
  - — IEEE Real Time conference (1997)
  - — IEEE Nuclear Science Symposium and Medical Imaging conference (1997)
- 4 conférences nationales ; les conférences ciblées étaient :

  - Colloque National de la Recherche en IUT (2009) avec actes
  - Journées Doctorales d'Automatisme (2003) avec actes
  - Congrès général de la Société Française de Physique (2001) conférence 1 heure
  - Colloque de la section 22 du comité national du CNRS (1993) dont j'ai participé à l'organisation.
- 1 conférence invitée le 30 juin 2011 au Laboratoire de Mécanique et Génie Civil, Université Montpellier 2, UMR 5508 CNRS.

Liste des publications internationales

B. Schwaller, D. Ensminger, B. Dresp & J. Ragot (2017), 'Resistance to noise of non-linear observers in canonical form. Application to a Sludge Activation Model'. *Journal of Mathematics Research*, **9**(2), 38–55.DOI : 10.5539/jmr.v9n2p38.

B. Schwaller, D. Ensminger, B. Dresp, & J. Ragot (2016). 'State estimation for miso non-linear systems in generalized canonical form of regulation'. *International Journal of Applied Mathematics and Computer Science*, **26**(3), 569–583. DOI : 10.1515/amcs-2016-0040.

B. Schwaller, D. Ensminger, B. Dresp, & J. Ragot (2013). 'State estimation for a class of non-linear systems'. *International Journal of Applied Mathematics and Computer Science*, **23**(2), 383–394. DOI : 10.2478/amcs-2013-0029.

S. Faucheux, B. Schwaller, & A. Buizza (April 2007). 'Automatic detection and removal of fast phases from nystagmographic recordings by optimal thresholding'. *Biomedical Signal Processing and Control*, **2**(2), 144–150. ISSN 1746-8094, DOI 10.1016/j.bspc.2007.05.002.

S. Fischer, D. Hurter, P. Schmitt, B. Schwaller, & O. Scrivener (4 février 2004). 'Procédé et dispositif de mesure des vitesses de liquides dans des conduites et des canaux'. Technical Report 02764928.4-2209-FR0202187, *Office Européen des Brevets*, extension du principe aux mesures à grande profondeur ; date de dépôt : 24 juin 2002.

F. Drouhin, B. Schwaller, A. Pallarès, J.C. Fontaine, F. Jeanneau, D. Huss, & P.G. Verdini (2000). 'The data acquisition system for the cms tracker beam tests'. *IEEE Transaction on Nuclear Science*, **47**(6), 2773–2780.

Y. Benhammou, R. Blaes, F. Drouhin, J.C. Fontaine, D. Huss, F. Jeanneau, V. Mack, B. Schwaller, J.M. Brom, I. Ripp-Baudot, & A Zghiche (2000). 'Beam test results of a wedge-shaped msgc+gem detector at cern'. *Nuclear Instruments And Methods In Physics Research*, **A**(441), 452–458.

M. Ackermann, S. Atz, V. Aulchenko, S. Bachmann, B. Baiboussinov, S. Barthe, W. Beaumont, T. Beckers, F. Beissel, Y. Benhammou, A.M. Bergdolt, K. Bernier, P. Blukm, A. Bondar, O. Bouhali, I. Boulogne, M. Bozzo, J.M. Brom, C. Camps, V. Chorowicz, J. Coffin, V. Commichau, D. Contardo, J. Croix, J. De Troy, F. Drouhin, H. Eberlé, G. Flukgge, J.C. Fontaine, W. Geist, U. Goerlach, K. Gundlfinger, K. Hangarter, R. Haroutunian, J.M. Helleboid, T. Henkes, M. Hoffer, C. Hoffman, D. Huss, R. Ischebeck, F. Jeanneau, P. Juillot, S. Junghans, M.R. Kapp, K. Karcher, D. Knoblauch, M. Kraeber, M. Krauth, J. Kremp, A. Lounis, K. Lubelsmeyer, C. Maazouzi, D. Macke, R. Metri, L. Mirabito, T. Muller, V. Nagaslaev, D. Neuberger, A. Nowack, A. Pallarès, D. Pandoulas, M. Petertill, O. Pooth, C. Racca, I. Ripp-Baudot, E. Ruoff, A. Sauer, P. Schmitz, R. Schulte, A. Schultz von Dratzig, J.P. Schunk, G. Schuster, B. Schwaller, L. Shektman, R. Siedling, M.H. Sigward, H.J. Simonis, G. Smadja, J. Stefanescu, H. Szesny, A. Tatarinov, W.H. Thukmmel, S. Tissot, V. Titov, T. Todorov, M. Tonutti, F. Udo, C. Vander Velde, W. Van Doninck, Ch. Van Dyck, P. Vanlaer, L. Van Lancker, P.G. Verdini, S. Weseler, B. Wittmer, R. Wortmann, A. Zghiche, & V. ZHUKOV (1999). 'Large scale test of wedge shaped micro strip gas counters. *Nuclear Instruments And Methods In Physics Research*, **3, sectionA**(436), 313–325.

P. Fessler, J. Coffin, H. Eberlé, C. de Raad Iseli, B. Hilt, D. Huss, F. Krummenacher, J.R. Lutz, G. Prevost, A. Renouprez, M.H. Sigwart, B. Schwaller, & C. Voltolini (1999). 'An important step forward in continuous spectroscopic imaging of ionising radiations using asics'. *Nuclear Instruments And Methods In Physics Research*, **1-2, Section A**(421), 130–141.

B. Schwaller, F. Drouhin, A. Pallarès, J.C. Fontaine, Y. Benhammou, F. Charles, D. Huss, & C. Hoffmann (1998). *The trigger system of the cms barrel and forward milestones.* Technical Report 029, CERN.

B. Schwaller, C. Hoffmann, F. Drouhin, A. Pallarès, J.C. Fontaine, Y. Benhammou, F. Charles, & D. Huss (october 1998). 'The trigger system of the first cms tracker beam tests'. *IEEE Transaction on Nuclear Science*, **45**(5), 2314–2318.

F. Drouhin, B. Schwaller, J.C. Fontaine, F. Charles, & D. Huss (August 1997). 'A unix svr4-os9 distributed data acquisition for high energy physics'. *IEEE Transaction on Nuclear Science*, **45**(4) 1923–1927. Part. 1 of 3 parts, selective paper for then's conference on real time RT97 Beaune France, 22-26 september 1997.

Liste des publications nationales

S. Fischer, P. Schmitt, & B. Schwaller (2004). 'Suppression du bruit par identification en vélocimétrie ultrasonore pulsée'. *Instrumentation - aspects fondamentaux*, **1** 143–150.

S. Faucheux (2003). *Identification paramétrique des capteurs d'accélération angulaire chez l'Homme*. Thèse de doctorat de l Université Louis Pasteur de Strasbourg, soutenance le 6 février 2003, option EEA. Directeurs de thèse : B. Schwaller et A. Buizza ; cotutelle franco-italienne entre l'Université Louis Pasteur de Strasbourg et l'Università degli Studi di Pavia.

S. Fischer, D. Hurter, P. Schmitt, B. Schwaller, & O.Scrivener (2001). 'Procédé et dispositif de mesure des vitesses de liquides dans des conduites et des canaux'. Rapport technique 0108346, Institut National de la Propriété Industrielle, 25 juin 2001. Brevet français.

F. Drouhin (1998). *Conception d'un système d'acquisition de données dans le cadre des tests Milestone Barrel 1 et Milestone Forward 1*. Thèse de doctorat de l'Université de Haute Alsace (France), soutenance le 4 décembre 1998. Directeur de thèse : D. Huss ; co-encadrant : B. Schwaller.

B. Schwaller (1993). *Plasticité du réflexe vestibulo oculomoteur et commande adaptative, modèles et procédures*. Thèse de doctorat de l'Université Louis Pasteur de Strasbourg, soutenance le 30 novembre 1993. Directeur de thèse : J.L. Eichhorn ; prix de thèse ADRÉRUS 1994 (Association pour le Développement des Relations entre l'Économie et la Recherche auprès des Universités de Strasbourg et de l'Université de Haute Alsace).

C. Conraux, A. Gentine, C. Kopp, & B. Schwaller (1993). 'L'exploration multifréquentielle du système vestibulaire'. *La revue d'ONO*, **19/20**.

Liste des conférences internationales

B. Schwaller, D. Ensminger, J. Ragot, & B. Dresp (2008). State estimates for non-linear siso systems, *in* '16th IEEE Mediterranean Conference on Control and Automation, Ajaccio', (pp 1464–1471), 25-27 June 2008. DOI 10.1109/MED.2008.4602117, ISBN 978-1-4244-2504-4.

S. Faucheux, B. Schwaller, A. Gentine, & A. Buizza (2006). Experimental identification of vertical vestibulo-ocular reflex, *in* '24th Bàràny Society Meeting, University of Uppsala (S)', page 14, 11th-14th june 2006.

S. Fischer, P. Schmitt, & B. Schwaller (2004). Spectral reconstruction method for liquid velocity measurement beyond the nyquist limit, *in* '4. International Symposium on Ultrasonic Doppler Method for Fluid Mechanics and Fluid Engineering, University of Sapporo (Japan)', september 2004.

S. Faucheux, B. Schwaller, & D. Ensminger (2003). Identification par une méthode basée sur la stabilité selon liapounoff, *dans la* 'Quatrième Conférence Internationale sur l'Automatisation Industrielle, Université de Montréal (CND)', n° MS-06, juin 2003, Session 6.

R. Willinger, N. Bourdet, B. Schwaller, & F. Le Gall (2002). Modal characteristics of the head-neck-system in vivo, *in* 'IRCOBI conference, Munich (Germany)', september 2002.

C. Kopp, A. Bouveresse, J.L. Eichhorn, A. Gentine, B. Hoess, E. Njeugna, & B. Schwaller (1998). What is smart in the inner ear pressure sensors, *in* 'Second Japan-France seminar on Intelligent Materials and Structures, Université Louis Pasteur de Strasbourg (France)', 1-3 july 1998.

B. Schwaller, F. Drouhin, J.C. Fontaine, F. Charles, & D. Huss (1997). A unix svr4-os9 distributed data acquisition for high energy physics, *in* 'X$^{\text{th}}$ IEEE Real time conference, Beaune', pages 379–382, 22-26 september 1997. Publié dans les actes de la conférence.

B. Schwaller, F. Drouhin, J.C. Fontaine, F. Charles, & D. Huss (1997). A unix svr4-os9 distributed data acquisition for high energy physics, *in* 'IEEE Nuclear Science Symposium and Medical Imaging conference, Beaune', september 1997. Affichage par poster.

Liste des conférences nationales

B. Schwaller (2011). Estimation d'état pour systèmes non linéaires à une entrée une sortie, *conférence invitée, séminaire interne, Université de Montpellier 2*, 30 juin 2011, Laboratoire de Mécanique et Génie Civil de Montpellier, UMR CNRS 5508.

B. Schwaller, D. Ensminger, J. Ragot, & B. Dresp (2009). Estimation d'état pour systèmes non linéaires à une entrée une sortie, *dans le* '15ème Colloque National de la Recherche en IUT, Université de Lille 1', volume hal-00443039, 8-10 juin 2009.

S. Faucheux, B. Schwaller, & D. Ensminger (2003). Identification paramétrique des canaux semi-circulaires humains à partir des fonctions de transfert et par une méthode basée sur la stabilité selon Liapounoff, *dans* 'JDA03 Journées Doctorales d'Automatisme', ValenSciences n^{o} 3, actes des JDA 2003, pages 167–172, 25-27 juin 2003. Presses Universitaires de Valencienne.

B. Schwaller (2001). L'homme dans le complexe espace temps en situation normale, extrême et pathologique, *dans le* 'Congrès général de la Société Française de Physique, Strasbourg', 9-13 juillet 2001.

B. Schwaller, B. Hoess, G. GRAIGNIC, F. Moeller, C. Speeg, & J.M. Krempff (1993). Plasticité du réflexe vestibulo oculomoteur et contrôle adaptatif, *dans le* 'IV $^{\text{ième}}$ colloque « De la recherche à la découverte », section 22 du comité national du CNRS', 1-2 décembre 1993, Strasbourg.

Références bibliographiques générales

Ackermann, M., Atz, S., Aulchenko, V., Bachmann, S., Baiboussinov, B., Barthe, S., Beaumont, W., Beckers, T., Beissel, F., Benhammou, Y., Bergdolt, A., Bernier, K., Blukm, P., Bondar, A., Bouhali, O., Boulogne, I., Bozzo, M., Brom, J., Camps, C., Chorowicz, V., Coffin, J., Commichau, V., Contardo, D., Croix, J., Troy, J. D., Drouhin, F., Eberlé, H., Flukgge, G., Fontaine, J., Geist, W., Goerlach, U., Gundlfinger, K., Hangarter, K., Haroutunian, R., Helleboid, J., Henkes, T., Hoffer, M., Hoffman, C., Huss, D., Ischebeck, R., Jeanneau, F., Juillot, P., Junghans, S., Kapp, M., Karcher, K., Knoblauch, D., Kraeber, M., Krauth, M., Kremp, J., Lounis, A., Lubelsmeyer, K., Maazouzi, C., Macke, D., Metri, R., Mirabito, L., Muller, T., Nagaslaev, V., Neuberger, D., Nowack, A., Pallarès, A., Pandoulas, D., Petertill, M., Pooth, O., Racca, C., Ripp-Baudot, I., Ruoff, E., Sauer, A., Schmitz, P., Schulte, R., von Dratzig, A. S., Schunk, J., Schuster, G., Schwaller, B., Shektman, L., Siedling, R., Sigward, M., Simonis, H., Smadja, G., Stefanescu, J., Szesny, H., Tatarinov, A., Thukmmel, W., Tissot, S., Titov, V., Todorov, T., Tonutti, M., Udo, F., Velde, C. V., Doninck, W. V., Dyck, C. V., Vanlaer, P., Lancker, L. V., Verdini, P., Weseler, S., Wittmer, B., Wortmann, R., Zghiche, A. & ZHUKOV, V. (1999), 'Large scale test of wedge shaped micro strip gas counters', *Nuclear Instruments And Methods In Physics Research* **3, section A**(436), 313–325.

Alma1, M. & Darouach, M. (2014), 'Adaptive observers design for a class of linear descriptor systems', *Automatica* **50**(2), 578–583.

Back, J. & Seo, J. (2008), 'Constructive algorithm for system immersion into non-linear observer form', *International Journal of Control* **81**(2), 317–331.

Bastin, G. & Gevers, M. (1988), 'Stable adaptive observers for nonlinear time-varying systems', *IEEE Transactions on Automatic Control* **33**(7), 650–658.

Benhammou, Y., Blaes, R., Drouhin, F., Fontaine, J., Huss, D., Jeanneau, F., Mack, V., Schwaller, B., Brom, J., Ripp-Baudot, I. & Zghiche, A. (2000), 'Beam test results of a wedge-shaped msgc+gem detector at cern', *Nuclear Instruments And Methods In Physics Research* **A**(441), 452–458.

Besançon, G., Voda, A. & Jouffroy, G. (2010), 'A note on state and parameter estimation in a van der pol oscillator', *Automatica* **46**(10), 1735–1738.

Besançon, G., Zhang, Q. & Hammouri, H. (2004), High gain observer based state and parameter estimation in nonlinear systems, *in* 'Symposium on Nonlinear Control Systems, 6th IFAC Symposium, NOLCOS', pp. 325–332.

Bestle, D. & Zeitz, M. (1983), 'Canonical form observer design for non-linear time-variable systems', *International Journal of Control* **38**(2), 419–431.

Bezzaoucha, S., Marx, B., Maquin, D. & Ragot, J. (2013), State and parameter estimation for time-varying systems : a Takagi-Sugeno approach, *in* 'American Control Conference (ACC), Washington DC', pp. 1050–1055.

Bodizs, L., Srinivasan, B. & Bonvin, D. (2011), 'On the design of integral observers for unbiased output estimation in the presence of uncertainty', *Journal of Process Control* **21**(3), 379–390.

Boker, A. & Khalil, H. (2013), 'Nonlinear observers comprising high-gain observers and extended Kalman filters', *Automatica* **49**(12), 3583–3590.

Bouraoui, I., Farza, M., Ménard, T., Abdennour, R. B., M'Saad, M. & Mosrati, H. (2015), 'Observer design for a class of uncertain nonlinear systems with sampled outputs : application to the estimation of kinetic rates in bioreactors', *Automatica* **55**, 78–87.

Bullinger, E., Ilchmann, A. & Allgoewer, F. (1998), A simple adaptive observer for nonlinear systems, *in* 'Nonlinear control systems design 1998 : a proceedings volume from the 4th IFAC Symposium, NOLCOS', Vol. 2, Ed. by H. J.C. Huijberts, Oxford : Pergamon Press, pp. 781–786.
URL: *http ://www.db-thueringen.de/servlets/DerivateServlet/Derivate-7386/NOLCOS1998_ BIA.pdf*

Chen, W., Khan, A., Abid, M. & Ding, S. (2011), 'Integrated design of observer based fault detection for a class of uncertain nonlinear systems', *International Journal of Applied Mathematics and Computer Science* **21**(3), 423–430.

Ciccarella, G., Mora, M. D. & Germani, A. (1993), 'A *Luenberger*-like observer for nonlinear systems', *International Journal of Control* **57**(3), 537–556.

C.Kopp, Bouveresse, A., Eichhorn, J., Gentine, A., Hoess, B., Njeugna, E. & Schwaller, B. (1998), What is smart in the inner ear pressure sensors, *in* 'Second Japan-France seminar on Intelligent Materials and Structures'.

Conraux, C., Gentine, A., Kopp, C. & Schwaller, B. (1993), 'L'exploration multifréquentielle du système vestibulaire', *La revue d'ONO* (19/20).

Drouhin, F., Schwaller, B., Fontaine, J., Charles, F. & Huss, D. (1997), 'A unix svr4-os9 distributed data acquisition for high energy physics', *IEEE Transaction on Nuclear Science* **45**(4), 1923–1927. Part. 1 of 3 parts, selective paper for then's conference on real time RT97 Beaune France, 22-26 september 97.

Drouhin, F., Schwaller, B., Pallarès, A., Fontaine, J., Jeanneau, F., Huss, D. & Verdini, P. (2000), 'The data acquisition system for the cms tracker beam tests', *IEEE Transaction on Nuclear Science* **47**(6), 2773–2780.

Efimov, D. & Fridman, L. (2011), 'Global sliding-mode observer with adjusted gains for locally Lipschitz systems', *Automatica* **47**(3), 565–570.

Farza, M., Bouraoui, I., Ménard, T., Abdennour, R. B. & M'Saad, M. (2014), 'Adaptive observers for a class of uniformly observable systems with nonlinear parametrization and sampled outputs', *Automatica* **50**(11), 2951–2960.

Farza, M., M'Saad, M., Triki, M. & Maatoug, T. (2011), 'High gain observer for a class of non-triangular systems', *Systems and Control Letters* **60**(1), 27–35.

Faucheux, S. (2003), Identification paramétrique des capteurs d'accélération angulaire chez l'Homme, PhD thesis, Université Louis Pasteur de Strasbourg. option EEA, directeurs de thèse : B. Schwaller et A. Buizza ; cotutelle de thèse franco-italienne entre l'Université Louis Pasteur de Strasbourg et l'Università degli Studi di Pavia.

Faucheux, S., Schwaller, B. & Buizza, A. (2007), 'Automatic detection and removal of fast phases from nystagmographic recordings by optimal thresholding', *Biomedical Signal Processing and Control* **2**(2), 144–150. ISSN 1746-8094, DOI 10.1016/j.bspc.2007.05.002.

Faucheux, S., Schwaller, B. & Ensminger, D. (2003*a*), Identification par une méthode basée sur la stabilité selon liapounoff, *in* 'Quatrième Conférence Internationale sur l'Automatisation Industrielle', number MS-06. Session 6.

Faucheux, S., Schwaller, B. & Ensminger, D. (2003*b*), Identification paramétrique des canaux semi-circulaires humains à partir des fonctions de transfert et par une méthode basée sur la stabilité selon liapounoff, *in* 'JDA03 Journées Doctorales d'Automatisme, ValenSciences n°3, actes des JDA 2003', Presses Universitaires de Valencienne, pp. 167–172.

Faucheux, S., Schwaller, B., Gentine, A. & Buizza, A. (2006), Experimental identification of vertical vestibulo-ocular reflex, *in* '24th Bàràny Society Meeting', p. 14.

Fessler, P., Coffin, J., Eberlé, H., de Raad Iseli, C., Hilt, B., Huss, D., Krummenacher, F., Lutz, J., Prevost, G., Renouprez, A., Sigwart, M., Schwaller, B. & Voltolini, C. (1999), 'An important step forward in continuous spectroscopic imaging of ionising radiations using asics', *Nuclear Instruments And Methods In Physics Research* **1-2, Section A**(421), 130–141.

Fischer, S., D.Hurter, Schmitt, P., Schwaller, B. & O.Scrivener (2001), Procédé et dispositif de mesure des vitesses de liquides dans des conduites et des canaux, Technical Report 0108346, Institut National de la Propriété Industrielle. brevet français.

Fischer, S., Hurter, D., Schmitt, P., Schwaller, B. & Scrivener, O. (2004), Procédé et dispositif de mesure des vitesses de liquides dans des conduites et des canaux, Technical Report 02764928.4-2209-FR0202187, Office Européen des Brevets. extension du principe aux mesures à grande profondeur ; date de dépôt : 24 juin 2002.

Fischer, S., P.Schmitt & Schwaller, B. (2004), 'Suppression du bruit par identification en vélocimétrie ultrasonore pulsée', *Instrumentation - aspects fondamentaux* **1**, 143–150.

Fischer, S., Schmitt, P. & Schwaller, B. (2004), Spectral reconstruction method for liquid velocity measurement beyond the nyquist limit, *in* '4. International Symposium on Ultrasonic Doppler Method for Fluid Mechanics and Fluid Engineering'.

Fliess, M. (1990), 'Generalized controller canonical forms for linear and nonlinear dynamics', *IEEE Transactions on Automatic Control* **35**(9), 994–1001.

Gauthier, J. & Bornard, G. (1981), 'Observability for any $u(t)$ of a class of nonlinear systems', *IEEE Transactions on Automatic Control* **AC-26**(4), 922–926.

Gauthier, J., Hammouri, H. & Othman, S. (1992), 'A simple observer for nonlinear systems. applications to bioreactors', *IEEE Transactions on Automatic Control* **37**(6), 875–880.

Ghosh, D., Saha, P. & Chowdhury, A. (2010), 'Linear observer based projective synchronization in delay Roessler system', *Communications in Nonlinear Science and Numerical Simulation* **15**(6), 1640–1647.

Gille, J., Decaulne, P. & Pélegrin, M. (1988), *Systèmes asservis non linéaires*, 5 ième edn, Dunod, Paris. chap. 9, 110-146, isbn 2-04-016955-5.

Guerra, T., Estrada-Manzo, V. & Lendek, Z. (2015), 'Observer design for Takagi–Sugeno descriptor models : an lmi approach', *Automatica* **52**, 154–159.

Hermann, R. & Krener, A. (1977), 'Nonlinear controllability and observability', *IEEE Transactions on Automatic Control* **AC-22**(5), 728–740.

Ibrir, S. (2009), 'Adaptive observers for time-delay nonlinear systems in triangular form', *Automatica* **45**(10), 2392–2399.

J.H. Stoev, J. C. (2002), Towards a spline based adaptive nonlinear observer, *in* 'Proceedings of the 10th Mediterranean Conference on Control and Automation MED 2002 Lisbon'.
URL: *http ://med.ee.nd.edu/MED10-2002/pdf/301.pdf*

Khalifa, T. & Mabrouk, M. (2015), 'On observer for a class of uncertain nonlinear systems', *Nonlinear Dynamics* **79**(1), 359–368.

Krener, A. & Isidori, A. (1983), 'Linearization by output injection and nonlinear observers', *Systems & Control Letters* **3**(1), 47–52.

Lorenz, E. (1963), 'Deterministic nonperiodic flow', *Journal of the Atmospheric Sciences* **20**(2), 130–141.

Luenberger, D. (1966), 'Observers for multivariable systems', *IEEE Transactions on Automatic Control* **AC-11**(2), 190–197.

Martínez-Guerra, R., Mata-Machuca, J., Aguilar-López, R. & Rodríguez-Bollain, A. (2011), *Applications of Chaos and Nonlinear Dynamics in Engineering*, Vol. 1, Springer-Verlag Berlin Heidelberg. ISBN 978-3-642-21921-4.

Mazenc, F. & Dinh, T. (2014), 'Construction of interval observers for continuous-time systems with discrete measurements', *Automatica* **50**(10), 2555–2560.

Menini, L. & Tornambè, A. (2011), 'Design of state detectors for nonlinear systems using symmetries and semi-invariants', *Systems and Control Letters* **60**(2), 128–137.

Mobki, H., Sadeghia, M. & Rezazadehb, G. (2015), 'Design of direct exponential observers for fault detection of nonlinear mems tunable capacitor', *IJE Transactions A : Basics* **28**(4), 634–641.

Morales, A. & Ramirez, J. (2002), 'A pi observer for a class of nonlinear oscillators', *Physics Letters A* **297**(3-4), 205–209.

Nagy-Kiss, A., Marx, B., Mourot, G., Schutz, G. & Ragot, J. (2010), State estimation of two-time scale multiple models with unmeasurable premise variables. application to biological reactors, *in* '49th IEEE Conference on Decision and Control (CDC)', pp. 5689–5694.

Naslin, P. (1960), 'Nouveau critère d'amortissement', *Automatisme* **5**(6), 229–236.

Naslin, P. (1963), 'Polynomes normaux et critère algébrique d'amortissement', *Automatisme* **8**(6), 215–223.

Nikiforov, V., Fradkov, A. & Andrievsky, B. (2005), Adaptive observer-based synchronization of nonlinear nonpassifiable systems, Technical Report math.OC/0509650, Cornell University Library. Optimization and Control, Dynamical Systems, MSC-class : 37D99.
URL: *http ://arxiv.org/abs/math.OC/0509650*

Raghavan, S. & Hedrick, J. (1994), 'Observer design for a class of nonlinear systems', *Int. J. Control* **59**(2), 515–528.

Rauh, A., Butt, S. & Aschemann, H. (2013), 'Nonlinear state observers and extended Kalman filters for battery systems', *International Journal of Applied Mathematics and Computer Science* **23**(3), 539–556.

Röbenack, K. & Lynch, A. (2004), 'An efficient method for observer design with approximately linear error dynamics', *International Journal of Control* **77**(7), 607–612.

Röbenack, K. & Lynch, A. (2006), 'Observer design using a partial nonlinear observer canonical form', *International Journal of Applied Mathematics and Computer Science* **16**(3), 333–343.

Schwaller, B. (1993), Plasticité du réflexe vestibulo oculomoteur et commande adaptative, modèles et procédures, PhD thesis, Université Louis Pasteur de Strasbourg. directeur de thèse : J.L. Eichhorn ; prix de thèse ADRÉRUS 1994 (Association pour le Développement des Relations entre l'Économie et la Recherche auprès des Universités de Strasbourg et de l'Université de Haute Alsace).

Schwaller, B. (2001), L'homme dans le complexe espace temps en situation normale, extrême et pathologique, *in* 'Congrès général de la Société Française de Physique'.

Schwaller, B. (2011), Estimation d'état pour systèmes non linéaires à une entrée une sortie, *in* 'conférence invitée, séminaire interne'.

Schwaller, B., Drouhin, F., Fontaine, J., Charles, F. & Huss, D. (1997*a*), A unix svr4-os9 distributed data acquisition for high energy physics, *in* 'X$^{\text{th}}$ IEEE Real time conference', pp. 379–382. Publié dans les actes de la conférence.

Schwaller, B., Drouhin, F., Fontaine, J., Charles, F. & Huss, D. (1997*b*), A unix svr4-os9 distributed data acquisition for high energy physics, *in* 'IEEE Nuclear Science Symposium and Medical Imaging conference'. Affichage par poster.

Schwaller, B., Drouhin, F., Pallarès, A., Fontaine, J., Benhammou, Y., Charles, F., Huss, D. & Hoffmann, C. (1998), The trigger system of the cms barrel and forward milestones, Technical Report 029, CERN.

Schwaller, B., Ensminger, D., Dresp, B. & Ragot, J. (2016), 'State estimation for miso non-linear systems in generalized canonical form of regulation', *International Journal of Applied Mathematics and Computer Science* **26**(3), 569–583. DOI : 10.1515/amcs-2016-0040.

Schwaller, B., Ensminger, D., Dresp, B. & Ragot, J. (2017), 'Resistance to noise of non-linear observers in canonical form. application to a sludge activation model', *Journal of Mathematics Research* **9**(2). accepté pour publication.

Schwaller, B., Ensminger, D., Dresp-Langley, B. & Ragot, J. (2013), 'State estimation for a class of non-linear systems', *International Journal of Applied Mathematics and Computer Science* **23**(2), 383–394. DOI : 10.2478/amcs-2013-0029.

Schwaller, B., Ensminger, D., Ragot, J. & Dresp, B. (2008), State estimates for non-linear siso systems, *in* '16th IEEE Mediterranean Conference on Control and Automation, Ajaccio', pp. 1464–1471. DOI 10.1109/MED.2008.4602117, ISBN 978-1-4244-2504-4.

Schwaller, B., Ensminger, D., Ragot, J. & Dresp, B. (2009), Estimation d'état pour systèmes non linéaires à une entrée une sortie, *in* '15ème Colloque National de la Recherche en IUT', Vol. hal-00443039.

Schwaller, B., Hoess, B., GRAIGNIC, G., Moeller, F., Speeg, C. & Krempff, J. (1993), Plasticité du réflexe vestibulo oculomoteur et contrôle adaptatif, *in* 'IV ième colloque « De la recherche à la découverte », section 22 du comité national du CNRS'.

Schwaller, B., Hoffmann, C., Drouhin, F., Pallarès, A., Fontaine, J., Benhammou, Y., Charles, F. & Huss, D. (1998), 'The trigger system of the first cms tracker beam tests', *IEEE Transaction on Nuclear Science* **45**(5), 2314–2318.

Söffker, D., Yu, T. & Müller, P. (1995), 'State estimation of dynamical systems with nonlinearities by using proportional-integral observers', *International Journal of Systems Science* **26**(9), 1571–1582.

Thabet, R., Raïssi, T., Combastel, C., Efimov, D. & Zolghadri, A. (2014), 'An effective method to interval observer design for time-varying systems', *Automatica* **50**(10), 2677–2684.

Tornambè, A. (1992), 'High-gain observers for non-linear systems', *International Journal of Systems Science* **23**(9), 1475–1489.

Tyukina, I., Steurb, E., Nijmeijerc, H. & van Leeuwenb, C. (2013), 'Adaptive observers and parameter estimation for a class of systems nonlinear in the parameters', *Automatica* **49**(8), 2409–2423.

Veluvolu, K., Soh, Y. & Cao, W. (2007), 'Robust observer with sliding mode estimation for nonlinear uncertain systems', *IET Control Theory and Applications* **1**(5), 1533–1540. DOI : 10.1515/amcs-2016-0040.

Willinger, R., Bourdet, N., Schwaller, B. & Gall, F. L. (2002), Modal characteristics of the head-neck-system in vivo, *in* 'IRCOBI conference'.

Zeitz, M. (1985), Canonical forms for nonlinear-systems, *in* B. Jakubczyk, W. Respondek & K. Tchoń, eds, 'Proceedings of the Conference on Geometric Theory of Nonlinear Control Systems', Wroclaw Technical University Press, pp. 255–278.

Zeitz, M. (1987), 'The extended Luenberger observer for nonlinear systems', *Systems and Control Letters archive* **9**(2), 149–156.

Zheng, G., Boutat, D. & Barbot, J. (2009), 'Multi-output dependent observability normal form', *Nonlinear Analysis : Theory, Methods and Applications* **70**(1), 404–418.

Annexe A

Code source de l'observateur d'un modèle de boues activées ASM1

A.1 Fichier de définitions « station.h »

```
// paramètres du système physique

#define PARA_SO_SAT     10.0            // g/m^3
#define PARA_KOH        0.2             // g/m^3
#define PARA_KS         20.0            // g/m^3
#define PARA_KS_KOH     4.0             // g/m^3
#define PARA_k1         5e-11
#define PARA_k2         1.08e-5
#define PARA_k3         2.872e-4
#define PARA_k4         3.5e-4
#define PARA_k5         9e-5
#define PARA_k6         1.3161e-12
#define PARA_k7         4.8e-06
#define PARA_k8         7.47e-5
#define PARA_k9         2e-11

#define FC_FILTRE       0.025           // $T_c = 40s$

// conditions initiales du système physique + observateur

#define VI_Z1           4.1             // condition initiale en pollution carbonée
                                        // ou concentration  de substrat biodégradable
#define VI_Z2           3.0             // condition initiale en oxygène dissous
#define VI_Z3           867.0           // condition initiale de concentration de biomasse
#defineVI_XC2          -0.16392         // condition initiale $\widecheck{x}_2(0)$
#defineVI_Z3C           600.0           // condition initiale $\hat{z}_3(0)$

// paramètres de l'observateur avec $T_o = 160 s $

#define OMEGA_o         2.6179938e-2    // $\omega_o=(2 \pi)/T_o \quad T_o=240s$
#define OMEGA_o_2       6.85389e-4      // ${\omega_o}^2$
#define OMEGA_o_M1      38.19718634     // $1/\omega_o$
#define OMEGA_o_M2    1459.025044       // $1/{\tilde{\omega}}^2$
#define OMEGA_o_M3   55730.6515         // $1/{\omega_3}^3$
#define OMEGA_3          5.235987e-3    // $\omega_3=\omega_o/5$
#define OMEGA_3_M1     190.9859317      // $1/\omega_3$
#define TE_OMEGA_o       2.0943951e-1   // $\omega_o \; T_e$, le pas d'intégration renormalisé
#define TE_OMEGA_3       4.1887902e-2   // $\omega_3 \; T_e$, le pas d'intégration renormalisé
```

```
#define PARA_H0          1.0
#define PARA_H1          3.0
#define PARA_H2          3.0
#define PARA_H10         1.0
#define PARA_H11         2.0
#define PARA_H20         1.0
#define PARA_H21         2.0
#define PARA_H30         1.0
#define PARA_H31         2.0

// paramètres de règlage de la simulation

#define G1_BRUIT         0.5           // le gain d'ajustement de la bande de bruit sur $z_1(t)$
#define G2_BRUIT         0.02          // le gain d'ajustement de la bande de bruit sur $z_2(t)$
#define GRAINE_1         1             // la graine du générateur aléatoire sur $z_1(t)$
#define GRAINE_2         0.5           // la graine du générateur aléatoire sur $z_2(t)$

#define TE               8.0           // delta Te en secondes
#define S2H              2.778e-4      // conversion secondes-heures

#define NITERATIONS   4050             // nombre d'itérations pour 9 heures de test
#define PLUS             1             // coef. plus d'un additionneur
#define MOINS           -1             // coef. moins d'un additionneur
```

A.2 Définitions des classes, fichier « station.cls »

```
%% // parties de programme "C" en ligne

REAL z3c_contition_initiale;

int init_z3c( void ) {

 z3c_contition_initiale = VI_Z3C;

 return( 0 );
}

void fct_z3c( double **pe,
      double *ps ) {

double *p_direct = pe[0];       // pointeur sur résultat équation de sortie
double *p_etat = pe[1];         // pointeur sur résultat équation d'état
double *pt = pe[2];             // pointeur sur variable temps en heures

 if ( pt[0] < 0.4 ) {
    if ( z3c_contition_initiale >= 0.0 ) {
      *ps = z3c_contition_initiale ;
      z3c_contition_initiale = -1.0 ;
    } else {
      *ps = p_direct[0] ;
    }
 }
 else {
     *ps += (p_etat[0] * TE) ;
 }
}

%% // partie de progamme "oS"
```

```
class filtre
{
 type     in         LINK_IN;          // in = lien d'entrée standard
 type     add        ADDER 1 -1;       // add = additionneur +-
 type     up         CONST FC_FILTRE;  // $\dot{u}(t)$: fréquence de coupure du filtre Tc=40s, Fc = 1/T
 type     up_tau     CONST OMEGA_o_M1; // $\dot{u}(\tau) = \dot{u}(t) / \omega_o$
 type     u          INTEGRATOR 0 0 0;

 link    in,        add;
 link    add,       up;
 link    up,        u;
 link    up,        up_tau;
 link    u,         add;
}

class fich_entrees {

 // les entrées

 type  u11    FILE_IN ./stimulation/q_in.dat 1;   // l'entrée $u_{11}(t)$
 type  u21    FILE_IN ./stimulation/qa_in.dat 1;  // l'entrée $u_{21}(t)$
 type  u31    FILE_IN ./stimulation/Ss_in.dat 1;  // l'entrée $u_{31}(t)$
 type  u41    FILE_IN ./stimulation/Xbh_in.dat 1; // l'entrée $u_{41}(t)$
}

class c_gene fich_entrees filtre[4] {

 link        c_gene.u11:0,   c_gene.in[0];
 link        c_gene.u21:0,   c_gene.in[1];
 link        c_gene.u31:0,   c_gene.in[2];
 link        c_gene.u41:0,   c_gene.in[3];

 // initialiser les conditions initiales des filtres

 set c_gene.u[0]  INTEGRATOR 0 0  6.54000000e+06; // pour $u_{11}(t)$ filtré
 set c_gene.u[1]  INTEGRATOR 0 0  1.12750000e+00; // pour $u_{21}(t)$ filtré
 set c_gene.u[2]  INTEGRATOR 0 0  2.25500000e+02; // pour $u_{31}(t)$ filtré
 set c_gene.u[3]  INTEGRATOR 0 0  1.06000000e+01; // pour $u_{41}(t)$ filtré
}

class c_mbad {    // classe modèle de boue activée en dynamique

 type  z1          INTEGRATOR 0 0VI_Z1; // z1(0)
 type  z2          INTEGRATOR 0 0VI_Z2; // z2(0)
 type  z3          INTEGRATOR 0 0VI_Z3; // z3(0)

 // les entrées filtrées

 type  u11         LINK_IN;
 type  u21         LINK_IN;
 type  u31         LINK_IN;
 type  u41         LINK_IN;
 type  u12         LINK_IN;
 type  u22         LINK_IN;

 // calcul $\ell_1(t) = u_{11}(t) [ u_{31}(t) - z_1(t) ]$

 type  dSs         ADDER PLUS MOINS; // en dynamique
```

```
type  l1          MULTIPLIER 2;

link              u31,       dSs;
link              z1,        dSs;
link              dSs,       l1;
link              u11,       l1;

// calcul $\ell_2 =
// \frac{z_1(t)}{K_S + z_1(t)} \; \frac{z_2(t)}{K_{OH} + z_2(t)}$

type  nl2         MULTIPLIER 2;

link              z1,        nl2;
link              z2,        nl2;

type  Ks          VALUE PARA_KS;
type  df1         ADDER PLUS PLUS;
type  Koh         VALUE PARA_KOH;
type  df2         ADDER PLUS PLUS;
type  dl2         MULTIPLIER 2;
type  l2          DIVIDER;

link              Ks,        df1;
link              z1,        df1;
link              Koh,       df2;
link              z2,        df2;
link              df1,       dl2;
link              df2,       dl2;
link              nl2,       l2;
link              dl2,       l2;

// calcul $\ell_3(t) = u_{21}(t) [ S_{O,SAT} - z_2(t) ]$

type  So_sat      VALUE PARA_SO_SAT;
type  dSo         ADDER PLUS MOINS;
type  l3          MULTIPLIER 2;

link              So_sat,    dSo;
link              z2,        dSo;
link              dSo,       l3;
link              u21,       l3;

// calcul $\ell_4(t) = u_{11}(t) \; z_2(t)$

type  l4          MULTIPLIER 2;

link              u11,       l4;
link              z2,        l4;

// calcul $\ell_5(t) = u_{11}(t) \; u_{41}(t)$

type  l5          MULTIPLIER 2;

link              u11,       l5;
link              u41,       l5;

// calcul $\ell_6(t) = u_{11}(t) \; z_3(t)$

type  l6          MULTIPLIER 2;
```

```
link              u11,       l6;
link              z3,        l6;

// calcul $\dot{\hat{z}}_1(t) = k_1 \; \ell_1(t) + k_2 \; z_3(t) - k_3 \; \ell_2(t)$

type  l2_z3       MULTIPLIER 2; // calcul $\ell_2(t) \; z_3(t)$
type  k3_l2_z3    CONST PARA_k3;
type  k1_l1       CONST PARA_k1;
type  k2_z3       CONST PARA_k2;
type  z1p         ADDER PLUS PLUS MOINS;
type  s1          CONST OMEGA_o_M1;

link              l2,        l2_z3;
link              z3,        l2_z3;
link              l2_z3,     k3_l2_z3;
link              l1,        k1_l1;
link              z3,        k2_z3;
link              k1_l1,     z1p;
link              k2_z3,     z1p;
link              k3_l2_z3,  z1p;
link              z1p,       z1;
link              z1p,       s1;

// calcul $\dot{{\hat{z}}_2(t) =
// k_4 \; \ell_3(t) - k_1 \; \ell_4(t) - k_4 \; \ell_2(t)$

type  k4_l3       CONST PARA_k4;
type  k1_l4       CONST PARA_k1;
type  k5_l2_z3    CONST PARA_k5;
type  z2p         ADDER PLUS MOINS MOINS;
type  s2          CONST OMEGA_o_M1;

link              l3,        k4_l3;
link              l4,        k1_l4;
link              l2_z3,     k5_l2_z3;
link              k4_l3,     z2p;
link              k1_l4,     z2p;
link              k5_l2_z3,  z2p;
link              z2p,       z2;
link              z2p,       s2;

// calcul $\dot{z}_3(t) =
// k_9 \; \ell_5(t) - k_6 \; \ell_6(t) - k_7 \; z_3(t)
// + k_8 \; \ell_2(t) \; z_3(t) $

type  k9_l5       CONST PARA_k9;
type  k6_l6       CONST PARA_k6;
type  g7_z3       CONST PARA_k7;
type  k8_l2_z3    CONST PARA_k8;
type  z3p         ADDER PLUS MOINS MOINS PLUS;
type  s3          CONST OMEGA_3_M1;

link              l5,        k9_l5;
link              l6,        k6_l6;
link              z3,        g7_z3;
link              l2_z3,     k8_l2_z3;
link              k9_l5,     z3p;
link              k6_l6,     z3p;
```

```
link              g7_z3,     z3p;
link              k8_l2_z3,  z3p;
link              z3p,       z3;
link              z3p,        s3;

// calcul $\dot{n}(t)$

type  z1p_z2      MULTIPLIER 2;
type  z2p_z1      MULTIPLIER 2;
type  np          ADDER PLUS PLUS;

link              z1p,       z1p_z2;
link              z2,        z1p_z2;
link              z2p,       z2p_z1;
link              z1,        z2p_z1;
link              z1p_z2,    np;
link              z2p_z1,    np;

// calcul $\dot{d}(t)$

type  z1p_df2     MULTIPLIER 2;
type  z2p_df1     MULTIPLIER 2;
type  dp          ADDER PLUS PLUS;

link              z1p,       z1p_df2;
link              df2,       z1p_df2;
link              z2p,       z2p_df1;
link              df1,       z2p_df1;
link              z1p_df2,   dp;
link              z2p_df1,   dp;

// calcul $\dot{\ell}_2(t)$

type  np_d        MULTIPLIER 2;
type  n_dp        MULTIPLIER 2;
type  d_d         MULTIPLIER 2;
type  num_l2p     ADDER PLUS MOINS;
type  l2p         DIVIDER;

link              np,        np_d;
link              dl2,       np_d;
link              nl2,       n_dp;
link              dp,        n_dp;
link              dl2,       d_d;
link              dl2,       d_d;
link              np_d,      num_l2p;
link              n_dp,      num_l2p;
link              num_l2p,   l2p;
link              d_d,       l2p;

// calcul $\dot{\ell}_3(t)$

type  u22_dSo     MULTIPLIER 2;
type  u21_z2p     MULTIPLIER 2;
type  l3p         ADDER PLUS MOINS;

link              dSo,       u22_dSo;
link              u22,       u22_dSo;
link              u21,       u21_z2p;
```

```
 link              z2p,       u21_z2p;
 link              u22_dSo,   l3p;
 link              u21_z2p,   l3p;

 // calcul $\dot{\ell}_4(t)$

 type  u12_z2      MULTIPLIER 2;
 type  u11_z2p     MULTIPLIER 2;
 type  l4p         ADDER PLUS PLUS;

 link              z2,        u12_z2;
 link              u12,       u12_z2;
 link              u11,       u11_z2p;
 link              z2p,       u11_z2p;
 link              u12_z2,    l4p;
 link              u11_z2p,   l4p;

 // calcul $\ell_7(t) =
 // z_3(t) \; \dot{\ell}_2(t) + \ell_2(t) \; \dot{z}_3(t)$

 type  l2p_z3      MULTIPLIER 2;
 type  l2_z3p      MULTIPLIER 2;
 type  l7          ADDER PLUS PLUS;

 link              z3,        l2p_z3;
 link              l2p,       l2p_z3;
 link              l2,        l2_z3p;
 link              z3p,       l2_z3p;
 link              l2p_z3,    l7;
 link              l2_z3p,    l7;

 // calcul $\tilde{\psi}(\tau)$ et sa dérivée

 type  k4_l3p      CONST PARA_k4;
 type  k1_l4p      CONST PARA_k1;
 type  g5_l7       CONST PARA_k5;
 type  psit        ADDER PLUS MOINS MOINS;
 type  dpsit       DERIVATIVE;
 type  psitp       CONST OMEGA_o_M3;

 link              l3p,       k4_l3p;
 link              l4p,       k1_l4p;
 link              l7,        g5_l7;
 link              k4_l3p,    psit;
 link              k1_l4p,    psit;
 link              g5_l7,     psit;
 link              psit,      dpsit;
 link              dpsit,     psitp;
}

class mbad c_gene c_mbad { // classe modèle dynamique + géné de signaux

 // liens entrées-filtrées et variables d'états

 link              mbad.c_gene.u[0],mbad.u11;
 link              mbad.c_gene.u[1],mbad.u21;
 link              mbad.c_gene.u[2],mbad.u31;
 link              mbad.c_gene.u[3],mbad.u41;
 link              mbad.c_gene.up[0],mbad.u12;
```

```
link              mbad.c_gene.up[1],mbad.u22;

// liens entre classe c_mref -> c_imba

type  ts          LINK_IN;
type  th          CONST S2H;  // la constante de conversion en heures

link              ts,        th;

// liens d'entrées venant des variables d'états "statiques"

type  z1_0        LINK_IN;  // $z_1(0)$
type  x1_0        LINK_IN;  // $x_1(0)=z_2(0)$
type  x2_0        LINK_IN;  // $x_2(0)=\dot{z}_2(0)$ en normé
type  x3_0        LINK_IN;  // $x_3(0)=\ddot{z}_2(0)=\tilde{\Psi}(0)$ en normé
type  z3_0        LINK_IN;  // $z_3(0)$

// calcul du vecteur $\Delta \underline{y}_a(\tau)$

type  x2          CONST OMEGA_o_M1;
type  x3          CONST OMEGA_o_M2;
type  dy1         ADDER PLUS MOINS;
type  dy2         ADDER PLUS MOINS;
type  dy3         ADDER PLUS MOINS;
type  dy1_dy1     MULTIPLIER 2;
type  dy2_dy2     MULTIPLIER 2;
type  dy3_dy3     MULTIPLIER 2;
type  sdy         ADDER PLUS PLUS PLUS;
type  mdya        SQRT;

link              mbad.z2p,    x2;
link              mbad.psit,   x3;
link              mbad.z2,     dy1;
link              x1_0,        dy1;
link              x2,          dy2;
link              x2_0,        dy2;
link              x3,          dy3;
link              x3_0,        dy3;
link              dy1,         dy1_dy1;
link              dy1,         dy1_dy1;
link              dy2,         dy2_dy2;
link              dy2,         dy2_dy2;
link              dy3,         dy3_dy3;
link              dy3,         dy3_dy3;
link              dy1_dy1,     sdy;
link              dy2_dy2,     sdy;
link              dy3_dy3,     sdy;
link              sdy,         mdya;

// calcul de $\left\| \Delta \dot{\tilde{\Psi}}(\tau) \right\|$ puis $L(\tau)$

type  mdpsitp     ABSOLUTE;
type  L           DIVIDER;

link              mbad.psitp,mdpsitp;
link              mdpsitp,   L;
link              mdya,      L;

// calcul du module du vecteur $\Delta \underline{Z}(\tau)$
```

```
type  dz1          ADDER PLUS MOINS;
type  dz3          ADDER PLUS MOINS;
type  dz1_dz1      MULTIPLIER 2;
type  dz3_dz3      MULTIPLIER 2;
type  ds1_ds1      MULTIPLIER 2;
type  ds2_ds2      MULTIPLIER 2;
type  ds3_ds3      MULTIPLIER 2;
type  sdz          ADDER PLUS PLUS PLUS PLUS PLUS PLUS;
type  mdz          SQRT;
type  sdz1         ADDER PLUS PLUS;
type  mdz1         SQRT;
type  sdz2         ADDER PLUS PLUS;
type  mdz2         SQRT;
type  sdz3         ADDER PLUS PLUS;
type  mdz3         SQRT;

link               mbad.z1,   dz1;
link               z1_0,      dz1;
link               mbad.z3,   dz3;
link               z3_0,      dz3;
link               dz1,       dz1_dz1;
link               dz1,       dz1_dz1;
link               dz3,       dz3_dz3;
link               dz3,       dz3_dz3;
link               mbad.s1,   ds1_ds1;
link               mbad.s1,   ds1_ds1;
link               mbad.s2,   ds2_ds2;
link               mbad.s2,   ds2_ds2;
link               mbad.s3,   ds3_ds3;
link               mbad.s3,   ds3_ds3;
link               dz1_dz1,   sdz;
link               dz1_dz1,   sdz1;
link               dy1_dy1,   sdz;
link               dy1_dy1,   sdz2;
link               dz3_dz3,   sdz;
link               dz3_dz3,   sdz3;
link               ds1_ds1,   sdz;
link               ds1_ds1,   sdz1;
link               ds2_ds2,   sdz;
link               ds2_ds2,   sdz2;
link               ds3_ds3,   sdz;
link               ds3_ds3,   sdz3;
link               sdz,       mdz;
link               sdz1,      mdz1;
link               sdz2,      mdz2;
link               sdz3,      mdz3;

// calcul de $\left\| \Delta \dot{\tilde{\Psi}}_1(\tau) \right\|$

type  dpsit1       DERIVATIVE;
type  psitp1       CONST OMEGA_o_M2;
type  mdpsitp1     ABSOLUTE;
type  L1           DIVIDER;

link               mbad.z1p,  dpsit1;
link               dpsit1,    psitp1;
link               psitp1,    mdpsitp1;
link               mdpsitp1,  L1;
```

```
link              mdz1,      L1;

// calcul de $\left\| \Delta \dot{\tilde{\Psi}}_2(\tau) \right\|$

type  dpsit2      DERIVATIVE;
type  psitp2      CONST OMEGA_o_M2;
type  mdpsitp2    ABSOLUTE;
type  L2          DIVIDER;

link              mbad.z2p,  dpsit2;
link              dpsit2,    psitp2;
link              psitp2,    mdpsitp2;
link              mdpsitp2,  L2;
link              mdz2,      L2;

// calcul de $\left\| \Delta \dot{\tilde{\Psi}}_3(\tau) \right\|$

type  dpsit3      DERIVATIVE;
type  psitp3      CONST OMEGA_o_M2;
type  mdpsitp3    ABSOLUTE;
type  L3          DIVIDER;

link              mbad.z3p,  dpsit3;
link              dpsit3,    psitp3;
link              psitp3,    mdpsitp3;
link              mdpsitp3,  L3;
link              mdz3,      L3;

// préparer les sorties fichiers

type  fich-z1     FILE_OUT z1.dat 2;
type  fich-z2     FILE_OUT z2.dat 2;
type  fich-z3     FILE_OUT z3.dat 2;
type  fich-psit   FILE_OUT psit-d.dat 2;
type  fich-psitp  FILE_OUT psitp-d.dat 2;
type  fich-mdpsitp FILE_OUT mdpsitp.dat 2;
type  fich-mdya   FILE_OUT mdya.dat 2;
type  fich-L      FILE_OUT L.dat 2;
type  fich-L1     FILE_OUT L1.dat 2;
type  fich-L2     FILE_OUT L2.dat 2;
type  fich-L3     FILE_OUT L3.dat 2;
type  fich-z1p    FILE_OUT z1p.dat 2;

link              th,        fich-z1;
link              mbad.z1,   fich-z1;
link              th,        fich-z2;
link              mbad.z2,   fich-z2;
link              th,        fich-z3;
link              mbad.z3,   fich-z3;
link              th,        fich-psit;
link              mbad.psit, fich-psit;
link              th,        fich-psitp;
link              mbad.psitp,fich-psitp;
link              th,        fich-mdpsitp;
link              mdpsitp,   fich-mdpsitp;
link              th,        fich-mdya;
link              mdya,      fich-mdya;
link              th,        fich-L;
link              L,         fich-L;
```

```
 link              th,        fich-L1;
 link              L1,        fich-L1;
 link              th,        fich-L2;
 link              L2,        fich-L2;
 link              th,        fich-L3;
 link              L3,        fich-L3;
 link              th,        fich-z1p;
 link              mbad.z1p,  fich-z1p;

 type  fich-z3p    FILE_OUT z3p.dat 2;
 type  fich-l3p    FILE_OUT l3p.dat 2;
 type  fich-l4p    FILE_OUT l4p.dat 2;
 type  fich-np     FILE_OUT np.dat 2;
 type  fich-dp     FILE_OUT dp.dat 2;
 type  fich-l2p    FILE_OUT l2p.dat 2;

 link              th,        fich-z3p;
 link              mbad.z3p,  fich-z3p;
 link              th,        fich-l3p;
 link              mbad.l3p,  fich-l3p;
 link              th,        fich-l4p;
 link              mbad.l4p,  fich-l4p;
 link              th,        fich-np;
 link              mbad.np,   fich-np;
 link              th,        fich-dp;
 link              mbad.dp,   fich-dp;
 link              th,        fich-l2p;
 link              mbad.l2p,  fich-l2p;

 type  fich-u11    FILE_OUT u11.dat2;
 type  fich-u21    FILE_OUT u21.dat2;
 type  fich-u31    FILE_OUT u31.dat2;
 type  fich-u41    FILE_OUT u41.dat2;
 type  fich-u12    FILE_OUT u12.dat2;
 type  fich-u22    FILE_OUT u22.dat2;

 link              th,         fich-u11;
 link              mbad.c_gene.u[0], fich-u11;
 link              th,         fich-u21;
 link              mbad.c_gene.u[1], fich-u21;
 link              th,         fich-u31;
 link              mbad.c_gene.u[2], fich-u31;
 link              th,         fich-u41;
 link              mbad.c_gene.u[3], fich-u41;
 link              th,         fich-u12;
 link              mbad.c_gene.up[0], fich-u12;
 link              th,         fich-u22;
 link              mbad.c_gene.up[1], fich-u22;
}

class mbadb c_gene c_mbad { // classe modèle dynamique + géné de signaux + bruit

 // liens entrées-filtrées et variables d'états

 link              mbadb.c_gene.u[0],mbadb.  u11;
 link              mbadb.c_gene.u[1],mbadb.  u21;
 link              mbadb.c_gene.u[2],mbadb.  u31;
 link              mbadb.c_gene.u[3],mbadb.  u41;
 link              mbadb.c_gene.up[0],mbadb. u12;
```

```
link              mbadb.c_gene.up[1],mbadb. u22;

// liens entre classe c_mref -> c_imba

type  ts          LINK_IN;
type  th          CONST S2H;  // la constante de conversion en heures

link              ts,        th;

// calculer le bruit de mesure sur $z_1(t)$

type  gene1       RANDOMGRAINE_1;
type  kgene1      CONST G1_BRUIT;
type  ay1         ADDER PLUS PLUS;
type  y1          ABSOLUTE;

link              gene1,     kgene1;
link              kgene1,    ay1;
link              mbadb.z1,  ay1;
link              ay1,       y1;

// calculer le bruit de mesure sur $z_2(t)$

type  gene2       RANDOM GRAINE_2;
type  kgene2      CONST G2_BRUIT;
type  ay2         ADDER PLUS PLUS;
type  y2          ABSOLUTE;

link              gene2,     kgene2;
link              kgene2,    ay2;
link              mbadb.z2,  ay2;
link              ay2,       y2;

// préparer les sorties fichiers

type  fich-z1     FILE_OUT z1.dat 2;
type  fich-z2     FILE_OUT z2.dat 2;
type  fich-z3     FILE_OUT z3.dat 2;
type  fich-y1     FILE_OUT y1.dat 2;
type  fich-y2     FILE_OUT y2.dat 2;

link              th,        fich-z1;
link              mbadb.z1,  fich-z1;
link              th,        fich-z2;
link              mbadb.z2,  fich-z2;
link              th,        fich-z3;
link              mbadb.z3,  fich-z3;
link              th,        fich-y1;
link              y1,        fich-y1;
link              th,        fich-y2;
link              y2,        fich-y2;

type  fich-u11    FILE_OUT u11.dat2;
type  fich-u21    FILE_OUT u21.dat2;
type  fich-u31    FILE_OUT u31.dat2;
type  fich-u41    FILE_OUT u41.dat2;
type  fich-u12    FILE_OUT u12.dat2;
type  fich-u22    FILE_OUT u22.dat2;
```

```
 link              th,        fich-u11;
 link              mbadb.c_gene.u[0], fich-u11;
 link              th,        fich-u21;
 link              mbadb.c_gene.u[1], fich-u21;
 link              th,        fich-u31;
 link              mbadb.c_gene.u[2], fich-u31;
 link              th,        fich-u41;
 link              mbadb.c_gene.u[3], fich-u41;
 link              th,        fich-u12;
 link              mbadb.c_gene.up[0], fich-u12;
 link              th,        fich-u22;
 link              mbadb.c_gene.up[1], fich-u22;
}

class c_obs_mba {  // classe observateur modèle de boue activée

 // les intégrateurs des observateurs

 type  I1          INTEGRATOR 0 0 0 TE_OMEGA_o;
 type  zc1         INTEGRATOR 0 0 VI_Z1TE_OMEGA_o;
 type  I0          INTEGRATOR 0 0 0 TE_OMEGA_o;
 type  xc2         INTEGRATOR 0 0 VI_XC2TE_OMEGA_o;
 type  x1c         INTEGRATOR 0 0 VI_Z2TE_OMEGA_o;
 type  I2          INTEGRATOR 0 0 0 TE_OMEGA_3;
 type  zc2         INTEGRATOR 0 0 VI_Z2TE_OMEGA_3;
 type  I3          INTEGRATOR 0 0 0 TE_OMEGA_3;
 type  zc3         INTEGRATOR 0 0 VI_Z3CTE_OMEGA_3;

 // les entrées filtrées

 type  u11         LINK_IN;
 type  u21         LINK_IN;
 type  u31         LINK_IN;
 type  u41         LINK_IN;
 type  u12         LINK_IN;
 type  u22         LINK_IN;
 type  y1          LINK_IN;  // la variable $z_1(\tau)$
 type  y2          LINK_IN;  // la variable $z_2(\tau)$
 type  heures      LINK_IN;

 // calcul des erreurs d'observation sur $y_1(\tau)$

 type  dz1         ADDER PLUS MOINS;  // en dynamique
 type  h10         CONST PARA_H10;
 type  h11         CONST PARA_H11;

 link              y1,        dz1;
 link              zc1,       dz1;
 link              dz1,       h10;
 link              dz1,       h11;

 // calcul des erreurs d'observation sur $y_2(\tau)$

 type  dy1         ADDER PLUS MOINS;// en dynamique
 type  h0          CONST PARA_H0;
 type  h1          CONST PARA_H1;
 type  h2          CONST PARA_H2;

 link              y2,        dy1;
```

```
link              x1c,       dy1;
link              dy1,       h0;
link              dy1,       h1;
link              dy1,       h2;

// calcul $\check{\ell}_1(t) = u_{11}(t) [ u_{31}(t) - \check{z}_1(t) ]$

type  dSc1        ADDER PLUS MOINS;// en dynamique
type  lc1         MULTIPLIER 2;

link              u31,       dSc1;
link              zc1,       dSc1;
link              dSc1,      lc1;
link              u11,       lc1;

// calcul $\hat{\ell}_2(t) =
// \frac{\check{z}_1(t)}{K_S
// + \check{z}_1(t)} \; \frac{\hat{x}_1(t)}{K_{OH} + \hat{x}_1(t)}$

type  nl2c        MULTIPLIER 2;

link              zc1,       nl2c;
link              x1c,       nl2c;

type  Ks          VALUE PARA_KS;
type  dfc1        ADDER PLUS PLUS;
type  Koh         VALUE PARA_KOH;
type  df2         ADDER PLUS PLUS;
type  dl2c        MULTIPLIER 2;

link              Ks,        dfc1;
link              zc1,       dfc1;
link              Koh,       df2;
link              x1c,       df2;
link              dfc1,      dl2c;
link              df2,       dl2c;

type  l2c         DIVIDER;

link              nl2c,      l2c;
link              dl2c,      l2c;

// calcul $\check{\ell}_2(t) =
// \frac{\check{z}_1(t)}{K_S +
// \check{z}_1(t)} \; \frac{\check{z}_2(t)}{K_{OH} + \check{z}_2(t)}$

type  nlc2        MULTIPLIER 2;

link              zc1,       nlc2;
link              zc2,       nlc2;

type  dfc2        ADDER PLUS PLUS;
type  dlc2        MULTIPLIER 2;

link              Koh,       dfc2;
link              zc2,       dfc2;
link              dfc1,      dlc2;
link              dfc2,      dlc2;
```

```
type  lc2         DIVIDER;

link              nlc2,      lc2;
link              dlc2,      lc2;

// calcul $\hat{\ell}_3(t) = u_{21}(t) [ S_{O,SAT} - \hat{x}_1(t) ]$

type  So_sat      VALUE PARA_SO_SAT;
type  dSo         ADDER PLUS MOINS;
type  l3c         MULTIPLIER 2;

link              So_sat,    dSo;
link              x1c,       dSo;
link              dSo,       l3c;
link              u21,       l3c;

// calcul $\check{\ell}_3(t) = u_{21}(t) [ S_{O,SAT} - \check{z}_2(t) ]$

type  dSco        ADDER PLUS MOINS;
type  lc3         MULTIPLIER 2;

link              So_sat,    dSco;
link              zc2,       dSco;
link              dSco,      lc3;
link              u21,       lc3;

// calcul $\hat{\ell}_4(t) = u_{11}(t) \; \hat{x}_1(t)$

type  l4c         MULTIPLIER 2;

link              u11,       l4c;
link              x1c,       l4c;

// calcul $\check{\ell}_4(t) = u_{11}(t) \; \check{z}_2(t)$

type  lc4         MULTIPLIER 2;

link              u11,       lc4;
link              zc2,       lc4;

// calcul $\hat{\ell}_5(t) = u_{11}(t) \; u_{41}(t)$

type  l5c         MULTIPLIER 2;

link              u11,       l5c;
link              u41,       l5c;

// calcul $\widecheck{g}_3(t)$ équation de sortie

type  k4_l3c      CONST PARA_k4;
type  k1_l4c      CONST PARA_k1;
type  wo_xc2      CONST OMEGA_o;
type  nz3c        ADDER PLUS MOINS MOINS;
type  k5_l2c      CONST PARA_k5;
type  g3c         DIVIDER;

link              l3c,       k4_l3c;
link              l4c,       k1_l4c;
link              xc2,       wo_xc2; // traduire $\widecheck{x}_{22}(\tau)$ en temporel
```

```
link              k4_l3c,    nz3c;
link              k1_l4c,    nz3c;
link              wo_xc2,    nz3c;
link              l2c,       k5_l2c;
link              nz3c,      g3c;
link              k5_l2c,    g3c;

// calcul $\hat{\ell}_6(t) = u_{11}(t) \; \hat{g}_3(t)$

type  l6c         MULTIPLIER 2;

link              u11,       l6c;
link              g3c,       l6c;

// calcul $\check{\ell}_6(t) = u_{11}(t) \; \check{z}_3(t)$

type  lc6         MULTIPLIER 2;

link              u11,       lc6;
link              zc3,       lc6;

// calcul des erreurs d'observation sur $\widehat{g}_3(\tau)$

type  dz3         ADDER PLUS MOINS;  // en dynamique
type  h30         CONST PARA_H30;
type  h31         CONST PARA_H31;

link              g3c,       dz3;
link              zc3,       dz3;
link              dz3,       h30;
link              dz3,       h31;

// calcul $\tilde{Psi}_3(\tau)$

type  k6_lc6      CONST PARA_k6;
type  k7_zc3      CONST PARA_k7;
type  lc2_zc3     MULTIPLIER 2;
type  k8_lc2_zc3  CONST PARA_k8;
type  k9_l5c      CONST PARA_k9;
type  psi3t       ADDER PLUS PLUS MOINS MOINS;
type  psi3t_tau   CONST OMEGA_3_M1;

link              lc6,       k6_lc6;
link              zc3,       k7_zc3;
link              lc2,       lc2_zc3;
link              zc3,       lc2_zc3;
link              lc2_zc3,   k8_lc2_zc3;
link              l5c,       k9_l5c;
link              k8_lc2_zc3,psi3t;
link              k9_l5c,    psi3t;
link              k6_lc6,    psi3t;
link              k7_zc3,    psi3t;
link              psi3t,     psi3t_tau;

// calcul du corps de l'observateur sur $\widehat{g}_3(t)$

type  zc3p        ADDER PLUS PLUS PLUS;
type  wo_zc3p     CONST OMEGA_3;
```

```
link              h30,       I3;
link              h31,       zc3p;
link              I3,        zc3p;
link              psi3t_tau, zc3p;
link              zc3p,      zc3;
link              zc3p,      wo_zc3p;

// calcul $\tilde{\Psi}_1(\tau)$

type  k1_lc1      CONST PARA_k1;
type  k2_zc3      CONST PARA_k2;
type  k3_lc2_zc3  CONST PARA_k3;
type  psi1t       ADDER PLUS PLUS MOINS;
type  psi1t_tau   CONST OMEGA_o_M1;

link              lc1,       k1_lc1;
link              zc3,       k2_zc3;
link              lc2_zc3,   k3_lc2_zc3;
link              k1_lc1,    psi1t;
link              k2_zc3,    psi1t;
link              k3_lc2_zc3,psi1t;
link              psi1t,     psi1t_tau;

// calcul du corps de l'observateur sur $y_1(\tau)$

type  zc1p        ADDER PLUS PLUS PLUS;
type  wo_zc1p     CONST OMEGA_o;

link              h10,       I1;
link              h11,       zc1p;
link              I1,        zc1p;
link              psi1t_tau, zc1p;
link              zc1p,      zc1;
link              zc1p,      wo_zc1p;

// calcul des erreurs d'observation sur $\widehat{x}_1(\tau)$

type  dz2         ADDER PLUS MOINS; // en dynamique
type  h20         CONST PARA_H20;
type  h21         CONST PARA_H21;

link              x1c,       dz2;
link              zc2,       dz2;
link              dz2,       h20;
link              dz2,       h21;

// calcul $\tilde{\Psi}_2(\tau)$

type  k4_lc3      CONST PARA_k4;
type  k1_lc4      CONST PARA_k1;
type  k5_lc2_zc3  CONST PARA_k5;
type  psi2t       ADDER PLUS MOINS MOINS;
type  psi2t_tau   CONST OMEGA_3_M1;

link              lc3,       k4_lc3;
link              lc4,       k1_lc4;
link              lc2_zc3,   k5_lc2_zc3;
link              k4_lc3,    psi2t;
link              k1_lc4,    psi2t;
```

```
link              k5_lc2_zc3,psi2t;
link              psi2t,     psi2t_tau;

// calcul du corps de l'observateur sur $\widehat{x}_1(\tau)$

type  zc2p        ADDER PLUS PLUS PLUS;

link              h20,       I2;
link              h21,       zc2p;
link              I2,        zc2p;
link              psi2t_tau, zc2p;
link              zc2p,      zc2;

// calcul $\dot{\hat{z}}_1(t)$

type  k2_g3c      CONST PARA_k2;
type  l2c_g3c     MULTIPLIER 2;
type  k3_l2c_g3c  CONST PARA_k3;
type  z1cp        ADDER PLUS PLUS MOINS;

link              g3c,       k2_g3c;
link              g3c,       l2c_g3c;
link              l2c,       l2c_g3c;
link              l2c_g3c,   k3_l2c_g3c;
link              k1_lc1,    z1cp;
link              k2_g3c,    z1cp;
link              k3_l2c_g3c,z1cp;

// calcul $\dot{\hat{n}}(t)$

type  z1cp_x1c    MULTIPLIER 2;
type  wo_zc1_xc2  MULTIPLIER 2;
type  ncp         ADDER PLUS PLUS;

link              z1cp,      z1cp_x1c;
link              x1c,       z1cp_x1c;
link              zc1,       wo_zc1_xc2;
link              wo_xc2,    wo_zc1_xc2;
link              z1cp_x1c,  ncp;
link              wo_zc1_xc2,ncp;

// calcul $\dot{\hat{d}}(t)$

type  df2_z1cp    MULTIPLIER 2;
type  wo_dfc1_xc2 MULTIPLIER 2;
type  dcp         ADDER PLUS PLUS;

link              z1cp,      df2_z1cp;
link              df2,       df2_z1cp;
link              dfc1,      wo_dfc1_xc2;
link              wo_xc2,    wo_dfc1_xc2;
link              df2_z1cp,  dcp;
link              wo_dfc1_xc2, dcp;

// calcul $\dot{\hat{\ell}}_2(t)$

type  ncp_dl2c    MULTIPLIER 2;
type  nl2c_dcp    MULTIPLIER 2;
type  nl2cp       ADDER PLUS MOINS;
```

```
type  dl2c_dl2c   MULTIPLIER 2;
type  l2cp        DIVIDER;

link              ncp,       ncp_dl2c;
link              dl2c,      ncp_dl2c;
link              nl2c,      nl2c_dcp;
link              dcp,       nl2c_dcp;
link              ncp_dl2c,  nl2cp;
link              nl2c_dcp,  nl2cp;
link              dl2c,      dl2c_dl2c;
link              dl2c,      dl2c_dl2c;
link              nl2cp,     l2cp;
link              dl2c_dl2c, l2cp;

// calcul $\dot{\hat{\ell}}_3(t)$

type  u22_dSo     MULTIPLIER 2;
type  wo_u21_xc2  MULTIPLIER 2;
type  l3cp        ADDER PLUS MOINS;

link              dSo,       u22_dSo;
link              u22,       u22_dSo;
link              u21,       wo_u21_xc2;
link              wo_xc2,    wo_u21_xc2;
link              u22_dSo,   l3cp;
link              wo_u21_xc2, l3cp;

// calcul $\dot{\hat{\ell}}_4(t)$

type  u12_x1c     MULTIPLIER 2;
type  wo_u11_xc2  MULTIPLIER 2;
type  l4cp        ADDER PLUS PLUS;

link              x1c,       u12_x1c;
link              u12,       u12_x1c;
link              u11,       wo_u11_xc2;
link              wo_xc2,    wo_u11_xc2;
link              u12_x1c,   l4cp;
link              wo_u11_xc2, l4cp;

// calcul $\dot{\hat{z}}_3(t)$

type  k6_l6c      CONST PARA_k6;
type  k7_g3c      CONST PARA_k7;
type  k8_l2c_g3c  CONST PARA_k8;
type  z3cp        ADDER PLUS MOINS MOINS PLUS;

link              l6c,       k6_l6c;
link              g3c,       k7_g3c;
link              l2c_g3c,   k8_l2c_g3c;
link              k9_l5c,    z3cp;
link              k6_l6c,    z3cp;
link              k7_g3c,    z3cp;
link              k8_l2c_g3c, z3cp;

// calcul $\tilde{\psi}(\tau)$ et sa dérivée

type  k4_l3cp     CONST PARA_k4;
type  k1_l4cp     CONST PARA_k1;
```

```
 type  g3c_l2cp    MULTIPLIER 2;
 type  l2c_z3cp    MULTIPLIER 2;
 type  k5_g3c_l2cp CONST PARA_k5;
 type  k5_l2c_z3cp CONST PARA_k5;
 type  psit_t      ADDER PLUS MOINS MOINS MOINS;
 type  psit_tau    CONST OMEGA_o_M2;

 link              l3cp,      k4_l3cp;
 link              l4cp,      k1_l4cp;
 link              g3c,       g3c_l2cp;
 link              l2cp,      g3c_l2cp;
 link              l2c,       l2c_z3cp;
 link              z3cp,      l2c_z3cp;
 link              g3c_l2cp,  k5_g3c_l2cp;
 link              l2c_z3cp,  k5_l2c_z3cp;
 link              k4_l3cp,   psit_t;
 link              k1_l4cp,   psit_t;
 link              k5_g3c_l2cp, psit_t;
 link              k5_l2c_z3cp, psit_t;
 link              psit_t,    psit_tau;

 // calcul du corps de l'observateur sur $y_2(\tau)$

 type  xc2p        ADDER PLUS PLUS PLUS;
 type  wo2_xc2p    CONST OMEGA_o_2;  // traduire $\widecheck{x}_{22}(\tau)$ en temporel
 type  x1cp        ADDER PLUS PLUS;

 link              h0,        I0;
 link              h1,        xc2p;
 link              I0,        xc2p;
 link              psit_tau,  xc2p;
 link              xc2p,      xc2;
 link              xc2p,      wo2_xc2p;
 link              h2,        x1cp;
 link              xc2,       x1cp;
 link              x1cp,      x1c;
}

class omba c_gene c_obs_mba { // classe observateur

 // liens entrées-filtrées et variables d'états

 link              omba.c_gene.u[0],omba.u11;
 link              omba.c_gene.u[1],omba.u21;
 link              omba.c_gene.u[2],omba.u31;
 link              omba.c_gene.u[3],omba.u41;
 link              omba.c_gene.up[0],omba.u12;
 link              omba.c_gene.up[1],omba.u22;

 // traduction chrono par pas de 8s en heures

 type  ts          LINK_IN;
 type  th          CONST S2H; // la constante de conversion en heures

 link              ts,        th;
 link              th,        omba.heures;

 // calcul de $\Delta z_2(\tau) = z_2(\tau) - \hat{x}_1(\tau)$
 // calcul de $\Delta z_3(\tau) = z_3(\tau) - \check{z}_3(\tau)$
```

```
type  z2          LINK_IN;
type  z3          LINK_IN;
type  dz2         ADDER PLUS MOINS;
type  dg2         ADDER PLUS MOINS;
type  dz3         ADDER PLUS MOINS;
type  dg3         ADDER PLUS MOINS;

link              z2,        dz2;
link              z2,        dg2;
link              z3,        dz3;
link              z3,        dg3;
link              omba.zc2,  dz2;
link              omba.x1c,  dg2;
link              omba.zc3,  dz3;
link              omba.g3c,  dg3;

// fichiers des variables d'état

type  fich-zc1    FILE_OUT zc1.dat2;
type  fich-zc2    FILE_OUT zc2.dat2;
type  fich-zc1p   FILE_OUT zc1p.dat2;
type  fich-x1c    FILE_OUT x1c.dat2;
type  fich-xc2    FILE_OUT xc2.dat2;
type  fich-xc2p   FILE_OUT xc2p.dat2;
type  fich-g3c    FILE_OUT g3c.dat2;
type  fich-zc3p   FILE_OUT zc3p.dat2;
type  fich-zc3    FILE_OUT zc3.dat2;

link              th,        fich-zc1;
link              omba.zc1,  fich-zc1;
link              th,        fich-zc2;
link              omba.zc2,  fich-zc2;
link              th,        fich-zc1p;
link              omba.wo_zc1p, fich-zc1p;
link              th,        fich-x1c;
link              omba.x1c,  fich-x1c;
link              th,        fich-xc2;
link              omba.xc2,  fich-xc2;
link              th,        fich-xc2p;
link              omba.wo2_xc2p, fich-xc2p;
link              th,        fich-g3c;
link              omba.g3c,  fich-g3c;
link              th,        fich-zc3p;
link              omba.wo_zc3p, fich-zc3p;
link              th,        fich-zc3;
link              omba.zc3,  fich-zc3;

// les fichiers des distances d'état

type  fich-dz1    FILE_OUT dz1.dat2;
type  fich-dy1    FILE_OUT dy1.dat2;
type  fich-dz2    FILE_OUT dz2.dat2;
type  fich-dg2    FILE_OUT dg2.dat2;
type  fich-dz3    FILE_OUT dz3.dat2;
type  fich-dg3    FILE_OUT dg3.dat2;

link              th,        fich-dz1;
link              omba.dz1,  fich-dz1;
```

```
 link              th,        fich-dy1;
 link              omba.dy1,  fich-dy1;
 link              th,        fich-dz2;
 link              dz2,       fich-dz2;
 link              th,        fich-dg2;
 link              dg2,       fich-dg2;
 link              th,        fich-dz3;
 link              dz3,       fich-dz3;
 link              th,        fich-dg3;
 link              dg3,       fich-dg3;

 // les fichiers des fonctions non-linéaires

 type  fich-l2c    FILE_OUT l2c.dat2;
 type  fich-l2cp   FILE_OUT l2cp.dat2;
 type  fich-psit   FILE_OUT psit.dat2;
 type  fich-l3cp   FILE_OUT l3cp.dat2;
 type  fich-l4cp   FILE_OUT l4cp.dat2;
 type  fich-ncp    FILE_OUT ncp.dat2;
 type  fich-dcp    FILE_OUT dcp.dat2;

 link              th,        fich-l2c;
 link              omba.l2c,  fich-l2c;
 link              th,        fich-l2cp;
 link              omba.l2cp, fich-l2cp;
 link              th,        fich-psit;
 link              omba.psit_t, fich-psit;
 link              th,        fich-l3cp;
 link              omba.l3cp, fich-l3cp;
 link              th,        fich-l4cp;
 link              omba.l4cp, fich-l4cp;
 link              th,        fich-ncp;
 link              omba.ncp,  fich-ncp;
 link              th,        fich-dcp;
 link              omba.dcp,  fich-dcp;
}
```

A.3 Définitions des objets de calculs « stationb.obs.s »

```
/*
Modèle bruité de boues activées du système physique,
2 observateurs indépendants
*/

includestation.h;         // inclure les définitions symboliques au niveau "C"
input./src/station.cls;// inclure les définitions de classes

class c_principal {

// Calcul de l'échelle des temps

 type  k-t         VALUE 1.0;
 type  t_s         INTEGRATOR 0.0 0.0 -TE;

 link              k-t, t_s;
}

object principal c_principal;
```

```
object modele_d mbadb { // objet système physique dynamique

 link              principal  principal.t_s,     modele_d  modele_d.ts;
}

object obs omba { // objet observateur modèle de boue activée

 link              principal principal.t_s,      obs  obs.ts;
 link              modele_d  modele_d.y1,        obs  obs.omba.y1;
 link              modele_d  modele_d.y2,        obs  obs.omba.y2;
 link              modele_d  modele_d.mbadb.z2,  obs  obs.z2;
 link              modele_d  modele_d.mbadb.z3,  obs  obs.z3;
}

main TE, NITERATIONS {  // $T_e$ fixé, NITERATIONS de calculs

 compute;         // lancer les calculs
}
```