Caractérisation, identification et optimisation des systèmes mécaniques complexes par mise en oeuvre de simulateurs hybrides matériels/logiciels

Sébastien Salmon

▶ **To cite this version:**

Sébastien Salmon. Caractérisation, identification et optimisation des systèmes mécaniques complexes par mise en oeuvre de simulateurs hybrides matériels/logiciels. Autre. Université de Technologie de Belfort-Montbeliard, 2012. Français. NNT : 2012BELF0179 . tel-00823562

HAL Id: tel-00823562

https://theses.hal.science/tel-00823562v1

Submitted on 17 May 2013

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

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

N° d'ordre : 179 **Année : 2012**

Université de Technologie de Belfort - Montbéliard

Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques

Thèse

Présentée pour obtenir le grade de

Docteur de l'Université de Technologie de Belfort – Montbéliard en Sciences pour l'ingénieur

Spécialité : Sciences pour l'ingénieur

Présentée et soutenue publiquement le 21 mai 2012 par

Sébastien Salmon

Caractérisation, identification et optimisation de systèmes mécaniques complexes par mise en œuvre de simulateurs hybrides matériels / logiciels

Président de jury et rapporteur : **Pr. P. Mognol, ENS Cachan**
Rapporteur : **Pr. L. Gaudiller, INSA Lyon**
Directeur de recherche : **Pr. W. Charon, UTBM – M3M**

Remerciements

Je tiens à remercier en tout premier lieu Willy Charon, qui après m'avoir recruté sur un poste d'ingénieur, m'a permis de réaliser cette thèse. Je le remercie pour sa confiance, ma liberté de choix de solutions et nos discussions tout au long de ces années.

Je remercie Pascal Mognol qui a bien voulu présider le jury de thèse ainsi que Luc Gaudiller d'avoir été rapporteur de celle-ci.

Je remercie bien sûr Pascal Aldinger pour ses corrections sur le mémoire ainsi que pour ses séances de préparation de soutenance.

Un grand merci à Béatrice Rossez pour les innombrables services rendus et à sa bonne humeur.

Je remercie aussi les deux personnes qui ont successivement partagé le bureau et qui ont réussi à me supporter : Moustapha Kerdy et Sophie Collong.

Je remercie aussi tous les membres du laboratoire aussi bien pour les discussions techniques que pour les pauses café notamment Philippe Lesage et Dominique Chamoret.

Je remercie aussi toutes les personnes que j'ai côtoyées tout au long de ces années : Coin-Coin / Clop' et les parties de belote, Marie pour ces nombreuses soirées passées en station au lieu de rédiger, les habitués ou non de la Bergerie, les gens du BdF, …

J'ai sûrement oublié un nombre important de personnes mais sachez que je vous remercie aussi.

A mon père,

Introduction

Cadre de recherche

Cette thèse a été développée dans l'équipe Systèmes Mécaniques Adaptatifs (SMA) du laboratoire Mécatronique, Méthodes, Modèles et Métiers (M3M) EA3318 de l'Université de Technologie de Belfort - Montbéliard (UTBM) sous la direction de Willy CHARON, professeur des Universités.

L'équipe SMA, dirigée par Willy CHARON, développe plusieurs axes de recherches :

- la modélisation probabiliste des systèmes mécatroniques;
- l'expérimentation des systèmes mécaniques complexes;
- la fiabilité, sûreté de fonctionnement, la durabilité et la gestion des pannes.

Les applications principales sont les systèmes à piles à combustibles pour le domaine du transport ainsi que les dispositifs mécaniques actifs pour le transport et pour le biomédical.

Cette thèse se place dans le cadre de cette dernière application en s'appuyant sur un projet FUI comme illustration. Ce projet a pour but de concevoir un nouveau type de valve implantable dans le corps humain pour le traitement de l'hydrocéphalie. Il s'agit d'une valve active permettant un réglage fin de la pression intracrânienne de façon aisée et fiable dans le temps.

Objectifs de la thèse

La conception de systèmes complexes, et plus particulièrement de microsystèmes complexes embarqués, posent des problèmes tels que l'intégration des composants, la consommation d'énergie, la fiabilité, délais de mise sur marché, ...

La conception mécatronique (Fig. 1) apparait alors comme étant une solution idéale pour ces systèmes. Cette méthode allie fortement simulations, expérimentations, interactions entre sous-systèmes et cycles de reconception à tous les niveaux afin d'obtenir un produit plus performant, mieux intégré et de réduire les délais de mise sur marché.

Cette approche a donc été utilisée lors de l'exécution d'un projet FUI permettant alors d'apporter une aide aux concepteurs en termes de choix technologiques aussi bien que de validations de paramètres.

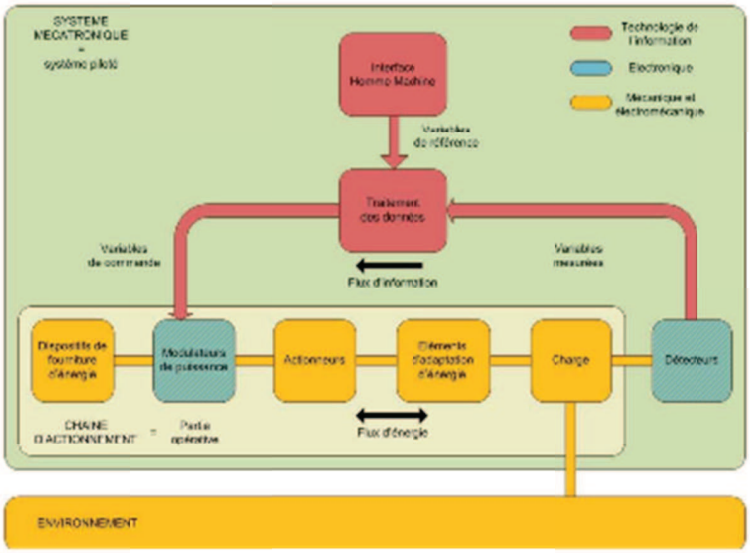

Figure 1 : Architecture générale d'un système mécatronique, source [31]

Cette thèse a pour but de trouver des méthodes de caractérisation, d'identification de paramètres ainsi que d'optimisation de systèmes mécatroniques actifs par la constitution de modèles numériques, de bancs d'expériences numériques, physiques et hybrides dans un cadre bien précis : celui de l'actionneur, de sa commande ainsi que de la constitution générale de la boucle fermée du système mécatronique.

Afin d'expliciter ces démarches, trois thématiques de mise en œuvre sont abordées : la simulation, l'hybridation et l'expérimentation.

Le lecteur devrait trouver ici des pistes pour la résolution des problèmes y afférant.

Présentation de la structure du mémoire

Le mémoire se compose de deux grandes parties : les quatre premiers chapitres présentent une approche globale de la problématique et de son traitement, et les trois derniers sont une mise en œuvre de cette approche sur un problème concret.

Le chapitre 1 met en place le cadre de l'étude des systèmes mécatroniques. Ce chapitre définit les frontières et constituants de ces systèmes ainsi que les méthodes de conception associées.

Dans la continuité du premier chapitre, le chapitre 2 définit ce qu'est une simulation numérique ainsi que son apport pour l'étude des systèmes mécatroniques. Les langages existants sont revus ainsi que les outils associés. En complément de cet aspect purement logiciel sont aussi présentés les matériels de calcul pouvant servir de support aux outils présentés.

Le chapitre 3 présente une démarche pour la caractérisation et l'identification inverse des paramètres physiques d'un système mécatronique. Cette démarche passe par une phase

d'observation conduisant à la création d'un modèle générique rassemblant les différents processus observés. Puis la structure d'un banc de test du système est étudiée au travers du choix de divers capteurs ainsi que leurs placements. Enfin trois solutions sont présentées pour traiter le problème de l'identification des paramètres : l'identification directe, le modèle inverse et l'identification inverse.

Le chapitre 4 présente différents outils nécessaires à une procédure d'identification inverse par des méthodes d'optimisation. Cette partie pose les différentes fonctions utilisées pour cette technique puis une revue de différents procédés est réalisée. Une attention particulière est portée sur la méthode méta-heuristique d'optimisation par essaim de particules en proposant deux améliorations de cet algorithme.

Le chapitre 5 met en application les différentes notions abordées dans les chapitres précédents sur un sous-système actionneur réel. La construction d'un modèle, puis la mise en place d'un système d'optimisation permettent de réaliser une identification inverse de paramètres de frottement mais aussi de réaliser une optimisation d'un signal de commande pour l'amélioration des performances de cet actionneur. Ces deux points sont réalisés par simulation informatique.

Le chapitre 6 présente le concept d'hybridation des processus de simulation informatique avec des composants réels pour, par exemple, la création de loi de commande et de contrôle. Dans cette partie sont abordés les concepts de systèmes embarqués, de temps-réel, d'instrumentation virtuelle et enfin de Hardware et Software in the Loop. Une application sera présentée afin d'illustrer l'intérêt de telles structures de simulation hybride.

Le chapitre 7 reprend une des méthodes d'optimisation méta-heuristique afin de montrer la possibilité que peut offrir le HiL / Sil en terme d'optimisation sur un système mécatronique. Cette partie dégage non seulement les différents avantages, mais aussi les difficultés inhérentes, que cette méthode peut offrir en termes de temps et performances par rapport à une méthode informatique classique.

Table des matières

1 Systèmes mécatroniques : Définitions et démarche de conception

Sommaire

Le but de cette partie est de circonscrire, par une revue de définitions qui ne se prétend évidemment pas exhaustive, ce qu'est un système mécatronique. Ensuite seront abordées des méthodes de conception pour de tels systèmes, permettant alors de choisir une démarche de conception adaptée.

1.1 Définition d'un système mécatronique

Avant de commencer la réflexion qui sera menée au cours de cette thèse, il convient de définir ce qu'est un système mécatronique.

Le mot mécatronique provient d'un assemblage de deux mots : electronics et mechanics afin de former le terme mechatronics dont la traduction française est mécatronique. Ce terme a, semble-t-il, été cité pour la première fois par un membre de YASKAWA ELECTRICS en 1969 [47]. Cependant, au fil des années, les produits mécatroniques se sont complexifiés en intégrant de plus en plus de domaines différents tels que les théories de contrôle des systèmes, des intelligences artificielles ou encore l'hydraulique, la thermique … Ainsi, la mécatronique peut apparaître comme une mise en œuvre communes des différents domaines d'ingénierie.

Diverses définitions sont extraites :

- par le journal Mechatronics :

Mechatronics in its fundamental form can be regarded as the fusion of mechanical and electrical disciplines in modern engeneering process. It is a relatively new concept to the design of systems, devices and products aimed at achieving an optimal balance between basic mechanical structures and its overall control. [57]

- par la publication IEEE Transactions on Mechatronics :

Mechatronics is the synergetic combinaison of mechanical engineering with electronics and intelligent computer control in the design and manifacturing of industrila products ans processes. [49]

- par IFAC Technical Committee on Mechatronics systems :

Many technical processes and products in the area of mechanical and electrical engineering show an increasing integration of mechanics with electronics and information processing. This integration is between the components (hardware) and the information-driven function (software), resulting in integrated systems called mechatronic systems. [13]

- par l'AFNOR :

Une démarche visant l'intégration en synergie de la mécanique, l'électronique, l'automatique et l'informatique dans la conception et la fabrication d'un produit en vue d'augmenter et/ou d'optimiser sa fonctionnalité. [88]

Il est possible d'extraire de ces définitions les caractéristiques suivantes :

• un système mécatronique est un système réunissant au minimum le domaine de l'électronique et de la mécanique

• un système mécatronique est un système intégré : il doit exister une relation forte entre les différents domaines (design, flux informatif ou énergétiques ….). Ce degré de relation permet alors une classification des systèmes mécatroniques du système automatisé au système intelligent interconnecté [88]

• une approche mécatronique de conception (ou de re-conception) d'un système peut participer à l'amélioration des performances comme indiqué dans la dernière définition. On peut citer par exemple le passage d'un système de stockage informatique à bande magnétique possédant des temps d'accès très longs (de l'ordre de la seconde à la minute) aux disques durs avec des temps d'accès rapides (de l'ordre de la milliseconde)

• mais aussi cette approche peut apporter de nouvelles fonctionnalité par l'innovation. On peut prendre pour exemple les systèmes de freinage de type ABS permettant la détection de la perte d'adhérence des pneumatiques ou encore l'ESP qui est un estimateur et correcteur de trajectoire

1.2 Architecture générale d'un système mécatronique

D'après les définitions précédentes, il est possible de d'établir un schéma global de l'architecture d'un système mécatronique qui permet de décrire les différents flux d'interconnexion des différents composants (Fig. 11 extrait de [37]).

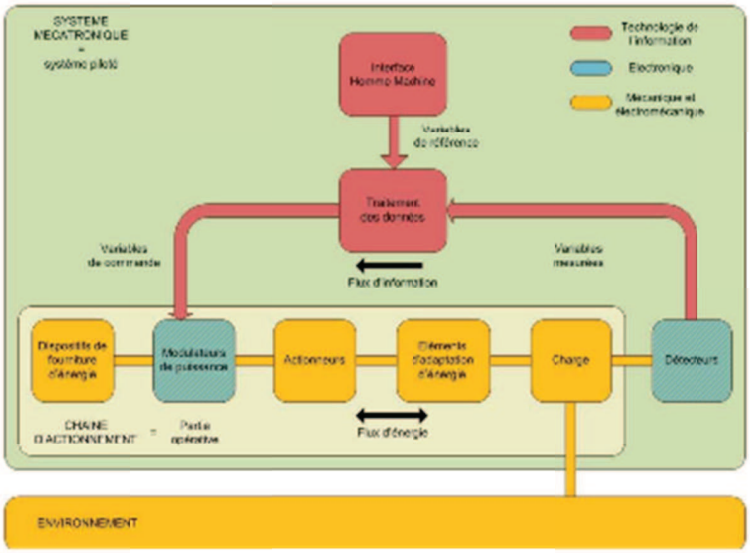

Figure 2 : Architecture générale d'un système mécatronique

On y retrouve tous les ensembles précédemment cités tels qu'actionneurs, capteurs, traitement de l'information, … reliés entre eux par des flux éventuellement bidirectionnels que sont les flux d'interaction.

1.3 Conception des systèmes mécatroniques

Un système mécatronique, dans les définitions précédentes, est un système dont tous les sous-systèmes ou domaines physiques sont étroitement liés les uns aux autres. Cette conception entraine donc de devoir modifier les démarches de conceptions classiques vers une démarche adaptée à la mécatronique.

1.3.1 D'une conception en ligne vers une conception en étoile

Originellement la conception d'un produit s'est faite de façon linéaire en découpant explicitement les domaines d'interventions des équipes de concepteurs. Cette découpe peut se faire suivant des domaines d'expertise mécanique, électrique, fluidique … ou bien par métier comme motoriste, carrosserie …. Cette approche est donc séquentielle dans le sens où une équipe intervient sur un problème puis une autre la remplace sur le problème suivant (Fig. 3).

Figure 3 : Approche de conception en ligne

Il en découle alors un temps de conception relativement long et qui ne permet pas forcément une bonne prise en compte des interactions entre les différents composants (problème du notamment à la difficulté de la transmission de l'information d'où l'apparition du Knowledge Management).

La conception en mécatronique fait appel à un système en étoile (Fig. 4). Les différentes équipes travaillent alors simultanément et de façon concertée sur le produit. Cette démarche, bien que plus difficile à gérer pour le chef de projet, permet de réduire de façon importante le temps de développement du produit. Cette forme de conception permet de mettre en place un point de vue système du produit à concevoir et de ne plus aborder la conception comme une succession de sous-systèmes à assembler.

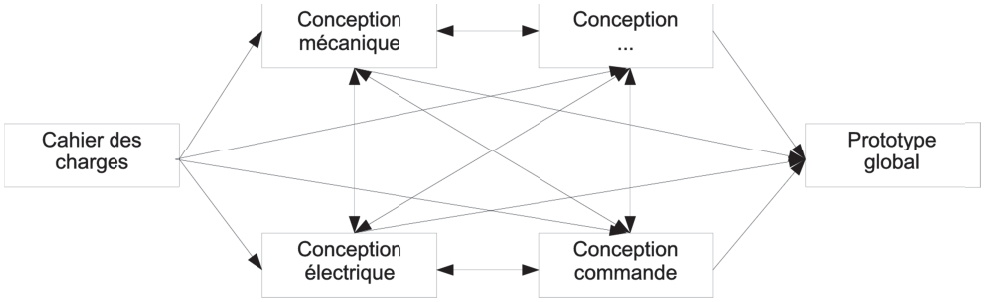

Figure 4 : Approche de conception étoile

Cette approche système de conception permet d'optimiser:

- de façon pertinente l'espace occupé par les sous-ensembles (très important pour les microsystèmes)
- les interactions entre sous-systèmes (très important pour la gestion des flux notamment énergétiques)
- le fonctionnement du système par une optimisation globale et non pas par une somme d'optimisation des sous-systèmes
- enfin le temps de conception

1.3.2 Le cycle en V , complément de la conception parallèle

Le cycle en V (Fig. 5, extrait de [31]) est à l'origine une méthode de fonctionnement utilisée dans le cadre du développement de logiciels informatiques et a été adapté à la conception mécatronique.

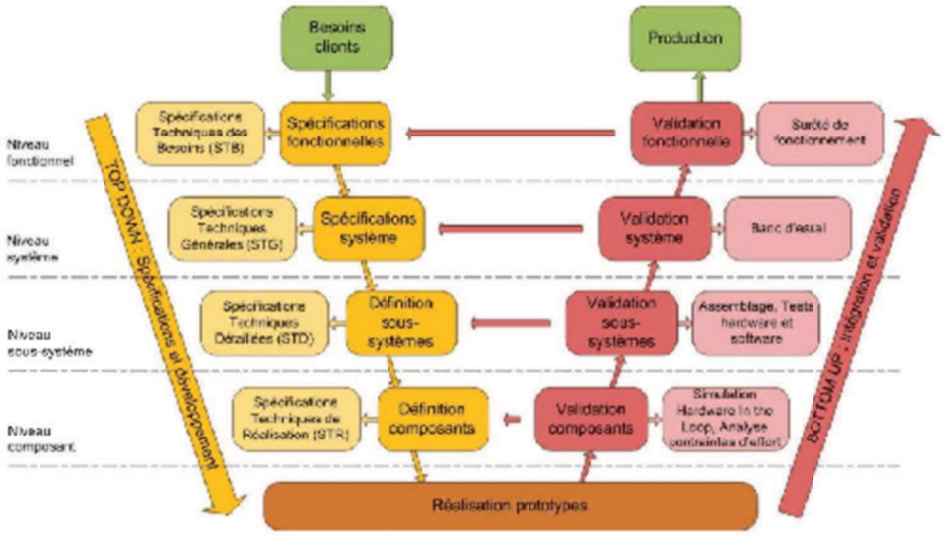

Figure 5 : Diagramme d'un cycle en V

Ce cycle se présente sous forme de plusieurs niveaux d'abstraction en partant de l'expression du besoin, des spécifications fonctionnelles et systèmes, jusqu'aux composants permettant la réalisation du produit. Cette phase descendante de spécification et de conception permet de conserver à l'esprit une conception système et donc de prendre directement en compte toutes les interactions.

A chaque niveau d'abstraction est associée une démarche ascendante d'intégration et de validation. Cette phase de validation se fait par l'intermédiaire de bancs de tests ou de simulations pour vérifier que l'ensemble testé répond bien aux spécifications techniques et que son intégration dans le système se fait dans les normes définies dans les spécifications. Dans le cas où l'une de ces validations échoue, se met alors en place une boucle de correction qui ramène à l'étape de définition des spécifications autorisant le cycle de correction.

Dans cette figure, plusieurs éléments importants sont cités :

- simulation pour validation aux différents niveaux
- l'expérimentation des ensembles créés
- le mixage de l'expérimentation et de la simulation par le hardware in the loop

Ces différents points seront abordés plus en profondeur au cours de ce document.

Cependant, ce cycle doit être complété par l'intérêt de plus en plus grand de l'outil de simulation pour le prototypage virtuel. La dernière étape qui consiste à la réalisation de

prototypes est ainsi neutralisée dans ce nouveau processus. Mais cette technique ne peut être valide que pour des composants ou sous-systèmes parfaitement connus et modélisables.

1.4 Conclusion

Afin de mettre en place une démarche pertinente pour l'étude des système mécatroniques, il apparait que le choix d'un cycle de conception en V couplé à une démarche en parallèle permet d'obtenir un cadre fiable par la prise en compte des différents niveaux d'abstraction, et des tests de validation obligatoires avec la rapidité d'une démarche en parallèle.

Ce choix sera appliqué dans les différentes actions menées dans cette thèse.

2 Simulation : langages, outils et matériel de simulation pour la mécatronique

Sommaire

Le but de ce chapitre est d'introduire le concept de simulation puis de présenter différentes démarches pour la modélisation des systèmes mécatroniques. Une fois cette présentation réalisée, il est fait une revue non exhaustive des principaux moyens de modélisation que sont les langages utilisés par les principaux simulateurs (ou logiciels de simulation) capables de traiter des problèmes mécatroniques. Pour illustrer cette revue, un moteur à courant continu commandé par l'inducteur servira d'exemple. Ce composant électromécanique, dans une modélisation simple, présente l'avantage d'allier deux domaines d'études et est linéaire temporellement invariant (systèmes LTI). Une comparaison des différentes modélisations est alors possible en utilisant cet exemple commun. En complément de ce point de vue logiciel, une revue du matériel supportant des outils de simulation est aussi réalisée.

2.1 Simulation

2.1.1 Définition

La simulation, de façon générale, est un processus permettant de représenter de façon mathématique un phénomène réel afin de le reproduire et de pouvoir prédire le comportement de celui-ci suivant différentes données d'entrées. Une définition de la simulation donnée par le Larousse est la suivante [39]:

Représentation du comportement d'un processus physique, industriel, biologique, économique ou militaire au moyen d'un modèle matériel dont les paramètres et les variables sont les images de ceux du processus étudié.

Les modèles de simulation prennent le plus souvent la forme de programmes d'ordinateurs auxquels sont parfois associés des éléments physiques réels, on parle alors d'hybridation.

Cette définition est intéressante car elle permet de mettre en relief les différents aspects de l'action de simuler :

Comportement d'un processus physique :

Cette notion de comportement découle d'une observation d'un processus qui se veut physique c'est-à-dire réel et matériel. Cela implique alors la définition des bornes du processus étudié, ainsi qu'une observation, donc mesure, de grandeurs physiques du processus (ou phénomène).

Représentation :

La représentation est la description, par un moyen tel qu'un langage, de quelque chose. Ainsi, il est possible d'obtenir des représentations différentes d'un même phénomène suivant le

type de langage employé (représentation par éléments finis ou par équation d'une barre en traction par exemple). Il n'y a donc pas forcément unicité de la représentation.

Paramètres et variables :

Un paramètre est une constante (par exemple une constante physique) et est donc un invariant du modèle, contrairement à la variable qui peut par exemple dépendre du temps. On a alors commencé à définir des classes d'objet utilisées lors de la simulation.

Modèle matériel et éléments physiques réels :

Les modèles utilisés actuellement sont calculés de façon numérique grâce à l'augmentation rapide des capacités de calculs [43, 74] et à l'amélioration des techniques mises en œuvre. Cependant, il est possible d'établir une différentiation sur les moyens de calculs (utilisation de cluster [69], ordinateurs classiques ou bien systèmes embarqués de type temps réel). Les éléments de calculs physiques réels ont été utilisés dans les années 50 pour palier la déficience des moyens de calculs numériques de l'époque, on peut notamment citer l'utilisation de ce genre de système pour réaliser des fonctions d'intégration. Cependant, cet aspect n'a pas complètement disparu par l'introduction des techniques de type Hardware in the Loop (HIL), autorisant alors l'introduction d'un matériel et de son environnement dans une boucle de simulation numérique.

L'utilisation de l'outil de simulation et sa mise en œuvre dans le domaine mécatronique permet de façon globale de réaliser les actions suivantes :

- recherche sur les phénomènes physiques mis en jeux et leurs interactions
- analyse comportementale des sous-systèmes et du système entier
- conception et intégration d'un sous-système dans l'environnement système
- optimisation du système et /ou des sous-systèmes

Ces différents points sont développés dans d'autres chapitres.

2.1.2 Axes de simulation

Plusieurs axes de simulation sont disponibles pour l'utilisateur :

- les simulations multi-niveaux
- les simulations multi-domaines
- les co-simulations
- les simulations avec prise en compte de systèmes réels (HIL)

Les simulations multi-niveaux :

La simulation multi-niveaux (ou multi-échelles) permet de réaliser des simulations aussi bien à des échelles microscopiques qu'à des échelles macroscopiques, les deux échelles étant mises en relation. L'intérêt de cette méthode est la possibilité donnée au concepteur de pouvoir, soit créer des modèles macroscopiques à partir de phénomènes microscopiques (exemple : détermination des matrices de rigidité d'un matériau composite à partir d'un volume élémentaire par éléments finis), ou bien à partir de connaissance du modèle macroscopique de pouvoir déterminer un modèle microscopiques (exemple : détermination des contraintes sur une pièce d'un avion à partir des contraintes exercées sur l'ensemble de l'avion Fig. 6). De telles modélisations sont utilisées de façon intensive par, notamment, les météorologues [58]. Le premier cas de figure est une approche dite ascendante nécessitant la mise en œuvre de méthodes d'homogénéisation, alors que le second cas est une approche descendante demandant d'imposer des conditions limites du modèle macroscopique au microscopique. Ces deux approches sont complémentaires et permettent à l'utilisateur de construire uniquement les modèles adaptés au niveau de détail souhaité. Cela permet ainsi un gain en termes de coût de calcul non négligeable.

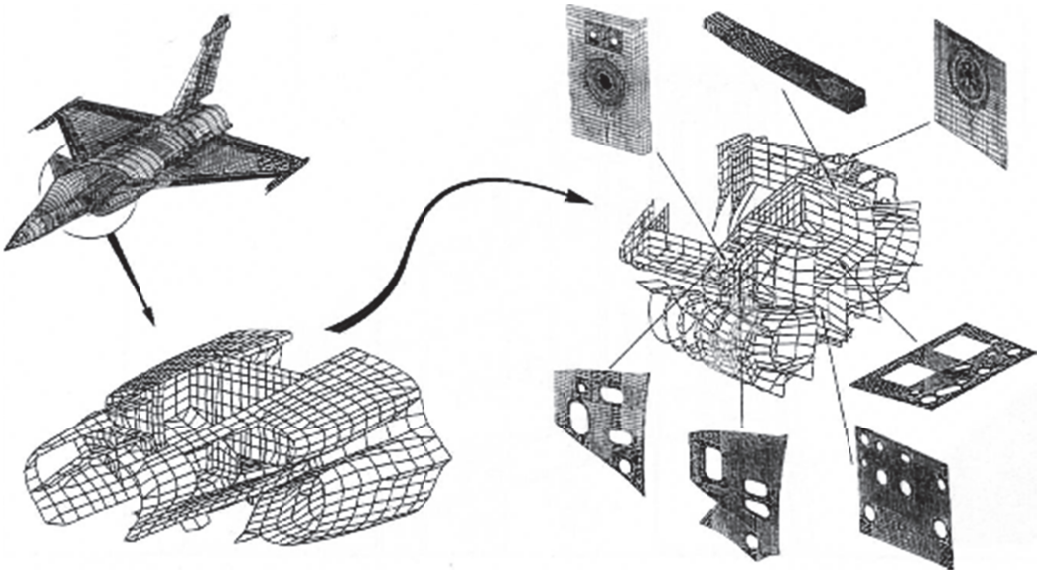

Figure 6 : Exemple de simulation multi-niveau, source Dassault Aviation

Les simulations multi-domaines :

Une simulation multi-domaines est tout à fait adaptée à la mécatronique car elle permet la prise en compte des différents domaines de la physique (mécanique, électrique, informatique, …). La difficulté de ce type de simulation est la gestion des interfaces entre les

différents domaines ou plutôt la gestion des interactions (par exemple des efforts mécaniques interne à un vérin hydraulique peuvent être source de perturbation pour le circuit hydraulique). Zenati présente dans sa thèse [87] l'exemple d'un microsystème autonome mettant en œuvre différents domaines Fig. 7.

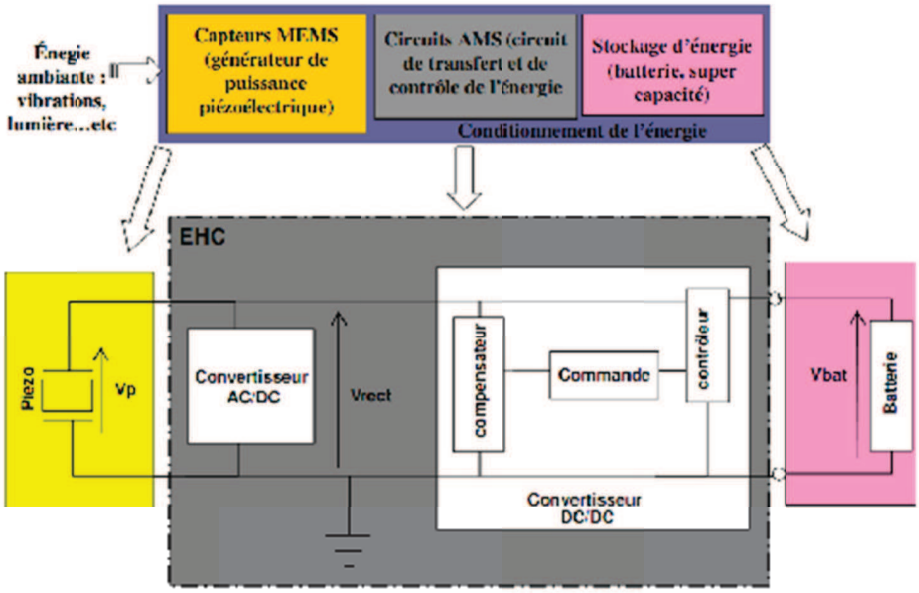

Figure 7 : Exemple de simulation multi-domaine, source [87]

Les co-simulations :

Une co-simulation est la mise en commun de données issues de sous-simulations effectuées à partir de simulateurs de natures différentes (exemple d'un simulateur mécanique par éléments finis couplé avec un simulateur électrique de type SPICE [50]). La difficulté de la co-simulation est le transfert de données entre les différents simulateurs. Finalement on se retrouve dans le même cas de figure que dans une simulation multi-domaine. Un exemple de co-simulation peut être trouvé dans [64] présentant le couplage de différents logiciels de simulation, dont notamment un logiciel de calcul par éléments finis avec un autre de simulation 1D, pour l'étude d'arbres de transmission.

Les simulations avec prise en compte de sous-systèmes réels

L'introduction de systèmes réels dans le cadre d'une simulation peut se faire lorsqu'il n'est pas évident de réaliser une modélisation correcte de ce système. Cela peut se produire

lorsque le système est composé de parties à comportement fortement non-linéaires ou bien encore soumis à des phénomènes aléatoires peu modélisables aussi bien d'origines internes qu'environnementales. Des exemples de systèmes HIL sont présentés dans [54, 42], respectivement pour l'étude de la motorisation de véhicules équipés de pile à combustible et pour le réglage de l'injection de moteur thermique. Plus précisément, dans le premier cas, un modèle de véhicule est créé puis les sous-parties sont remplacées par des prototypes construits à une échelle plus réduite (ex: PAC réduite) permettant de limiter les coûts tout en conservant des comportements réels identiques. Une remise à l'échelle des mesures est réalisée par le modèle ce qui permet alors de simuler le reste des composants non-prototypés. Dans le second cas, un banc de test comportant un moteur thermique est créé afin de tester une commande d'injecteur adaptative permettant de limiter les émissions de polluants. La mise en place de tels systèmes peut s'avérer coûteuse car il est nécessaire de disposer non seulement du système réel mais aussi de l'instrumentation adéquate interfacée avec le simulateur. Cependant, cela peut s'avérer très intéressant lorsque la simulation du système réel est plus coûteuse en temps qu'une expérimentation sur celui-ci, notamment dans le cadre d'optimisation de signaux de commande (point qui est abordé plus loin).

2.2 Langages de simulation

2.2.1 Structure générale d'un système à simuler

Un système mécatronique se compose d'éléments informatiques, électriques, mécaniques, …. Ces éléments se comportent de différentes façons:

- un élément informatique ou programme se comporte de façon séquentielle suivant des instructions de passage d'une étape à une autre, c'est donc un système discontinu lié à des expressions conditionnelles
- les autres éléments sont des composants physiques réels, ils sont donc soumis à des équations différentielles (éventuellement non-linéaires) complétées par des équations de contraintes

2.2.2 Causalité des systèmes

La causalité est un postulat en physique définissant qu'un effet ne peut précéder la cause. Ce principe est respecté directement dans la forme de réalisation des modèles dans la plupart des langages de modélisation. Cet aspect introduit alors la nécessité de réaliser des boucles de rétroaction qui peuvent alors rendre un modèle peu lisible. Il est alors intéressant de pouvoir se doter de langages, dits acausaux, autorisant une représentation plus simple des systèmes. Ce point est notamment utilisé par le langage Modelica qui est abordé par la suite. La causalité du modèle généré est alors restaurée automatiquement par les solveurs (implicites ou

explicites) suivant les données disponibles aux entrées du modèle : dans tout le modèle créé, une variable est aussi bien une donnée d'entrée qu'un résultat, et sa spécialisation n'est connue que par la propagation des caractéristiques des données d'entrée du problème. Cette spécificité introduit une programmation implicite pour l'utilisateur autorisant une compacité et clarté de représentation intéressante.

2.2.3 Fonctions de transfert

Les fonctions de transfert sont largement utilisées dans le cadre des études de systèmes asservis aussi bien purement analogiques, numériques ou encore hybrides. Cette représentation se fait à partir de la connaissance des équations différentielles entre une entrée et une sortie du système étudié, grâce à l'utilisation de la transformée de Laplace (variable p ou s) et de sa discrétisation (variable z) dans le cas des systèmes numériques ou hybrides. Il est également possible de pouvoir déterminer la fonction de transfert expérimentale d'un système par l'utilisation, par exemple, d'un Dirac sur l'entrée et la transformée de Fourier de la sortie. Les fonctions de transfert sont bien adaptées aux systèmes linéaires mais sont relativement dépourvues face à des systèmes non-linéaires. Il existe cependant des techniques permettant de traiter certaines non-linéarités afin de revenir à un système classique. Un avantage des fonctions de transfert est la bonne maitrise des correcteurs, aussi bien numériques qu'analogiques (PID, RST, Zdan, prédicteurs de Smith, …), qui permettent une mise en œuvre rapide de systèmes asservis (fonctionnement en boucle fermée). Un inconvénient de cette modélisation peut être sa causalité ainsi qu'un éventuel besoin de recalculer les fonctions de transfert des correcteurs dès l'ajout ou modification d'une fonction dans la boucle ouverte.

Exemple : moteur à courant continu commandé par l'inducteur

Soit un moteur à courant continu commandé par l'inducteur. Une entrée possible du système est la commande en tension de l'inducteur et la sortie la vitesse de rotation de l'arbre du moteur car ce composant peut aussi servir de générateur électrique. Le système est à variable unique, linéaire et invariant dans le temps. Une équation différentielle à coefficients constant peut alors représenter le système.

Schéma du moteur :

Le schéma du moteur est présenté par la figure 8 et est complété par la table des paramètres associés 1 :

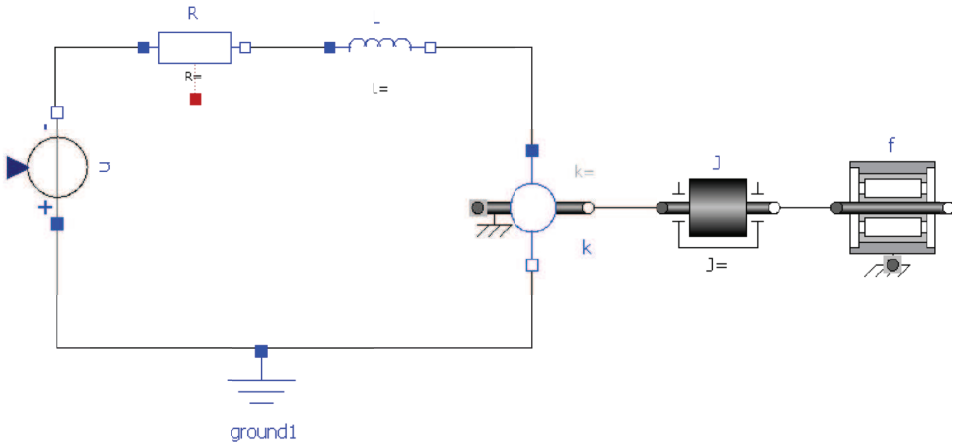

Figure 8 : Schéma du moteur à courant continu

Nom de la variable	Symbole
Couple électromagnétique	γ
Tension d'alimentation	u
Intensité dans l'inducteur	i
Inertie mécanique de rotation	J
Couple de frottement fluide	f
Constante de couplage électromécanique	k
Inductance du bobinage de l'induit	L
Résistance du bobinage de l'induit	R

Table 1 : Table des paramètres du moteur à courant continu

Mise en équation :

$$\begin{cases} u = R.i + L\frac{di}{dt} \\ J\frac{d\omega}{dt} = -f\omega + \gamma \\ \gamma = k.i \end{cases} \quad (1)$$

Ce qui permet d'obtenir l'équation différentielle suivante d'ordre 2 à coefficients constants :

$$\frac{d^2\omega}{dt^2} + (\frac{f}{J} + \frac{R}{L})\frac{d\omega}{dt} + \frac{Rf}{LJ}\omega = \frac{k}{LJ}u \quad (2)$$

On utilise alors la transformée de Laplace :

$$\begin{aligned} \mathcal{L}[e(t)] &= \int_0^\infty e^{-pt}e(t)dt \\ \mathcal{L}\left[\frac{de(t)}{dt}\right] &= pE(p) - e(0) \end{aligned} \quad (3)$$

qui est alors appliquée à l'équation différentielle. On pose aussi les constantes suivantes :

$$\begin{cases} a_0 = \frac{Rf}{LJ} \\ a_1 = \frac{f}{J} + \frac{R}{L} \\ b_0 = \frac{k}{LJ} \\ b_0 U(p) = p^2\Omega(p) - p\Omega(0) - \dot{\Omega}(0) + a_1(p\Omega(p) - \Omega(0)) + a_0\Omega(p) \end{cases} \quad (4)$$

En supposant les conditions initiales nulles, on peut écrire la fonction de transfert du système:

$$H(p) = \frac{U(p)}{\Omega(p)} = \frac{b_0}{p^2 + a_1 p + a_0} \quad (5)$$

L'obtention de la fonction de transfert permet alors de pouvoir calculer une réponse temporelle du système suivant une entrée donnée par transformée inverse de Laplace.

Exemple de réponse impulsionnelle du système :

$$\omega(t) = h(t) * u(t) = \int_0^\infty h(\tau)u(t-\tau)d\tau = \frac{k}{RJ - Lf}\left(e^{-\frac{f}{J}t} - e^{-\frac{R}{L}t}\right) \quad (6)$$

2.2.4 Représentation d'état

La représentation d'état est une écriture matricielle d'un système d'équations différentielles. L'intérêt de cette écriture se présente lorsque ce système d'équations différentielles est linéaire et invariant. Le système étudié peut être aussi bien continu (forme analogique) que discret (forme numérique) voire hybride. Le système d'équation est alors réécrit de manière à obtenir des vecteurs de variables d'état, de mesure et de commande ainsi que des matrices dites de dynamique (A), commande (B), observation (C) et d'action directe (D), comme indiqué dans l'équation suivante :

$$\begin{cases} \dot{X} = AX + Bu \\ Y = CX + Du \end{cases} \tag{7}$$

Une fois ces matrices déterminées, il est possible de déterminer la commandabilité et l'observabilité du système, et si ces hypothèses sont vérifiées un correcteur de placement de pôles par retour d'état est directement calculable afin de stabiliser le système en boucle fermée. Un avantage de cette représentation est qu'il n'existe pas de représentation unique d'un système et est donc adaptable par le concepteur du modèle à ses besoins. Cependant cette modélisation se fait sous forme d'entrée / sorties fixée et est donc causale.

Commandabilité

Un système est commandable si quel que soit l'état initial et final du système, il existe une commande d'entrée qui permet au système d'aller de l'état initial à l'état final. Cette condition se traduit par l'équation suivante :

$$rang(C) = rang([B\ AB \dots A^{n-1}B]) = n \tag{8}$$

où n est la dimension du vecteur d'état.

Observabilité

Un système est observable si par la mesure entre deux instants, initiaux et finaux, des entrées et des sorties, il est possible de retrouver l'état initial du système. Cette propriété est traduite par l'équation suivante :

$$rang(O) = rang({}^{t}[C\ CA \dots CA^{n-1}]) = n \tag{9}$$

où n est la dimension du vecteur d'état.

Exemple : moteur à courant continu commandé par l'inducteur

En reprenant l'équation différentielle de l'exemple précédent et en modifiant son écriture:

$$\frac{d^2\omega}{dt^2} = b_0 u - a_1 \frac{d\omega}{dt} - a_0 \omega \tag{10}$$

On a alors les variables d'état suivantes :

$$\begin{cases} x_1 = \omega \\ x_2 = \frac{\omega}{dt} \\ \dot{x}_1 = x_2 \\ \dot{x}_2 = b_0 u - a_0 x_1 - a_1 x_2 \end{cases} \quad (11)$$

Il est ainsi possible d'écrire les équations matricielles suivantes faisant apparaître les différentes matrices canoniques de la représentation d'état:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} u \\ \omega(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{cases} \quad (12)$$

Le système est alors commandable car le rang de la matrice de commandabilité est de 2 mais n'est pas observable car le rang de la matrice d'observabilité est de 1. Pour obtenir un système observable, il faut de la matrice d'observabilité soit d'une forme identique à la matrice identité de dimension 2.

2.2.5 VHDL-AMS

Le VHDL signifie VHSIC Hardware Description Language et VHSIC Very High Scale Integrated Circuit [77]. L'extension AMS correspond à Analog & Mixed Signals. Ce langage est normalisé par l'IEEE suivant les normes 1076,x . A l'origine ce langage était destiné à la conception et la simulation de circuits intégrés à haute vitesse pour le département de la défense américain dans les années 80 qui fut alors complété au fil du temps, la dernière normalisation datant de 1999. Ce langage est donc naturellement adapté à la description de circuits électroniques et électriques et possède l'avantage d'être directement vérifiable par des simulateurs. De par sa normalisation, le VHDL est devenu l'un des langages de programmation des composants de type FPGA et ce quel que soit le fabricant du composant. Il existe donc un fort potentiel pour les applications temps-réels et/ou de simulations accélérées. Ce langage est relativement facile à mettre en œuvre par la possibilité d'utiliser des bibliothèques normalisées, de créer ses propres bibliothèques mais aussi de pouvoir réaliser une conception modulaire grâce à une complémentarité entre interface graphique et code descriptif du composant. Comme cité précédemment une bibliothèque de composant électrique est fournie sous forme de composants délimités par des ports échangeant des flux (tension et intensité), il est alors possible de créer des bibliothèques pour les domaines de la mécanique, thermique, fluidique ….

Exemples :

Exemple d'une résistance:

```
library IEEE;
use work.electrical_system.all;
use IEEE.math_real.all;
```

```
entity resistor is
generic (resistance : real);
port (terminal n1, n2: electrical);
end resistor;

architecture core of resistor is
quantity resistor_u across n1 to n2;
quantity resistor_i through n1 to n2;
begin
resistor_u == resistance * resistance_i
end architecture core;
```

Exemple d'un amortisseur :

```
library IEEE;
use IEEE.MATH_REAL.all;
library IEEE_proposed;
use IEEE_proposed.MECHANICAL_SYSTEMS.all;

entity damper_t is
generic (d : real); -- Damping coefficient [N/(m/s)]
port ( terminal trans1, trans2 : translational);
end entity damper_t;

architecture ideal of damper_t is
quantity velocity : velocity;
quantity position across force through trans1 to trans2;

begin
velocity == position'dot;
force == d*velocity;
end architecture ideal;
```

2.2.6 Verilog-AMS

Le Verilog [76] est un autre langage de programmation des FPGA, qui à l'origine n'était pas normalisé car propriétaire et développé par Cadence Design System, qui a pour but de ressembler au C . Cependant ce langage a été ouvert afin d'obtenir une normalisation par l'IEEE (1364-1995), la dernière révision datant de 2005. Le Verilog-AMS est un dérivé du Verilog pour la prise en charge des Analog & Mixed Signals pour la modélisation de signaux analogiques (continus) et mixtes. Ce langage possède le même esprit que le VHDL par l'usage de

bibliothèques de composants pouvant échanger des flux d'énergie.

Exemples :

Exemple d'une résistance :

```
module res(p,n);
inout p,n;
electrical p,n;
parameter real r=0 from [0:inf);
analog V(p,n) <+ r*I(p,n);
endmodule
```

2.2.7 Bond Graphs

Le Bond graph a été développé en 1959 au MIT par Paynter [53]. Cette modélisation de type graphique a été conçue afin de pouvoir simuler des systèmes multi-physiques en utilisant une approche énergétique aussi bien pour les cas de systèmes linéaires que non-linéaires. Cette approche permet de découper un système en sous-système s'échangeant des flux d'énergie de puissance et génère une structuration modulaire du modèle. Le Bond graph est un langage causal dont il est possible de déterminer sa caractéristique : intégrale / dérivée ou résistance / conductance.

Exemple : Moteur à courant continu commandé par l'inducteur

Ce cas de figure a été traité par Jardin dans sa thèse [31], les figures présentées sont donc extraites de celle-ci. Le schéma 10 rappelle la problématique du moteur à courant continu et le schéma 9 est une représentation par les Bond Graphs du problème.

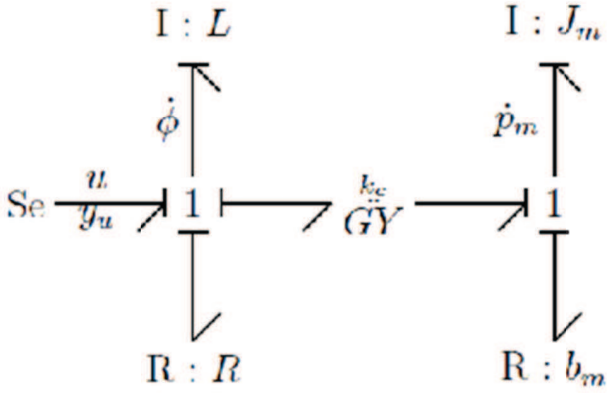

Figure 9 : Bond Graphs du moteur à courant continu

2.2.8 Modelica

Le langage Modelica est issu du travail de Hilding Elmqvist [18] et dont la première version fut présentée en 1997. En 2000, l'association Modelica est fondée afin de continuer à enrichir ce langage par l'amélioration de la bibliothèque libre Modelica Standard Library. La dernière version de Modelica est la 3.2 sortie en mars 2010 [82]. Modelica est un langage orienté objet c'est à dire qu'il est naturellement modulaire et permet donc un développement du système global vers le composant de façon souple. Ce langage de modélisation est acausal par défaut (la causalité peut être forcée si besoin), supporte une modélisation hybride (support de modèles mélangeant parties continues et discrètes) et gère l'aspect multiphysique d'un modèle. Il supporte aussi les modèles de type schéma bloc, fonctions de transfert ainsi que des fonctions codées en C, Fortan et Java. Un inconvénient de ce langage est la non prise en charge des équations aux dérivées partielles mais ce problème peut être réglé par l'interfaçage avec Modelica d'un logiciel capable de résoudre de telles équations.

Exemple : Moteur à courant continu commandé par l'inducteur

Modelica a la particularité de combiner aussi bien représentation graphique que code directement accessible à l'utilisateur. La figure 10 permet d'avoir une représentation graphique.

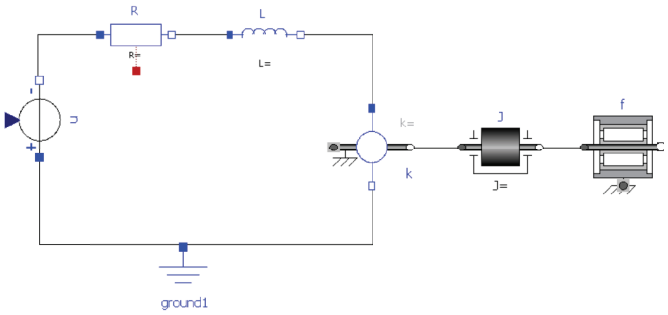

Figure 10 : Schéma du moteur à courant continu

Le code ci-dessous représente l'inductance de la figure 10 d'une façon textuelle. Il est alors possible à l'utilisateur de modifier l'équation de l'inductance afin de prendre, par exemple, en compte une éventuelle perturbation électromagnétique.

Exemple d'une inductance :

```
model Inductor "Ideal linear electrical inductor"
extends Interfaces.OnePort;
parameter SI.Inductance L(start = 1) "Inductance";

equation
  L * der(i) = v;
end Inductor;
```

2.2.9 Autres langages

On peut citer d'autres langages possibles pour la modélisation de systèmes :

• ADA [40], C [34], C++ [78], Fortran [3], OpenCl[26] …. qui sont des langages de programmation numériques ;

• le calcul formel, qui permet directement la manipulation des équations sans requérir à des évaluations. Ce type de calcul est puissant mais requière un temps de calcul non négligeable;

• L'utilisation des éléments finis dans le cadre d'un calcul multi-physique qui permet d'obtenir des résultats à des échelles relativement petites mais dont la taille de modèle et le coût de calcul peut devenir très rapidement très importants.

2.3 Outils de modélisation

2.3.1 Matlab et Simulink

Matlab et Simulink [80] sont deux environnements complémentaires développés par The Mathworks. Cette combinaison apparaît comme l'un des standards de la modélisation. Matlab se présente sous forme d'un interpréteur capable d'exécuter des calculs scientifiques sous forme de texte (ou ligne de commandes) avec la possibilité d'utiliser de nombreuses fonctions pré-programmées. Simulink est une interface graphique de simulation conçue pour être couplée avec les scripts écrits sous Matlab. Un modèle développé sous Simulink se présente sous forme de boites, reprenant les fonctions Matlab ou les scripts écrits en Matlab, dotées d'entrées / sorties fixes. Ces boites sont de type causal et le modèle nécessite alors la création de boucles de rétroactions explicites. Il est cependant à citer, par exemple, la bibliothèque de fonction «SimPower Systems» qui permet de rajouter une surcouche acausale à Simulink. Les modèles peuvent mixer aussi bien des fonctions continues, discrètes ou même d'état. Il est également possible pour l'utilisateur d'importer sous forme de bloc des fonctions codées en C, C++, Fortran ou encore ADA. Les modèles sont retraduits en langage S pour pouvoir être simulés par les différents solveurs présents. Il est également possible d'exporter le code ainsi généré vers des systèmes temps-réel aussi bien de type ordinateur (Opal-RT) que vers des cartes dédiées (dSPACE) ou encore vers des FPGA via une traduction en VHDL (Altera, …) . Un aspect critiquable de Matlab / Simulink est l'utilisation de fonctions pré-codées dont il n'est pas possible d'analyser les méthodes de calcul (accès aux sources impossible).

2.3.2 Scilab et Xcos

Le couple Scilab et Xcos [93] est développé par le consortium Digiteo constitué notamment par le CEA, le CNRS, l'INRIA … Ce logiciel est en fait un environnement de modélisation comparable à Matlab / Simulink mais qui possède l'avantage d'être open source (le code source est disponible pour l'utilisateur). A l'instar de Matlab, Scilab possède des boites à outils dédiées à certains usages (Traitement de signal, …) mais qui sont créées par des contributeurs au programme (ex : Particle Swarm Optimization Toolbox). Xcos, l'équivalent de Simulink, est constitué d'éléments semblables mais complété par la prise en charge de bloc acausaux grâce à l'implantation directe du langage Modelica.

2.3.3 Quartus, ISE, …

Ces logiciels, respectivement dédiés aux FPGA des marques Altera [92] et Xilink [97], fournissent un environnement au développement de systèmes adaptés à l'usage du VHDL mais aussi du Verilog. Tous deux possèdent une interface graphique de représentation des composants et de leurs liaisons, d'un outil de simulation temporel de ceux-ci ainsi que des

architectures systèmes, et d'un outil de compilation pour la création d'un code exécutable sur un FPGA. Il existe des environnements libres de développement et de simulation de code comme par exemple GHDL [22]. Cependant dès qu'il est nécessaire d'effectuer le transfert du programme vers du matériel il est requis de passer par le logiciel du fabricant.

2.3.4 Simplorer

Ce logiciel de modélisation, simulation et analyse est développé par Ansoft [95], plus connu pour la création du logiciel de calcul par éléments finis Ansys. La modélisation utilisée est de type 1D multi-physique et met en œuvre une modélisation par composants en utilisant des modèles codés en C/C++, VHDL ou langage propriétaire. De plus il est possible d'insérer des diagrammes blocs, des modèles sous SPICE ou encore de pouvoir être couplé avec des éléments finis.

2.3.5 SimulationX, AMEsim, Dymola

SimulationX, AMEsim, Dymola sont des logiciels [96, 73, 75] à destination des industriels utilisant principalement une modélisation à partir de Modelica. Ils se composent d'une interface graphique, de bibliothèques de composants, de possibilité de créer des composants à partir d'un code source et possèdent un simulateur. Ces logiciels sont interfaçables avec d'autres logiciels tels que Simulink. De plus ils possèdent tous trois la capacité de générer du code pour des systèmes temps-réel soit en passant par Simulink soit directement en C. Ils ont aussi la possibilité de pouvoir effectuer des analyses de robustesse (ex : Monte-Carlo) ainsi que des optimisations (ex : plan d'expérience, algorithmes génétiques …) suivant chaque logiciel. Il est cependant possible de programmer des scripts afin de combler un éventuel manque.

2.3.6 Jmodelica, OpenModelica

Ces deux logiciels libres Jmodelica et OpenModelica [79, 89], qui sont plutôt destinés à la recherche, utilisent Modelica comme langage de programmation. A l'instar des logiciels précédents, on retrouve les interfaces graphiques, les bibliothèques de composants, la possibilité de créer des composants ainsi qu'un simulateur. Ces logiciels étant à sources disponibles, il est possible de les interfacer de façon très puissante avec d'autres logiciels. Cependant il n'existe pas d'outils permettant de générer du code directement implantable pour des systèmes temps-réel mais la structure de compilation de Modelica permet de générer du code C++.

2.3.7 20Sim

Ce logiciel [94] se présente comme ceux décrits précédemment c'est à dire avec une interface graphique comme forme prioritaire de création de modèle. Deux formes de langage sont utilisées par ce logiciel : le SIDOPS et les Bonds Graphs. Le SIDOPS est non normalisé mais se présente de la même façon qu'un composant réalisé sous Modelica et donc il peut être tout à fait possible de réaliser le passage de l'un des langages à l'autre sans trop de problème. L'aspect intéressant de ce logiciel est l'utilisation des Bond Graphs.

2.4 Matériel de simulation

Cette partie a pour but de faire un point sur les différentes plateformes couramment disponibles afin de réaliser des simulations de systèmes. D'autres systèmes sont évidemment disponibles mais dans ce cas plutôt dédiés à des applications spécifiques.

2.4.1 Processeurs classiques et GPU

L'évolution des processeurs a été très rapide durant les vingt dernières années passant, par exemple, du Pentium I à l'architecture Core ix chez Intel. Jusqu'à quelques années et l'apparition des premiers Athlon x2, les fabricants ont augmenté les fréquences de fonctionnement de l'ordre de 100 MHz à 3.5 GHz. A cause de limitations physiques, les fondeurs s'orientent maintenant vers une multiplication des processeurs afin d'améliorer les capacités de calcul.

En 2007, Nvidia permet le calcul sur carte graphique permettant la programmation de celle-ci par l'intermédiaire d'un langage propriétaire (CUDA). En 2008, le groupe Khronos, regroupant des acteurs majeur du monde informatique, fait la démonstration d'OpenCL qui permet l'exécution de code sur toute carte graphique récente mais aussi sur processeur classique. L'intérêt de l'utilisation de ce matériel provient du fait que les cartes graphiques sont composées d'un nombre important d'unités de calculs en parallèle et donc très performant pour du calcul matriciel.

Pour comparaison un des derniers processeurs d'Intel (i7 980x) atteint 0.1 Tflops en double précision alors que les cartes graphiques de génération GTX 4xx atteignent 0.5 Tflops en double précision et 1 Tflops en simple précision. Le processeur coute actuellement de l'ordre de 800 euros contre 250 pour la carte graphique.

Les cartes graphiques sont donc très intéressantes pour les applications à calcul matriciel dont les éléments finis font partis.

Matlab offre la capacité de calcul sur carte graphique via l'utilisation d'une toolbox dédiée.

2.4.2 Cluster de calcul

Le cluster de calcul, ou grille de calcul, est la mise en réseau et en parallèle d'un nombre important d'unité de calcul afin de traiter d'important problèmes scientifiques tels que la prévision météorologique. Cette méthode de calcul est en évolution permanente grâce à l'évolution rapide du matériel informatique. Les plus puissants clusters ont des performances impressionnantes mais ne sont que peu accessibles car réservés à des projets scientifiques nécessitant d'important calculs sans commune mesure avec ce qui est abordé dans cette thèse, ou encore pour des programmes militaires. Une alternative pour les chercheurs peut être l'usage du projet Boinc permettant la création d'un cluster sur une base de participation volontaire.

2.4.3 xPC Target, OpalRT, RTOS

L'un des inconvénients des ordinateurs grand public est la quantité de ressource nécessaire au système d'exploitation pour fonctionner. Ce qui est préjudiciable pour le calcul scientifique. Les projets XPC Target [80], OpalRT [85] ou encore les RTOS (Real Time Operating Systems) sont des environnements minimalistes permettant de libérer au maximum les ressources informatiques pour l'exécution de calculs. Ce genre de plateforme permet alors de réaliser des calculs en temps réel et de réaliser des interfaces avec l'extérieur par l'usage de cartes d'entrées sorties dédiées. Matlab permet l'usage de tels systèmes.

2.4.4 dSpace

dSpace [84] fabrique des systèmes (cartes pour ordinateur ou boitiers externes) programmables et uniquement dédiés à la réalisation de calculs. Ce type de matériel est donc adapté à la réalisation de calculs en temps réels et possède des interfaces d'entrées sorties pour pouvoir interagir avec l'extérieur.

2.4.5 FPGA

Les FPGA sont des processeurs d'un type particulier car ils permettent une programmation matérielle d'un programme tout en étant reconfigurables. Ceux-ci n'atteignent pas des fréquences aussi hautes que les processeurs classiques (de l'ordre de la centaine de MHz) mais ce défaut est compensé par leur structure même. En effet les FPGA sont dotés d'une matrice d'interconnexion des éléments présents qui permet d'obtenir une programmation en parallèle des instructions. Ils sont utilisés pour le prototypage rapide de puces électroniques, la commande et le contrôle de systèmes mécatroniques. Ils sont aussi utilisés pour la gestion des entrées sorties dans les systèmes OpalRT ou encore dSpace. Dans le cadre scientifique, il est fait état de publications [25, 41, 55] utilisant de tels processeurs pour la simulation en temps-réel en embarquant des réseaux de neurones.

2.5 Conclusion

Dans un premier temps, il convient donc de choisir un niveau de simulation du système à étudier afin de déterminer les différentes contraintes liées aux différents niveaux et domaines de simulation. Puis doit s'effectuer un choix des techniques qui seront présentes dans le modèle final, ces techniques pouvant alors être de natures différentes suivant la facilité ou coutume de modélisation des composants. Ces différents points défini, le choix du langage de modélisation et de l'outil associé est alors primordial compte-tenu d'éventuelles contraintes telles que les interactions ou la clarté de modélisation finale. Le choix du matériel de simulation se faisant alors suivant les besoins spécifiques de l'utilisateur.

3 Démarche de caractérisation et d'identification des paramètres physiques d'un système

Sommaire

Les chapitres précédents ont défini le terme mécatronique, les différentes méthodes de conception associée, ainsi que les outils pouvant être utilisés pour leur modélisation. Cette partie a pour but de mettre en place une démarche standard dans l'établissement d'un modèle de type mécatronique.

La démarche proposée se compose de trois parties principales : l'observation du système et la création de modèles susceptibles de représenter celui-ci, la création de banc de test et de son instrumentation et enfin la sélection des modèles performants par l'identification des paramètres.

Pour cela, nous prendrons comme exemple d'application une électrovanne intelligente qui aura pour but de maintenir un niveau constant de fluide dans un réservoir, à partir de la connaissance du débit du fluide s'écoulant par la vanne, mais dont le débit de remplissage du réservoir sera variable et potentiellement inconnu dans le temps.

3.1 Observation et création d'un modèle

L'observation est la première étape en vue de la création d'un modèle mécatronique. Cette phase permet de différentier les différents constituants et de réaliser une recherche sur des modèles déjà existants. Un modèle de l'ensemble du système est alors créé par assemblage des modèles des différents constituants.

3.1.1 Structure du système

Comme tout système mécatronique, le système suit la composition de la Figure 11 où nous retrouvons les éléments suivants :

- une interface homme / machine qui doit permettre de suivre l'état du système;
- un système de traitement des données qui a pour charge la gestion totale des informations circulant dans le système;
- un modulateur de puissance qui est le fournisseur d'énergie à l'actionneur;
- une source d'énergie;
- un actionneur pour mettre en mouvement la charge;
- la charge qui a une influence sur le système;
- enfin, les détecteurs qui fournissent les informations à la partie intelligente.

Dans cette application, le système met en œuvre les domaines de la fluidique, de l'électromécanique, ainsi que du traitement de l'information et du signal.

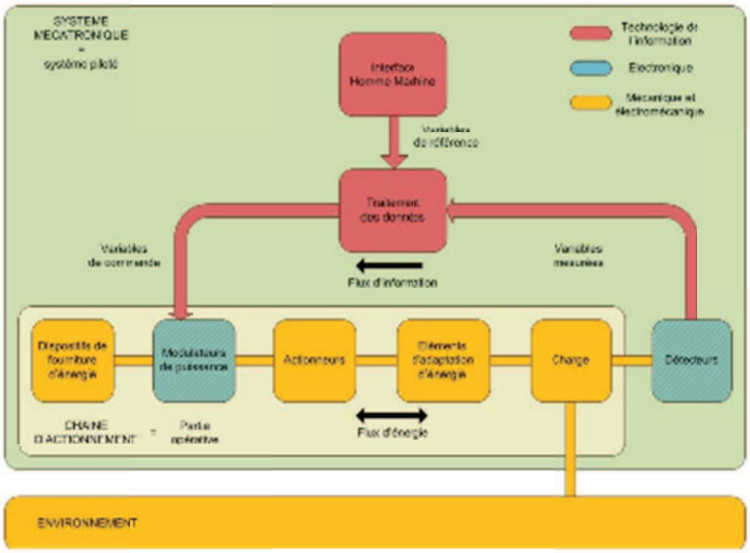

Figure 11 : Architecture générale d'un système mécatronique

3.1.2 Composition du système

Le système se présente donc sous forme de sous-systèmes interconnectés entre eux. Nous allons maintenant détailler ces sous-systèmes de façon analytique afin de construire le modèle global.

3.1.2.1 Interface homme / machine

L'Interface Homme / Machine (IHM) permet de suivre l'état du système par la présence de plusieurs indicateurs. L'indicateur le plus important est la connaissance de la hauteur de fluide contenu dans le réservoir contrôlé. Des indicateurs auxiliaires, tels que position de la valve ainsi que consommation électrique de l'ensemble du système, pourront aussi être ajoutés. Ces indicateurs se présentent sous forme d'une valeur numérique actualisée en temps réel.

3.1.2.2 Le système fluidique

Le système fluidique contrôlé se compose d'un réservoir ouvert, de deux tuyaux, d'une valve ainsi que d'un fluide. Dans tout le reste de cette description, l'écoulement de ce fluide, caractérisé par sa viscosité dynamique ainsi que par sa densité, est considéré comme laminaire. C'est à dire que le fluide s'écoule sans présence de turbulences, ni de rotations de celui-ci. Les équations fluidiques présentées ci-dessous sont issues de la bibliothèque fluidique QSSFluidFlow

pour Modelica [19]. Le fluide est considéré comme étant incompressible et homogène, ce qui permet de se placer dans le cas d'un fluide newtonnien : le tenseur des contraintes visqueuses est une fonction linéaire du tenseur des déformations. Pour ce genre de fluide, les équations reliant forces et accélérations sont celles de Navier-Stokes. Dans l'hypothèse d'un fluide incompressible et à masse volumique constante, elles s'écrivent :

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla(\rho \vec{v} \otimes \vec{v}) = -\nabla p + \nabla \bar{\bar{\tau}} + \rho \vec{f} \quad (13)$$

Nom de la variable	Symbole	Unité
Masse volumique	ρ	kg/m^3
Viscosité dynamique	μ	$Pa.s$
Vitesse du fluide	v	m/s
Pression du fluide	p	Pa
Tenseur des contraintes visqueuses	τ	Pa

Table 2 : Variables du fluide

Le réservoir est considéré comme cylindrique, indéformable (son volume ne peut varier en fonction de la pression du fluide). Le réservoir se remplit à pression ambiante suivant un débit de remplissage. La sortie du réservoir se situe en bas de celui-ci pour être connectée à un tuyau. Le fonctionnement du réservoir peut-être décrit par les équations suivantes :

Nom de la variable	Symbole	Unité
Rayon du réservoir	r	m
Surface du réservoir	S	m^2
Hauteur de fluide	h	m
Pression de sortie	P_s	Pa
Perte de charge de sortie	P_{lost}	Pa
Coefficient de perte de charge	K	$Pa.m.s^2/Kg$
Vitesse de sortie du fluide	v_s	m/s
Débits d'interfaces du fluide	q_i	m^3/s

Table 3 : Variables du réservoir

$$\begin{cases} S = \pi . r^2 \\ P_s = \rho . g . h + P_{atm} - P_{lost} \\ P_{lost} = K . \frac{\rho}{2} . v_s^2 \\ \frac{dh}{dt} = \frac{1}{S} . \sum q_i \end{cases} \quad (14)$$

Les différents tuyaux sont considérés comme cylindriques et indéformables (ils ne peuvent stocker de fluide par déformation sous l'effet de la pression). Lors du passage d'un fluide dans un tuyau, celui-ci est affecté par une perte de charge qui est fonction de la vitesse du fluide, de la géométrie du tuyau ainsi que par un coefficient de frottement. Le fonctionnement d'un tuyau est défini par :

Nom de la variable	Symbole	Unité
Longueur	L	m
Diamètre	S	m^2
Perte de charge	Δp	Pa
Vitesse du fluide	v	m/s
Viscosité cinématique	ν	m^2/s
Coefficient de Reynolds	R_e	
Coefficient de perte de charge	λ	

Table 4 : Variables du tuyau

$$\begin{cases} \Delta p = \lambda . \frac{L}{D} . \frac{\rho}{2} . v^2 \\ \lambda = \frac{64}{R_e} \\ R_e = \frac{|v| D}{\nu} \end{cases} \quad (15)$$

La valve est un élément d'interaction. C'est à la fois un composant mécanique qui peut être caractérisé par une masse, un amortisseur, ... mais qui agit aussi sur l'écoulement du fluide. La valve considérée est une vanne linéaire à éclipse : la perte de charge introduite par celle-ci est directement liée au taux d'ouverture et son mécanisme se fait par superposition de cercles.

Nom de la variable	Symbole	Unité
Rayon de valve	R	m
Distance de déplacement	d	m
Aire unitaire d'ouverture	A	
Débit du fluide	q	m^3/s
Constante de passage de fluide max.	K_{vs}	
Constante de passage de fluide	K_v	
Différence de pression aux bornes	Δp	Pa
Masse de la valve	m	kg
Raideur de la valve	k	N/m
Amortissement de la valve	q	$N.s/m$
Loi de frottement de la valve	$F_{friction}$	N
Force de l'actionneur	F	N

Table 5 : Variables de la valve

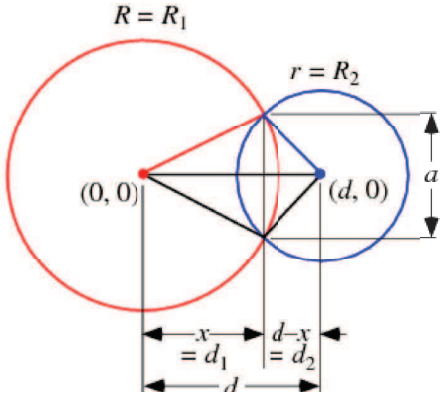

Figure 12 : Schéma d'une intersection de cercles, Wolfram.com

L'aire unitaire d'ouverture est donnée par la relation suivante, en posant l'hypothèse de rayons identiques :

$$\begin{cases} A_{inter} = 2R^2\cos^{-1}\left(\frac{d}{2R}\right) - \frac{1}{2}d(4R^2 - d^2)^{\frac{1}{2}} \\ A_{max} = \pi R^2 \\ A = \frac{A_{max} - A_{inter}}{A_{max}} \end{cases} \quad (16)$$

L'équation fluidique de la valve peut alors s'écrire comme :

$$\begin{cases} K_v = K_{vs}.A \\ \Delta p = \rho.\left(\frac{q}{K_v}\right)^2 \end{cases} \quad (17)$$

En considérant que la valve est un élément mécanique potentiellement non-linéaire par l'introduction de pertes par frottement, celle-ci est représentée par un système de masse en frottement en série avec un système ressort-amortisseur :

$$m.\ddot{d} = F - k.d - q.\dot{d} - F_{friction} \quad (18)$$

3.1.2.3 Actionneur

L'actionneur utilisé est un actionneur de type slip & stick inertiel. Ce type d'actionneur, de conception relativement simple, permet de réaliser des déplacements importants en effectuant de petits pas par le principe du slip & stick ou bien glissé - collé. L'aspect inertiel de l'actionneur est réalisé par la présence d'une masse importante comparée aux autres masses mises en mouvement. Ce principe consiste à mettre en mouvement un ensemble en frottement, le glissé, par l'apport d'une énergie, puis celle-ci diminuant par frottement devient insuffisante pour assurer le mouvement de l'ensemble qui se retrouve alors collé. Cette opération est alors répétée afin d'obtenir un déplacement conséquent.

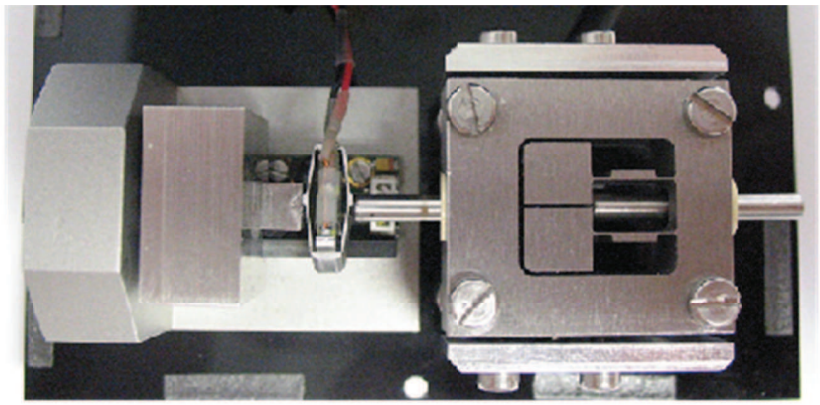

Figure 13 : Photo d'un actionneur slip & stick inertiel, SPA35XS de Cedrat

Ces actionneurs, et notamment les micro-actionneurs, de ce type ont été étudiés par Breguet [7]. L'auteur y présente une étude des différents paramètres intervenant dans ces actionneurs ainsi que la réalisation de prototypes. Les déplacements obtenus par ces réalisations sont de l'ordre du nm avec des vitesses de l'ordre du mm/s sur des longueurs importantes et donc compatibles avec des applications de micro-mécanique.

Compte-tenu de l'aspect inertiel de l'actionneur utilisé, l'énergie mécanique nécessaire à la mise en mouvement est fournie par un composant faisant partie de l'équipage mobile. Ce composant est un élément piézoélectrique qui devient alors un élément d'interface entre l'actionneur et le pré-actionneur. Du fait du déplacement linéaire de la valve, l'actionneur sera aussi de type linéaire.

L'actionneur utilisé se présente de la manière suivante : une masse d'inertie est mise en mouvement par un composant piézoélectrique dont la déformation est amplifiée par une coque déformable qui est aussi reliée à un arbre en liaison glissière avec le bâti. La glissière assure le frottement nécessaire au fonctionnement de l'actionneur. L'arbre est alors relié à la valve autorisant sa mise en mouvement.

Modèle électromécanique

Dans [60], Rodriguez présente l'étude d'un élément piézoélectrique amplifié mettant en mouvement une masse. Cet élément amplifié (APA120ML) est issu d'une même conception que celui étudié ici, seule l'échelle de l'étude diffère car l'actionneur étudié (APA35xs) est plus petit.

Nom de la variable	Symbole	Unité
Déplacement électrique	D	C/m^2
Contraintes	T	N/m^2
Champ électrique	E	V/m
Elongation	S	
Raideur mécanique avec déplacement électrique constant	C^D	N/m^2
Constante piézoélectrique	h	V/m
Résistance à contrainte mécanique constante	β^S	Vm/C

Table 6 : Variables de l'actionneur

En considérant que le matériau est isotrope, que le champ électrique est perpendiculaire à la surface du matériaux dans la direction de polarisation et que le comportement est associé à un condensateur, les équations suivantes peuvent êtres écrites:

$$\begin{cases} E = \nabla V \approx \frac{V}{l} \\ Q = \int_A D\delta\, \partial A \approx DA \end{cases} \quad (19)$$

Par ce biais, il est possible d'établir une relation entre déplacement, charge électrique, force et tension par le système suivant :

$$\begin{bmatrix} F \\ V \end{bmatrix} = \begin{bmatrix} C^D \frac{A}{l} & h \\ -h & \beta^S \frac{l}{A} \end{bmatrix} \cdot \begin{bmatrix} x \\ Q \end{bmatrix} \tag{20}$$

Tel quel, ce système d'équation s'avère peu utilisable. Rodrigez propose alors deux modèles électromécaniques du composant étudié. Dans les deux cas, la partie mécanique est représentée par un ressort et un générateur soit de force, soit de déplacement, la partie électrique étant représentée par un condensateur monté en parallèle ou en série avec un condensateur variable. Des constantes de liaisons sont utilisées pour faire le lien entre partie mécanique et partie électrique.

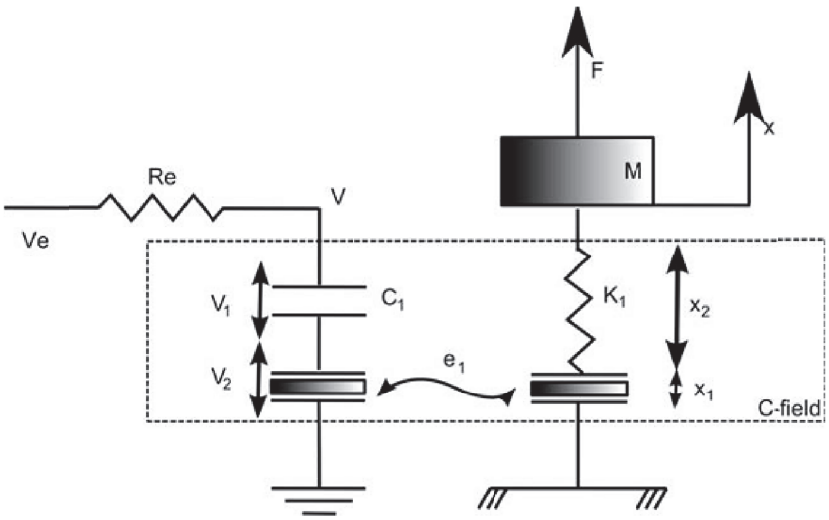

Figure 14 : Schéma de l'actionneur piézoélectrique pour le modèle 1

Le premier modèle est issu du calcul de l'énergie totale du système :

$$U = \frac{1}{2} K_1 (x - a_1 Q)^2 + \frac{1}{2} \frac{Q^2}{C_1} \tag{21}$$

En posant les dérivées partielles suivantes,

$$\begin{cases} F = \frac{\partial U}{\partial x} \\ V = \frac{\partial U}{\partial Q} \end{cases} \tag{22}$$

il est possible d'obtenir l'écriture matricielle suivante du système :

$$\begin{bmatrix} F \\ V \end{bmatrix} = \begin{bmatrix} K_1 & -K_1 a_1 \\ -K_1 a_1 & K_1 a_1^2 + \frac{1}{C_1} \end{bmatrix} \cdot \begin{bmatrix} x \\ Q \end{bmatrix} \tag{23}$$

Nom de la variable	Valeur	Condition
a_1	x/Q	$F = 0$
C_1	Q/V	$F = 0$
K_1	$F/(a_1 Q)$	$x = 0$

Table 7 : Variables pour le modèle 1

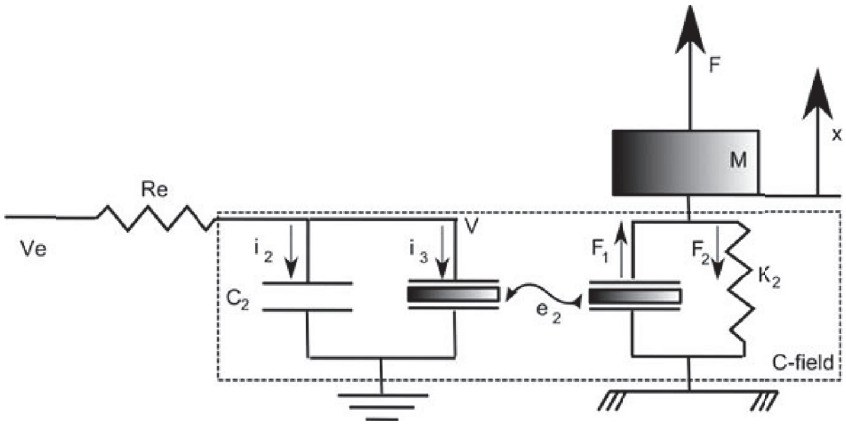

Figure 15 : Schéma de l'actionneur piézoélectrique pour le modèle 2

Dans le second modèle, l'énergie contenue dans le système s'écrit de manière différente :

$$U = \frac{1}{2}K_2 x^2 + \frac{1}{2C_2}\left(\frac{x}{a_2^2} - Q\right) \tag{24}$$

En appliquant de nouveau le calcul des dérivées partielles, le système d'équation s'écrit :

$$\begin{bmatrix} F \\ V \end{bmatrix} = \begin{bmatrix} K_2 + \frac{1}{a_2^2 C_2} & -\frac{1}{a_2 C_2} \\ -\frac{1}{a_2 C_2} & \frac{1}{C_2} \end{bmatrix} \cdot \begin{bmatrix} x \\ Q \end{bmatrix} \tag{25}$$

Nom de la variable	Valeur	Condition
a_2	V/F	$x = 0$
C_2	Q/V	$x = 0$
K_2	F/x	$V = 0$

Table 8 : Variables pour le modèle 2

Ces deux approches permettent d'obtenir un modèle linéaire d'un élément piézoélectrique. Le choix d'un modèle peut se faire suivant les données ou montages d'expériences disponibles. Cependant les composants piézoélectriques sont soumis à des phénomènes d'hystérésis ainsi que de dépendance face aux vitesses de variation des paramètres. Ce sont donc des phénomènes fortement non-linéaires.

Opérateurs d'hystérésis

Plusieurs opérateurs sont disponibles afin de représenter les phénomènes d'hystérésis intervenant dans les matériaux piézoélectriques.

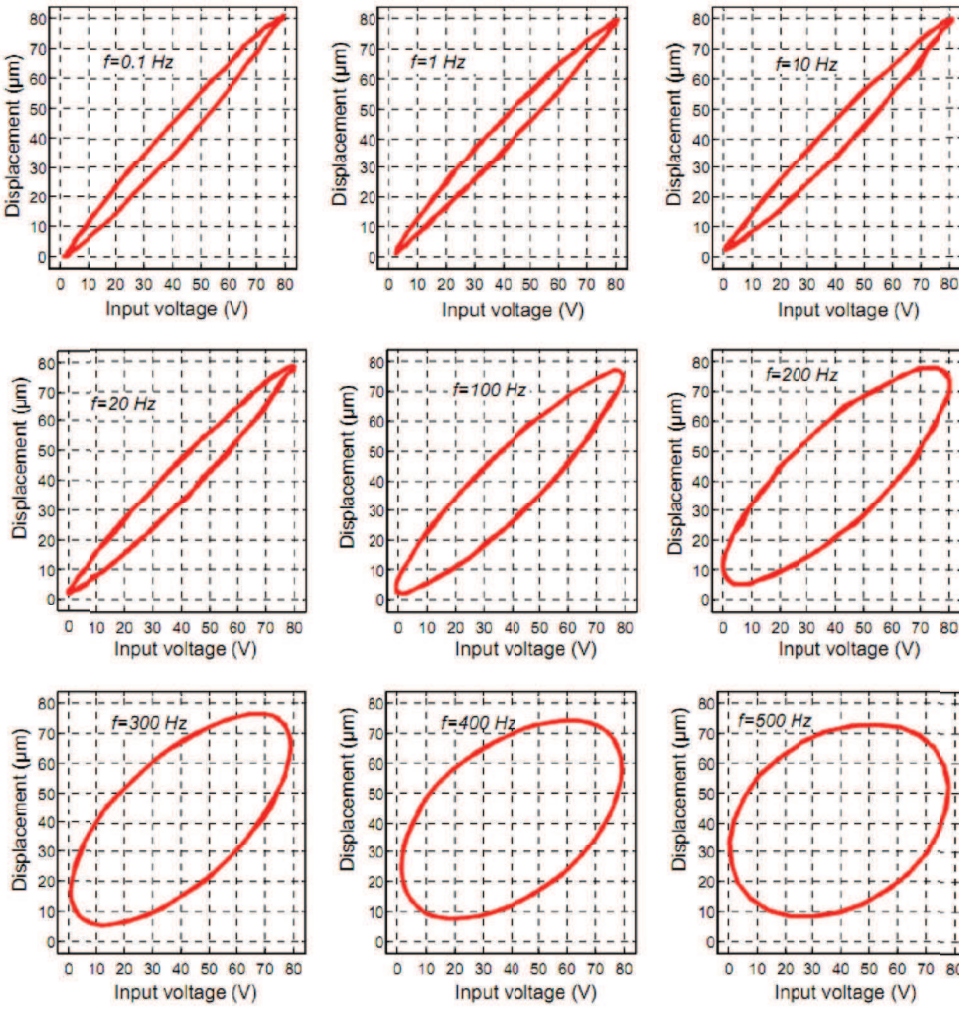

Figure 16 : Exemples d'hystérésis, extrait de [30]

On peut dégager trois modèles cités dans diverses publications : l'opérateur de Bouc-Wen [6, 81] constitué d'une unique équation non linéaire, et les modèles de type Preisach [56], ou bien Ishlinkii [30], composés d'une addition d'opérateurs non-linéaires élémentaires.

L'opérateur de Bouc-Wen peut s'écrire de façon générale : soit $Y(t)$ le phénomène soumis à hystérésis avec $u(t)$ l'excitation. On pose $z(t)$ une variable d'état et α, β, γ et n des paramètres permettant de contrôler l'amplitude et la forme de la courbe d'hystérésis. L'équation s'écrit alors :

$$\begin{cases} Y = f(u) + f(z) \\ \dot{z} = \alpha\dot{u} - \beta|\dot{u}|z|z|^{n-1} - \gamma\dot{x}|z|^{n} \end{cases} \tag{26}$$

Rodrigez propose de se baser sur la densité de polarisation P pour insérer l'opérateur :

$$\begin{cases} P = \frac{Q}{A} \\ E = K_x P + K_w z \\ \dot{z} = \rho\left(\dot{P} - \sigma|\dot{P}||\dot{z}|^{n-1} z + (\sigma - 1)\dot{P}|z|^n\right) \end{cases} \quad (27)$$

Dans [27], l'opérateur est lui inséré directement sur la tension appliqué à l'élément :

$$\begin{cases} V_{out} = V_{in} - V_h \\ \dot{V}_h = \alpha \dot{V}_i n - \beta|\dot{V}_{in}|\dot{V}_{in}|V_h - \gamma \dot{V}_{in}|V_h| \end{cases} \quad (28)$$

On remarque ici que l'auteur a simplifié l'équation en posant $n = 1$.

Afin de prendre en compte la dépendance de l'opérateur par rapport aux vitesses de variation, [71] propose d'ajouter un terme calculé à partir d'une puissance de la vitesse de variation. L'équation de Bouc-Wen devient alors, avec δ la fonction dirac :

$$\begin{cases} z = z_{BW} + z_{MD} \\ \dot{z}_{BW} = \alpha \dot{u} - \beta|\dot{u}|z|z_{BW}|^{n-1} - \gamma \dot{x}|z|^n \\ \dot{z}_{MD} = c\ddot{u}[m|\dot{x}|^{m-1} + |\dot{x}|^m \delta(\dot{x})] \end{cases} \quad (29)$$

Un modèle d'hystérésis non basé sur une équation est couramment utilisé : le modèle de Preisach. Celui-ci met en oeuvre une somme de comparateurs à hystérésis afin de d'écrire la courbe expérimentale. L'opérateur s'écrit ainsi :

$$\begin{cases} Y(t) = \sum_{j=1}^{N} \sum_{i=1}^{N} \mu(\alpha_i, \beta_j) \gamma_{\alpha_i, \beta_j}[x(t)] \\ \alpha_i = \beta_i = \alpha_1 - 2\frac{i-1}{N-1}\alpha_1 \\ \gamma_{\alpha\beta}[x(t)] = \begin{cases} -1,\ u(t) \le \beta \\ 1,\ u(t) \ge \alpha \\ statu\ quo,\ \alpha \le u(t) \ge \beta \end{cases} \end{cases} \quad (30)$$

Dans [30], est présenté un opérateur d'hystérésis basé sur une addition de simples opérateurs non-linéaires : le modèle d'Ishlinkii. Dans ce cas l'équation d'hystérésis s'écrit :

$$Y(t) = \sum_{i=1}^{n} w_i H_i(x(t)) \quad (31)$$

avec H_i l'opérateur backlash défini par, avec r_i et w_i des paramètres:

$$H_i(x(t)) = max[x(t) - r_i,\ min(x(t) + r_i,\ H_i(x(t-1)))] \quad (32)$$

Afin de rendre l'opérateur non-symétrique, il est possible d'inclure une fonction de zone morte S_j :

$$Y(t) = \sum_{j=1}^{m} w_{s_j} S_j(\sum_{i=1}^{n} w_i H_i(x(t))) \quad (33)$$

et la fonction S_j définie par :

$$S_j(x) = \begin{cases} max(0,\ x - d_j),\ \forall d_j > 0 \\ x,\ d_j = 0 \end{cases} \quad (34)$$

Modèles de frottement

L'actionneur slip & stick étant basé sur un phénomène de frottement, il est donc nécessaire d'étudier les modèles de frottement. Ces modèles peuvent être regroupés en deux grandes familles :

- les modèles statiques, directement basés sur la théorie de Coulomb
- et les modèles dynamiques, plus évolués basés sur l'utilisation d'équations différentielles et, permettant une meilleure représentation des phénomènes de frottement.

Le modèle statique le plus simple, dit frottement de Coulomb [14], est uniquement basé sur l'équation suivante :

$$F_r(t) = \begin{cases} F, \ v = 0 \\ F_c.sign(v), \ v \neq 0 \end{cases} \quad (35)$$

Dans le cas d'un frottement présentant une lubrification par exemple [59], l'ajout d'une composante proportionnelle à la vitesse de glissement est envisagé :

$$F_r(t) = \begin{cases} F, \ v = 0 \\ F_c.sign(v) + F_v.v, \ v \neq 0 \end{cases} \quad (36)$$

Enfin, afin de traduire l'effet de décollage d'un élément frottant, c'est-à-dire que l'effort nécessaire à la mise en mouvement d'un mobile est supérieur à l'effort résistant quand celui-ci est en mouvement, Stribeck [67] propose l'équation suivante :

$$F_r(t) = \begin{cases} F, \ v = 0 \\ F_c.sign(v) + F_v.v + F_s e^{-\left(\frac{|v|}{v_s}\right)^n} sign(v), \ v \neq 0 \end{cases} \quad (37)$$

Il est couramment admis que $n = 2$ dans l'écriture du phénomène de Stribeck.

Nom de la variable	Symbole	Unité
Force de Coulomb	F_c	N
Force visqueuse	F_v	$N.s/m$
Force de Stribeck	F_s	N
Vitesse de Stribeck	v_s	m/s
Vitesse de glissement	v	m/s

Table 9 : Variables des modèles de frottement statiques

Les modèles dynamiques de frottement ont été introduits afin de pouvoir traiter de façon plus efficace les phénomènes de pré-glissements (issus de la déformation des interfaces de frottement avant le décollage), des phénomènes d'hystérésis aussi bien pour les phases de pré-glissement que lors du glissement ainsi que l'insertion de dépendance face à des vitesses de variation des paramètres.

Dahl propose, par un modèle du même nom [15], de décrire le frottement par une

équation différentielle. Ce modèle est relativement simple, ne comportant que trois paramètres, et ne dépendant que du déplacement relatif des interfaces de frottement. Deux inconvénients potentiels à ce modèle est de ne pas prendre en compte le phénomène de Stribeck ni effets de cisaillement.

$$\frac{dF_r}{dt} = \sigma_0 v \left(1 - \frac{F_r}{F_c}.sign(v)\right)^{\alpha} \tag{38}$$

où on pose couramment $\alpha = 1$.

Bliman et Sorine [5] ont proposé un modèle dérivé de celui de Dahl afin de prendre en compte le phénomène de Stribeck. Cette modification est introduite en remplaçant le temps par une variable s permettant une représentation de la distance de glissement parcourue puis en insérant une équation différentielle du second ordre afin de permettre un dépassement lors d'une inversion de la vitesse de glissement.

$$\begin{cases} s = \int_0^t |v(\tau)| d\tau \\ \frac{d^2F_r}{ds^2} + 2\zeta\omega \frac{dF_r}{ds} + \omega^2 F_r = \omega^2 F_c.sign(v) \end{cases} \tag{39}$$

Il apparait, cependant, que ce modèle ne peut capturer qu'une partie restreinte de l'effet Stribeck par une non-dépendance directe à la vitesse de glissement ainsi qu'une représentation lors du passage d'un régime dynamique à un régime statique est difficile.

Le modèle de LuGre [10] a été créé dans le but de répondre à ces diverses attentes en utilisant une représentation du frottement, basée sur la description d'une interface de frottement comme des lames en flexion. La variable d'état z permet de représenter la flèche de la flexion d'une lame.

$$\begin{cases} \frac{dz}{dt} = v - \frac{\sigma_0}{g(v)} z|v| \\ g(v) = F_c + F_s e^{-|\frac{v}{v_s}|^2} \\ F_r = \sigma_0 z + \sigma_1 \frac{dz}{dt} + \sigma_2 v \end{cases} \tag{40}$$

Le modèle de Leuven [68] apparait comme étant une amélioration du modèle de LuGre par une meilleure modélisation du pré-glissement en utilisant un opérateur d'hystérésis.

$$\begin{cases} \frac{dz}{dt} = v\left(1 - sign\left(\frac{F_h(z)}{s(v)}\right) \left|\frac{F_h(z)}{s(v)}\right|^n\right) \\ s(v) = \left(F_c + F_s e^{-|\frac{v}{v_s}|^{\delta}}\right).sign(v) \\ F_r = F_h(z) + \sigma_1 \frac{dz}{dt} + \sigma_2 v \end{cases} \tag{41}$$

La grandeur $F_h(z)$ est issue de l'opérateur d'hystérésis. On peut retrouver comme opérateur un modèle de Maxwell par exemple.

Enfin, un dernier modèle peut être cité : le GMS ou Generalized Maxell Slip [2]. Ce modèle se constitue comme une somme de sous-modèles élémentaires et permet une représentation fine des différents phénomènes intervenant dans le cadre d'éléments frottants. Dans une phase de pré-glissement, un sous-modèle i est décrit par :

$$\frac{dF_i}{dt} = k_i.v \tag{42}$$

La limite de la phase de pré-glissement est atteinte lorsque (condition de commutation):

$$\begin{cases} F_i = \alpha_i s(v) \\ s(v) = \left(F_c + F_s e^{-\frac{v}{v_s}^2}\right).sign(v) \end{cases} \tag{43}$$

En phase de glissement, le sous-modèle se décrit alors par :

$$\frac{dF_i}{dt} = sign(v).\alpha_i C\left(1 - \frac{F_i}{\alpha_i s(v)}\right) \tag{44}$$

La condition de retour en phase de pré-glissement est le passage de la vitesse de glissement par zéro.

La force de frottement totale est alors donnée par les relations suivantes :

$$\begin{cases} \sum_{i=1}^{N} \alpha_i = 1 \\ F_r = \sum_{i=1}^{N} F_i + F_v.v \end{cases} \tag{45}$$

Dans la formulation présentée ci-dessus, les différentes fonctions de Stribeck sont identiques. Il est éventuellement possible de les différencier.

3.1.2.4 Pré-actionneur

Le pré-actionneur est l'élément d'interface entre la partie commande et l'actionneur, il a donc pour charge de réaliser une adaptation énergétique entre une partie commande basse tension (ici une carte électrique de commande) et une partie haute tension (due à la présence de l'élément piézoélectrique de l'actionneur).

Schématiquement, la haute tension alternative est transformée en haute tension continue (par redressement), puis celle-ci peut-être descendue à une tension inférieure pour les différents composants électroniques de commande, ou bien encore modulée pour fournir l'énergie adéquate à l'actionneur.

Deux modélisations sont possibles pour cet élément :

- une représentation par simulation directe des composants électroniques, comme dans le logiciel SPICES où le schéma électronique est reproduit en utilisant les modèles de composants fournis (si disponibles) par les fabricants
- une représentation de type macro-modèle. Le système est alors être caractérisé par une bande passante, un slew-rate, une limitation d'intensité, un rendement global ...

Nom de la variable	Symbole	Unité
Gain d'amplification	K	
Limite de bande passante	F_{max}	Hz
Slew-rate	sr	V/s
Intensité maximale	I_{max}	A
Rendement global	η	

Table 10 : Variables du pré-actionneur dans un macro-modèle

3.1.2.5 Capteur de débit

Comme pour le pré-actionneur deux types de modélisation sont possibles :

• une modélisation fine par reproduction directe des phénomènes physiques utilisés : mesure de température après réchauffement d'un volume élémentaire, perturbation de flux électromagnétique, induction, ...

• ou bien une modélisation macroscopique du capteur : limite de sensibilité (dead-zone), saturation (limite de mesure inférieure et supérieure), temps de réaction et de mesure, et enfin fonction de transfert. A noter la possibilité de prise en compte d'incertitude des mesures ainsi que d'introduction de bruit.

Dans le cas présenté, le capteur fait une transformation du débit mesuré vers une tension interprétable par un équipement électronique.

Nom de la variable	Symbole	Unité
Gain de transformation	K	$V.s/m^3$
Limite de sensibilité	Q_lim	m^3/s
Limite de mesure	Q_max	m^3/s
Temps de mesure	T_m	s
Temps de réveil	T_r	s

Table 11 : Variables du capteur de débit dans un macro-modèle

3.1.2.6 Asservissement et stratégie de commande

Les différents sous-systèmes cités précédemment doivent maintenant être reliés afin de former un système en boucle fermée. Plusieurs éléments sont nécessaires à la création de cette boucle :

- les sous-systèmes mis en œuvre sont de nature analogique, alors que les systèmes de contrôles et d'interfaces utilisateurs sont numériques. Une adaptation est donc à réaliser en utilisant des convertisseurs numériques / analogiques et analogiques / numériques
- la consigne donnée au système par l'utilisateur est une hauteur de fluide, l'information délivrée par le capteur est l'image d'un débit. Il est donc nécessaire de réaliser un estimateur de la hauteur de fluide contenu dans le réservoir

De par sa nature numérique, le système doit posséder une ou plusieurs horloges de référence permettant de définir un temps d'échantillonnage du système T_e. Compte-tenu du comportement slip & stick de l'actionneur, on utilisera une commande de type tout ou rien dans un premier temps. Le système aura aussi pour charge de générer la forme d'onde nécessaire au bon fonctionnement de l'actionneur.

Estimateur de hauteur de fluide

En considérant les équations de la partie fluidique, ainsi que des conditions limites de pression connues, il est possible d'établir une relation entre hauteur de fluide du réservoir et le débit mesuré. Soit q_{mesure} le débit du fluide et A la section du capteur fluide (identique au tuyau), la vitesse de fluide v est donnée par :

$$v = \frac{q_{mesure}}{A} \tag{46}$$

La somme des pertes de pression étant égale à la hauteur totale de la pression générée par la colonne d'eau (réservoir compris), il est possible d'écrire :

$$h_{res} = \frac{\sum pertes_{pression} - \rho g L}{\rho g} \tag{47}$$

Cette dernière fournit alors un estimateur de la hauteur d'eau contenue dans le réservoir. Cependant cette équation fait intervenir le coefficient d'ouverture de la valve. Ne possédant pas de capteur de position de l'actionneur dans le système, il est encore une fois nécessaire de réaliser un estimateur de position de celui-ci.

Afin de le réaliser, le choix se porte non vers l'insertion du modèle de l'actionneur (à cause d'une perspective d'utilisation d'une électronique embarquée de faible puissance calculatoire) mais en faisant directement intervenir un méta-modèle relatant la taille des pas effectués par l'actionneur en fonction d'une période du signal d'excitation envoyé à celui-ci.

La mise en place de ce méta-modèle se fait à partir de mesures répétées du déplacement de l'arbre réel permettant alors de créer un modèle statistique de déplacement de l'actionneur. Ce modèle peut, par exemple, être réalisé par une distribution de type Weibull.

3.2 Mise en place d'un banc de test

Une revue des modélisations possibles étant effectuée, il apparait que la partie fluidique relève de modèles connus. Seule la vanne nécessite éventuellement de réaliser un simple test de débit en fonction de la différence de pression appliquée à ses extrémités. Il peut être envisagé de remplacer ce test par une simulation numérique d'écoulement fluide.

L'essentiel des tests à effectuer se concentre donc sur le pré-actionneur et l'actionneur.

3.2.1 Placement et choix des capteurs:

3.2.1.1 Le pré-actionneur

Le pré-actionneur étant un sous-système purement électrique, deux grandeurs sont mesurables : tension et intensité. Les différents paramètres de celui-ci sont accessibles via des tests mettant en œuvre un générateur de basse fréquence (GBF) ainsi qu'un oscilloscope (ou bornier d'acquisition) équipé de sonde de tension (dans le cas présent un pont diviseur à forte impédance sera préféré), et d'un capteur d'intensité par effet électromagnétique.

3.2.1.2 L'actionneur

En ce qui concerne l'actionneur proprement dit, trois composants nécessitent une identification : le composant piézoélectrique amplifié, la masse d'inertiel ainsi que l'arbre. Quatre capteurs ou instruments de mesure sont retenus : le pont diviseur de tension, le capteur d'intensité, un capteur de déplacement laser ainsi qu'une balance. Dans l'idéal, un capteur d'effort ainsi qu'un accéléromètre sont utilisables. Cependant, compte-tenu de la géométrie de l'actionneur, l'insertion d'un capteur d'effort est relativement difficile et l'insertion d'un accéléromètre entraine un changement non négligeable des masses en jeu.

3.2.2 Acquisition des données

Le choix est fait d'utiliser un bornier d'acquisition pour obtenir une mesure synchrone des données issues des différents capteurs. Cette acquisition se fera à une fréquence de $50KHz$, ce qui permet, d'après le théorème de Shannon, d'observer des phénomènes jusqu'à $25KHz$. L'acquisition se fait sur une entrée à $\pm 10V$ codé sur $16bits$, soit une sensibilité de $0.3mV/bits$.

3.3 Identification des paramètres

L'identification des paramètres consiste à établir, par différents moyens mathématiques, les valeurs fixes d'un modèle pour correspondre à une certaine réalité expérimentale. Certains de ces paramètres peuvent être facilement identifiables, et de façon unique (unicité de solution), par des expériences simples, ou bien sont fortement interconnectées entres eux, rendant toute interprétation physique difficile.

Trois grandes méthodes d'approches sont présentées ci-dessous :

- l'identification directe par application d'équations simples ou bien encore par l'utilisation d'abaques
- l'utilisation du modèle inverse qui permet, par la connaissance des sorties et des entrées, de retrouver les paramètres
- et l'identification inverse qui va se charger de calculer un modèle avec des couples de paramètres et d'en déterminer la pertinence.

3.3.1 Identification directe

L'identification directe d'un paramètre se fait de façon simple et directe par l'utilisateur. Cette identification est réalisée par la mise en œuvre d'une expérience simple dont les modèles en jeu sont simples et connus, ou bien suivant des abaques.

Par exemple, la connaissance de la tension aux bornes d'une résistance déterminée engendre la création d'un courant. Il est alors aisé, par la mesure de cette tension et du courant, de déterminer la valeur de la résistance :

$$R = \frac{U_{mesure}}{I_{mesure}} \tag{48}$$

L'utilisation d'un graphique peut aussi entrainer une détermination de paramètres. Cette possibilité est offerte, en matériaux, par l'analyse de la courbe de traction d'une éprouvette.

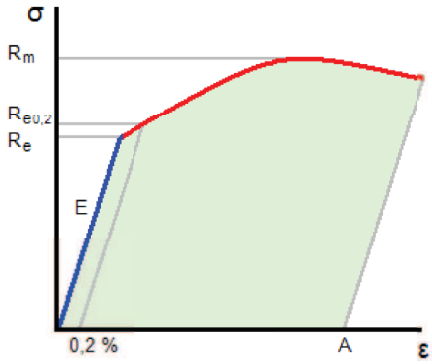

Figure 17: Exemple de courbe de traction, Wikipédia

De cette courbe, et par lecture directe, il est possible de déterminer les différentes valeurs de résistance ainsi que le module de Young.

Enfin, dans le cas d'un système suivant une mise en équation différentielle du second ordre avec dépassement, il est possible d'utiliser des abaques et pour la détermination des paramètres : dans le cas d'un essai de réponse à un échelon indiciel:

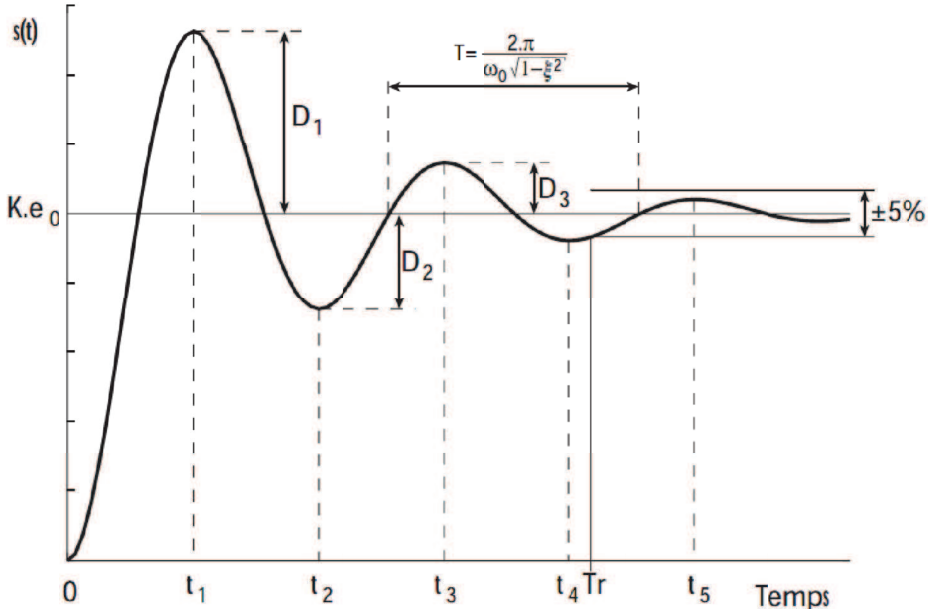

Figure 18: Exemple de réponse d'un système du second ordre

• une lecture du temps pour une réponse à 5% permet d'en déterminer l'amortissement

• la lecture de la valeur moyenne finale permet de déterminer le gain

• la lecture du temps du 1er dépassement permet de déduire la pulsation naturelle du système.

3.3.2 Identification par modèle inverse

Comme indiqué, l'identification par modèle inverse se fait par l'inversion d'un modèle donné. Cette méthode d'inversion de modèle est généralement utilisée pour des modèles linéaires car dans ces cas là l'inversion est directe. Pour les modèles non-linéaires, il est alors nécessaire d'effectuer préalablement une opération de linéarisation (développements limites, ...).

De façon générale, soit un modèle :

$$Y = AX \tag{49}$$

où Y est la sortie, X l'entrée et A la matrice des paramètres.

La détermination des paramètres A peut se faire, sous réserve de l'existence de l'inversion de matrice :

$$A = YX^{-1} \tag{50}$$

De fait cette méthode peut paraitre intéressante pour l'identification des paramètres mais dans le cas où une mesure réelle et donc bruitée doit être dérivée dans l'écriture du modèle inverse, la lecture des résultats peut devenir délicate.

3.3.3 Identification inverse

L'identification inverse a pour but de pallier les problèmes de l'identification par modèle inverse. Cette méthode est capable d'accepter des modèles importants (en nombre de paramètres) et pouvant être non-linéaires. En effet, l'identification inverse se compose de trois éléments :

- un modèle direct dont on fait varier les valeurs de paramètre
- un comparateur qui a pour but de fournir une valeur de pertinence du résultat du modèle par rapport à une mesure expérimentale
- un processus permettant de faire varier les paramètres, de façon à augmenter la pertinence des résultats du modèle

De ce fait, l'utilisation du modèle direct permet de pallier la possible difficulté de l'inversion du modèle, et la comparaison avec des mesures permet l'utilisation de résultats expérimentaux fortement bruités. La difficulté de mise en œuvre de cette technique se situe sur le choix et l'implantation du processus permettant la modification des valeurs de paramètre. Ce processus est une méthode d'optimisation.

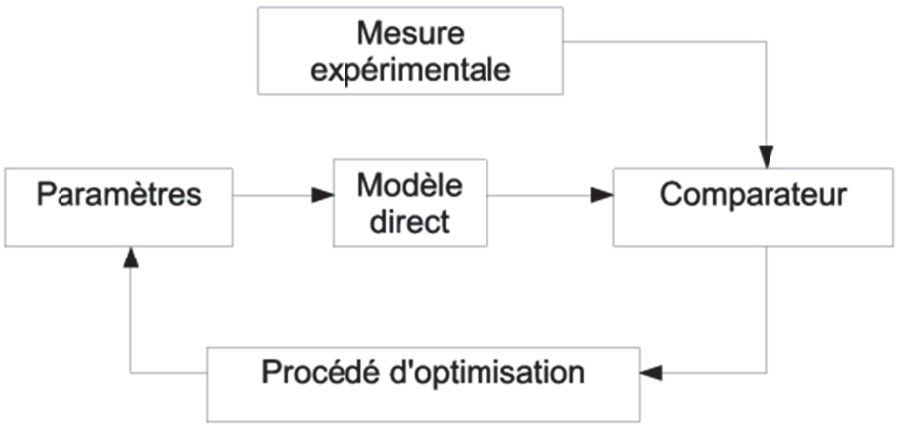

Figure 19 : Schéma d'une identification inverse

3.4 Conclusion

Ce chapitre a permis, en partant d'un schéma de système mécatronique global et par l'observation des sous-systèmes déjà existants, de définir des modèles génériques de comportement ainsi que de dégager clairement les différents éléments d'interaction. Afin de pouvoir établir les valeurs des paramètres composants les modèles, des tests doivent être réalisés. Cela passe par la création de bancs de tests ainsi que leur instrumentation et la création de plans de test. Une fois les données recueillies, le choix du procédé d'identification est à réaliser. Les sous-systèmes utilisés dans le cadre de cette thèse étant non-linéaires, l'identification inverse est privilégiée. Cette méthode est approfondie dans le chapitre suivant.

4 Méthodes d'optimisation pour l'identification inverse et l'amélioration des performances

Sommaire

Le chapitre précédent a permis de mettre en place un modèle mathématique d'un système mécatronique. Afin de pouvoir exploiter ce modèle, il est nécessaire d'en déterminer les paramètres. Ce chapitre présente différents outils et méthodes nécessaires au traitement des problèmes d'identification inverse et d'optimisation des systèmes de façon générale afin de pouvoir choisir la méthode la plus adaptée à notre objectif. Trois familles de méthodes d'optimisation sont présentées :

- les méthodes à base de gradient particulièrement adaptées à la résolution de système linéaires mais présentant des défauts majeurs dans le cadre de systèmes non-linéaires que l'on peut retrouver en mécatronique,
- les méthodes de recherches directes permettant de travailler sur des systèmes non-linéaires mais pouvant présenter certains défauts tels que le déterminisme,
- enfin les méthodes dites méta-heuristiques qui sont particulièrement adaptées à la résolution de problèmes difficiles. Un focus sera réalisé sur un des procédés d'optimisation méta-heuristique amélioré dans le cadre de la thèse.

La robustesse des résultats issus de ces méthodes est un élément important pour déterminer la qualité des résultats. Deux procédés de prise en compte de cette robustesse par des méthodes à priori et à posteriori sont présentés. En conclusion, on retiendra un algorithme dont la mise en œuvre est rapportée dans les chapitres suivants dans le cadre d'identifications inverses et d'optimisations numériques et expérimentales.

4.1 Optimisation et identification inverses, deux problématiques semblables

L'optimisation des performances d'un système revient à rechercher un extremum (minimum ou maximum) d'une fonction objectif que l'utilisateur définit.

L'identification inverse a pour but de déterminer les valeurs de paramètres permettant à un modèle de s'approcher d'une courbe expérimentale suivant les besoins de l'utilisateur. La fonction objectif est ici clairement définie.

Ces deux approches sont donc identiques dans le sens où une fonction objectif existe, que l'on cherche par exemple à minimiser.

4.1.1 Fonction objectif

Le problème peut s'écrire avec $f(x)$ la fonction objectif et x les paramètres à optimiser :

$$\begin{cases} min(f(x)) \\ x \in \Re^n \mapsto \Re^m \end{cases} \tag{51}$$

Dans le cas où l'espace d'arrivée de la fonction objectif est de dimension $m > 1$, on parle d'optimisation multi-objectif sinon de mono-objectif. Il est possible de traiter un problème multi-objectifs, en utilisant la courbe de Paretto [52] par exemple mais aussi de créer une

nouvelle fonction objectif réalisant une combinaison linéaire des éléments de résultats afin de revenir à un problème mono-objectif.

La courbe de Pareto est la représentation de l'ensemble des combinaisons de solutions possible dans le cadre d'une optimisation multi-objectifs. Une écriture mathématique de cette courbe dans le cadre d'une fonction à deux objectifs peut être la suivante :

$$f = \lambda f_1 + (1 - \lambda f_2), \lambda \in [0,1] \quad (52)$$

Cette équation se traduit alors graphiquement par une courbe que l'on nomme front de Pareto (Fig. 20).

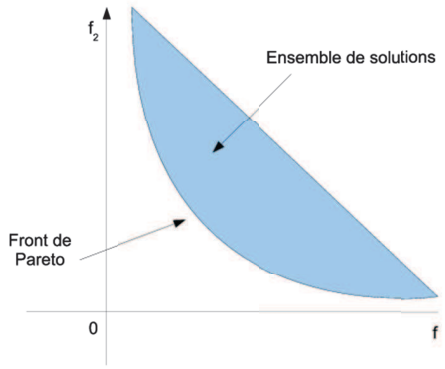

Figure 20: Courbe de Pareto

Afin d'obtenir une fonction objectif la plus performante et pertinente possible, celle-ci doit être convexe c'est-à-dire en forme de bol tel que la fonction x^2. Cette convexité est obtenue en vérifiant la définition suivante :

Une fonction f d'un intervalle I de $\Re^n$ dans $\Re^m$ est réputée convexe lorsque $\forall x \in I$ et $\forall \lambda \in [0,1]$

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2) \quad (53)$$

Cette convexité permet alors d'établir le théorème suivant :

Si X est une partie d'un espace vectoriel E et si f est strictement convexe sur X alors $min(f)$ possède au plus une solution.

Ce dernier théorème établi l'unicité de la solution. Cependant ceci est un cas idéal d'optimisation car les fonctions objectif contiennent souvent des minima locaux.

Exemples de fonctions objectif

Les normes sont très souvent employées comme fonctions objectif :

- la norme 2 ou euclidienne : cette norme permet d'établir un écart entre une courbe expérimentale et une courbe théorique. Elle est notamment utilisée lors d'identifications

inverses.

• la norme infinie : cette norme calcule l'écart maximal entre deux courbes. Elle peut être employée en automatique afin de minimiser par exemple un dépassement.

• la norme 1 ou Manhattan : cette norme fait la somme des valeurs absolues des composantes d'un vecteur. Celle-ci peut être utilisée pour des calculs de distances sur une grille.

4.1.2 Fonctions de contrainte

La fonction objectif est à compléter avec des fonctions de contrainte. Les fonctions de contrainte ont pour but de limiter la zone de recherche de solutions par l'utilisateur. Les raisons peuvent être liées à la physique même du système à étudier (masse d'un composant) ou encore une volonté d'obtenir la meilleure solution suivant le matériel disponible. Les fonctions de contrainte sont généralement découpées en deux parties complémentaires :

- q la fonction de contrainte d'inégalité
- et h la contrainte d'égalité.

Alors le processus d'optimisation s'écrit :

$$\begin{cases} min(f(x)) \\ x \in \Re^n \mapsto \Re^m \\ q(x) \leq 0 \\ h(x) = 0 \end{cases} \tag{54}$$

Les fonctions de contrainte ne sont pas forcément évidentes à traiter directement. Il est alors possible de faire appel à des fonctions dites de pénalisations.

4.1.3 Fonctions de pénalisation

Les fonctions de pénalisation ont pour but de faciliter le traitement des fonctions de contrainte en passant d'une optimisation sous contraintes à une optimisation sans contraintes. Pour cela deux techniques peuvent être utilisées :

• Les multiplicateurs de Lagrange :

Dans le cas où l'on a une fonction de contrainte, les multiplicateurs de Lagrange permettent l'ajout de la contrainte dans la fonction objectif à minimiser. Ainsi au lieu de minimiser $f(x)$, on minimise $f(x) + \lambda_1 h(x) + \lambda_2 q(x)$. L'inconvénient de cette technique peut être l'augmentation de la dimension du problème par l'ajout des multiplicateurs λ.

• La pénalisation directe :

La pénalisation directe consiste à vérifier si le couple de valeurs à tester respecte les contraintes établies précédemment. Si les contraintes sont vérifiées alors on procède à

l'évaluation de la fonction objectif, sinon on assigne au résultat une valeur issue d'une autre fonction qui aura pour but de décourager le processus d'optimisation de retenir ce couple de variables. Cette seconde fonction est à définir par l'utilisateur pour chaque optimisation par rapport aux résultats attendus de la fonction objectif. L'utilisateur, de plus, devra faire attention à créer une fonction assurant une convergence de l'algorithme vers la zone de variables potentiellement acceptables définie par les fonctions de contrainte.

4.2 Méthodes à base de gradient

Les méthodes à base de gradient sont couramment utilisées comme processus d'optimisation. Ces méthodes ont l'avantage d'être relativement simples à mettre en œuvre et peu couteuses en termes de calculs. L'utilisation de telles méthodes peut être conseillée pour la résolution de problèmes linéaires ou quasi-linéaires.

4.2.1 Cas idéal

Dans le cas idéal, la fonction objectif est deux fois dérivable sur l'espace de recherche et est connue. Il est alors possible de trouver x le vecteur de variables tel que :

$$\nabla f(x) = 0 \tag{55}$$

L'annulation du gradient permet de déterminer les points d'extremum de la fonction. Il reste à déterminer si ceux-ci sont des minima, maxima ou points selles par l'utilisation de la matrice Hessienne :

$$H(f(x))_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \tag{56}$$

Dans le cas où les valeurs de la diagonale ont le même signe alors on est en présence d'un extremum sinon d'un point selle. Si la diagonale est positive alors on est en présence d'un minimum sinon d'un maximum.

Si la fonction objectif n'était pas strictement convexe sur l'intervalle de recherche, l'évaluation de la fonction aux points retenus permet alors d'établir les minima locaux et globaux sur ce domaine.

2.2 Descente de gradient à pas fixe

Dans le cas d'une méthode de gradient à pas fixe, la matrice Hessienne de la fonction n'est pas connue mais l'équation de la fonction et ses dérivées le sont. Cet algorithme permet à

un point X_0 de descendre le long de la pente déterminée par le gradient. Le point va lors converger vers un minimum. La méthode est décrite par l'algorithme suivant :

Soit X_0 un point de départ dans l'espace de recherche E. On calcule la suite, avec d_k le pas de descente et g_k la direction de descente à l'itération k :

$$\begin{cases} X_{k+1} = X_k + d_k . g_k \\ g_k = -\nabla f(X_k) \end{cases} \quad (57)$$

Le problème ici est que d_k est constant et donc pas forcément optimal. Il est alors possible de faire varier celui-ci dans le temps afin d'obtenir un algorithme de descente à pas optimal.

4.2.3 Méthode de Newton

La méthode de Newton nécessite de pouvoir calculer la dérivée première ainsi que la dérivée seconde de la fonction. Cette méthode est définie par l'équation de récurrence suivante :

$$X_{k+1} = X_k - d_k H(f(X_k))^{-1} \nabla f(X_k) \quad (58)$$

La prise en compte de la dérivée seconde permet d'avoir une convergence plus rapide que par la descente de gradient. La contrepartie est un coût de calcul supplémentaire par le calcul du Hessien puis de son inversion.

4.2.4 Méthode du Quasi-Newton

La méthode du Quasi-Newton ne nécessite pas la connaissance de la matrice Hessienne de la fonction. Celle-ci est remplacée par un estimateur basé sur l'évaluation de la dérivée première qui est mis à jour à chaque itération. On peut par exemple citer le BFGS [63, 9, 20, 24] comme méthode de mise à jour :

$$\begin{cases} X_{k+1} = X_k - d_k \widehat{H}(f(X_k))^{-1} \nabla f(X_k) \\ s_k = -d_k \widehat{H}(f(X_k))^{-1} \nabla f(X_k) \\ y_k = \nabla f(X_{k+1}) - \nabla f(X_k) \\ \widehat{H}_{k+1} = \widehat{H}_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{\widehat{H}_k^{-1} y_k s_k^T + s_k y_k^T \widehat{H}_k^{-1}}{s_k^T y_k} \end{cases} \quad (59)$$

4.3 Méthodes par recherche directe

Les méthodes par recherche directe ne nécessitent pas l'utilisation du gradient et permettent alors une certaine insensibilité aux non-linéarités. Mise à part la méthode de Monte Carlo, ces méthodes nécessitent le choix d'un point de départ et sont déterministes.

4.3.1 Monte Carlo

La méthode dite de Monte Carlo [45] a été introduite juste après la seconde guerre mondiale pour l'étude de problèmes à grandes dimensions. Elle est utilisée pour l'étude des sensibilités, mais peut aussi servir à l'obtention de solutions dans des cas d'optimisation. Cette méthode consiste à effectuer des évaluations de la fonction objectif aléatoirement sur le domaine de recherche jusqu'à la découverte d'une solution admissible. Un des exemples d'utilisation est le calcul de la surface d'un cercle afin d'obtenir une estimation de π (Fig. 21). Le problème majeur de cette méthode est le nombre important d'évaluations à réaliser.

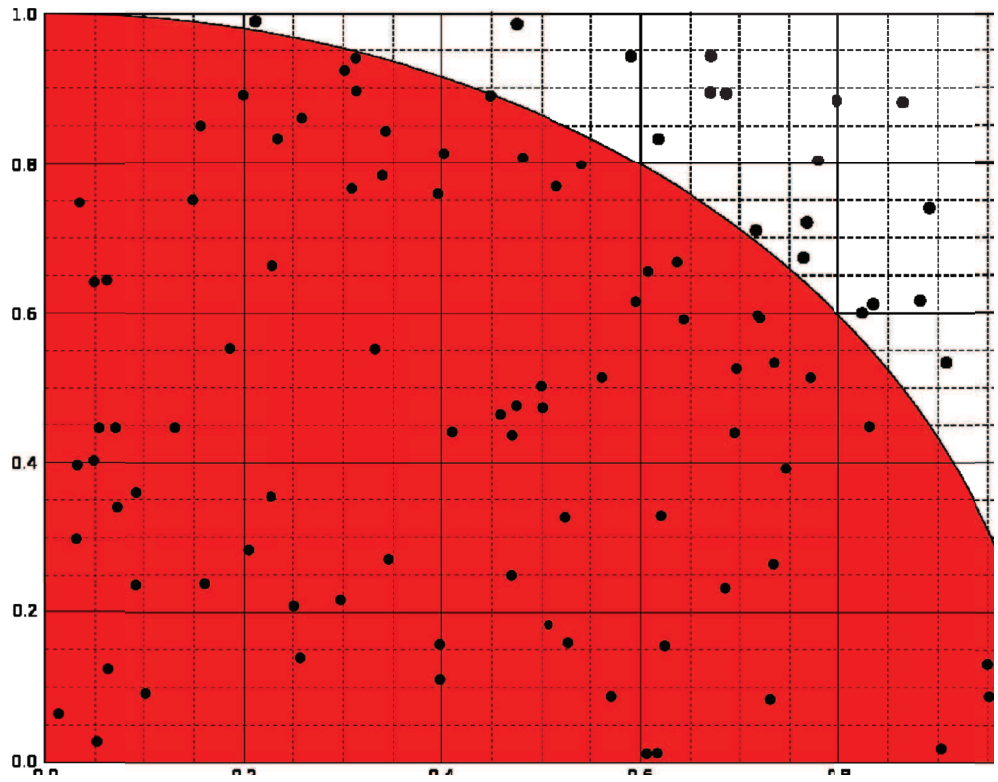

Figure 21 : Exemple de calcul de π par la méthode de Monte Carlo, source Wikipédia

4.3.2 Simplex

Les méthodes de simplexe, telles que Nelder-Mead [51] ou Torczon [70], sont basées sur les transformations d'un polytope. Un polytope est une figure géométrique constituée de $N+1$ sommets, N étant la dimension du problème dans l'espace de recherche. Pour un problème de dimension 2, le polytope associé est un triangle (Fig. 22). Le simplexe est le terme mathématique désignant une généralisation d'un triangle à N dimension. Ces méthodes peuvent être utilisées dans des cas de problèmes non-linéaires et sont déterministes à cause du choix initial du polytope de départ et de l'absence de nombres aléatoires.

Par exemple l'algorithme du simplex de Nelder-Mead peut s'écrire ainsi :

- Choix d'un polytope à N+1 dimensions créant un simplex $x_1, \ldots, x_{N+1}$
- Evaluation du simplex
- Classement suivant le résultat $f(x_1) \leq \ldots \leq f(x_{N+1})$
- Calcul d'un centre de gravité x_g à partir des N meilleurs résultats
- Création d'un point réfléchi x_r à partir de x_{N+1} et du centre de gravité par $x_r = x_g + \alpha(x_g - x_{N+1})$
- Si $f(x_r) < f(x_1)$, on calcul un point x_e d'étirement du simplex par $x_e = x_g + \gamma(x_g - x_{N+1})$. Si $f(x_e) < f(x_r)$, x_{N+1} est remplacé par x_e sinon par x_r puis retour au classement du simplex dans les deux cas.
- Sinon si $f(x_n) < f(x_r)$, on calcul un point de contraction x_c par $x_c = x_{N+1} + \rho(x_g - x_{N+1})$. Si $f(x_c) \leq f(x_r)$ alors x_{N+1} est remplacé par x_c puis retour au classement du simplex.
- Sinon on effectue une réduction du simplex par la relation suivante : $x_i = x_1 + \sigma(x_i - x_1)$

Par défaut, ces valeurs peuvent être utilisées pour les différents paramètres : $\alpha = 1$, $\gamma = 2$, $\rho = 0.5$ et $\sigma = 0.5$. Il est à noter que cet algorithme est connu dans Matlab comme étant fminseach .

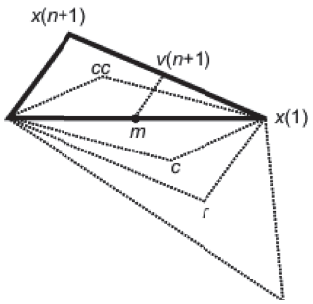

Figure 22 : Exemple d'un simplex et de ses transformations en dimension 2, source MathWorks

4.3.3 Pattern search

Proposée par Hooke et Jeeves [29], cette méthode consiste a créer une croix de dimension N autour d'un point initial. Les points extrêmes de cette croix sont évalués ainsi que le point central pour faire évoluer celle-ci vers le minimum global.

L'algorithme se présente sous la forme suivante :

• soit un point x_0, on définit la croix par $x_i = x_0 \pm \Delta_0 B$ où B est une base orthonormée de l'espace de recherche et la variable Δ_0 une variable permettant de définir la taille de la croix;

• à l'itération n, si un des points extrêmes possède une meilleur solution que le point central alors la croix est transférée en ce point;

• dans ce cas une augmentation de Δ_n peut être réalisée;

• sinon la croix reste centrée sur x_{n-1} est une contraction de celle-ci est réalisée par une diminution de Δ_n.

4.4 Méthodes méta-heuristiques

Le terme méta-heuristiques provient des deux termes grecs meta et heuriskein qui ont respectivement pour signification au-delà et trouver. Une interprétation de cette association peut être trouvée à haut-niveau. En effet les algorithmes méta-heuristiques sont quasi adaptés à tout problème d'optimisation moyennant certaines modifications de ceux-ci. Ces algorithmes sont utilisés pour la résolution de problèmes difficiles c'est à dire non-linéaires, pouvant varier dans le temps (dynamique), bruités et pouvant encore posséder de nombreux minima locaux. Cet avantage doit être contrebalancé par un inconvénient majeur qui est le nombre élevé d'évaluations de la fonction objectif. Cette partie présente quelques algorithmes méta-heuristiques principaux et un focus est réalisé sur l'un d'eux afin de présenter deux modifications apportant des améliorations qui seront ensuite vérifiées sur des fonctions tests.

4.4.1 Recuit simulé

Le recuit simulé est issue de l'observation de l'état énergétique de la matière lors d'un recuit pour augmenter la taille des agrégats et éliminer les défauts. Cette méthode a été développée par Kirkpatrick, Gelatt et Vecchi [35] en 1983. Lors d'un recuit, une élévation de température se produit afin de redonner de l'énergie aux particules et donc sortir celles-ci d'un minimum local et de permettre le changement de lieu. Puis un refroidissement se produit pour faire converger les particules vers un minimum énergétique et donc le minimum de la fonction objectif.

L'algorithme peut se présenter de la façon suivante :

• Soit une particule initiale x_0 possédant une énergie $f(x_0)$ à la température T_0;

• à l'itération suivante n, une modification est apportée à x_{n-1}, si $\Delta E =$

$f(x_{n-1}) - f(x_n) < 0$ alors x_n devient la meilleure solution courante sinon celle-ci est acceptée avec une probabilité de $e^{\frac{\Delta E}{T_n}}$.

La température est un élément très important pour cet algorithme : une température importante amène la probabilité d'acceptation proche de 1 mais en contrepartie permet d'obtenir une particule très mobile. La loi de décroissance de la température est soumise à de nombreuses variantes mais par défaut il est possible d'utiliser une décroissance linéaire de celle-ci en insérant des paliers.

4.4.2 Colonies de fourmis

Les colonies de fourmis, comme son nom l'indique, sont inspirées du comportement social de celles-ci. Cet algorithme a été proposé par Dorigo [17] dans les années 90. Le principe de fonctionnement est le suivant :

- Chaque solution possible est représentée par une fourmi qui parcours alors l'espace de recherche;
- si celle-ci trouve un résultat intéressant, alors elle laisse une trace de phéromone sur le chemin de retour;
- les autres fourmis sont alors attirées par cette trace et peuvent renforcer celle-ci en déposant aussi des phéromones;
- à la fin du processus, il ne reste alors qu'une seule piste menant à la solution.

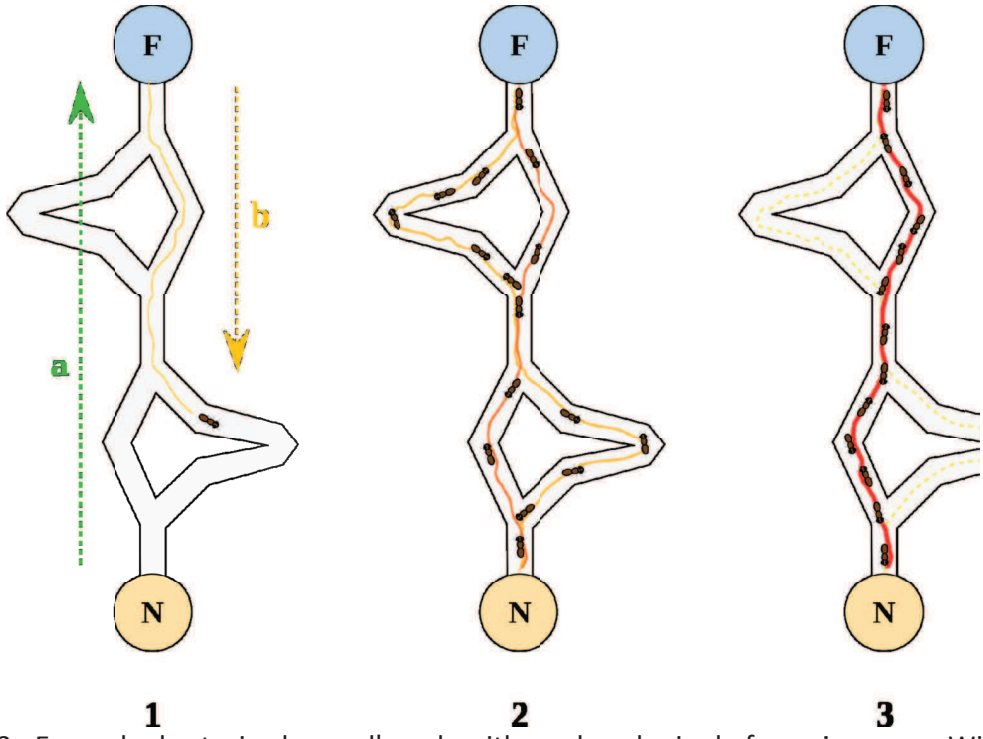

Figure 23 : Exemple des trois phases d'un algorithme de colonie de fourmis, source Wikipedia

4.4.3 Algorithmes génétiques

Les algorithmes génétiques sont basés sur des observations liées à la génétique ainsi qu'à l'évolution naturelle d'une population. Les premiers travaux sur ces algorithmes ont été menés par Holland [28] en 1962.

Les algorithmes génétiques ont la particularité de ne pas se baser directement sur les valeurs de variables mais sur un codage binaire, créant un gène, de celles-ci afin de pouvoir effectuer les opérations suivantes :

• croisement :

le croisement consiste à partir de deux individus à générer un nouvel individu à partir d'un mélange aléatoire et équitable du matériel génétique.

• mutation :

la mutation consiste à remplacer un gène par un autre gène tiré aléatoirement dont la valeur peut éventuellement être proche de celle du gène initial.

• sélection :

A la fin de chaque itération, les individus sont sélectionnés suivant leurs résultats. Les meilleurs individus seront conservés alors que les plus faibles seront recyclés suivant diverses méthodes. On peut par exemple citer la Roulette wheel selection ou encore la Stochastic remainder without replacement selection [23].

4.4.4 Essaims de particules

La méthode Particle swarm optimization (PSO) ou optimisation par essaim de particules est une méthode d'optimisation fondée sur une population stochastique de points initialement répartis sur un domaine de recherche. Cette méthode a été publiée la première fois par Kennedy et Eberhart en 1995 [33]. Elle génère un essaim de particules dont chaque membre est une solution éventuelle du problème d'optimisation. Cet essaim vole dans l'espace de recherche (à N dimensions) et chaque membre de celui-ci est attiré par sa meilleure solution et celle de ses voisins. Chaque particule est donc dotée d'une mémoire contenant les données relatives à son vol (position, vitesse et sa meilleur solution au problème) ainsi que de la faculté de communiquer (ou socialiser) avec son entourage.

A chaque itération, le résultat de la fonction objectif est calculé pour chaque membre de l'essaim permettant de déterminer alors le leader de celui-ci. Le leader étant la particule possédant la meilleure solution au problème de l'essaim.

Cette méthode est donc de type Direct Search et ne nécessite pas de connaitre les dérivées de la fonction à minimiser.

A chaque itération, et en prenant en compte l' inertial weight model , la position et la vitesse d'une particule est mise à jour par :

$$\begin{cases} V_{t+1} = \omega_t . V_t + C_1 . rand() . (pbest - x_t) + C_2 . rand() . (gbest - x_t) \\ x_{t+1} = V_{t+1} + x_t \end{cases} \quad (60)$$

où C_1 et C_2 sont des constantes d'apprentissage, $rand()$ le résultat d'un générateur de nombres aléatoires, $gbest$ est la position de la particule leader, $pbest$ est la meilleure position de la particule et ω_t l'inertie à l'instant t (Fig. 24).

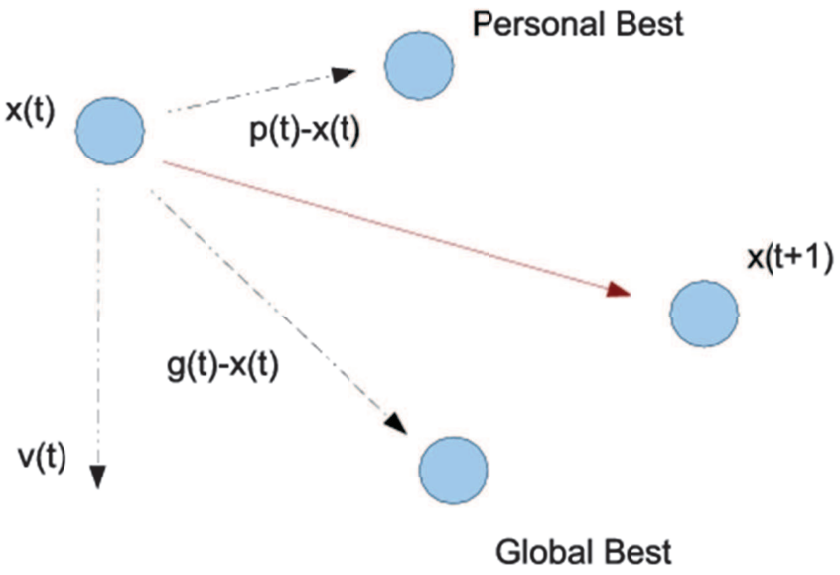

Figure 24 : Schéma de l'algorithme du PSO

La valeur de $gbest$ sera réévaluée à chaque itération afin de prendre en compte l'évolution possible du minimum global trouvé au cours de l'optimisation.

4.4.4.1 Focus sur les essaims de particules

Les essaims de particules ont pour particularité d'être l'un des algorithmes méta-heuristiques les plus simples en termes de complexité d'équations. Ainsi seule la mémorisation du $gbest$ et du $pbest$ sont nécessaires pour le calcul de l'itération suivante avec les deux équations précédentes. Cette particularité est intéressante dans le cadre d'une implantation dans un système à faibles ressources informatiques ou encore soumis à des contraintes de type temps réel. Cependant cet algorithme possède un inconvénient majeur : le nombre d'itérations nécessaires pour trouver un minimum potentiellement global. De nombreuses modifications ont déjà été réalisées pour cet algorithme. Pour en savoir plus, le lecteur est invité à se rendre sur Particle swarm Central [11] qui propose un regroupement de travaux, codes et bibliographies.

Deux modifications sont proposées dans cette thèse afin de réduire le nombre

d'évaluations de la fonction objectif ainsi que d'améliorer la précision des résultats. Les performances de ces modifications sont alors testées par rapport à une PSO classique dont les paramètres seront identiques.

Critère d'arrêt sur le rayon de l'essaim :

Pour un PSO classique, le critère d'arrêt est le nombre d'itérations maximal autorisé mais peut aussi être, dans le cadre d'une modification, la distance au minimum recherché. Dans le cadre d'une application sur un processus physique réel et non mathématique, la précision de la particule conduisant au meilleur résultat ne peut forcément être reproduite par les actionneurs du système. Ainsi il est possible d'imaginer un critère d'arrêt sur cette précision en mesurant le rayon de l'essaim. Cette mesure du rayon se fait grâce à une mesure de distance de toutes les particules de l'essaim par rapport à la particule leader à l'aide d'une norme 2. Si la distance maximale devient inférieure à un critère défini par l'utilisateur et cela pendant un certain nombre d'itérations, l'algorithme est alors stoppé (Fig. 27 & 25). Cette modification porte le nom de "PSO-Radius".

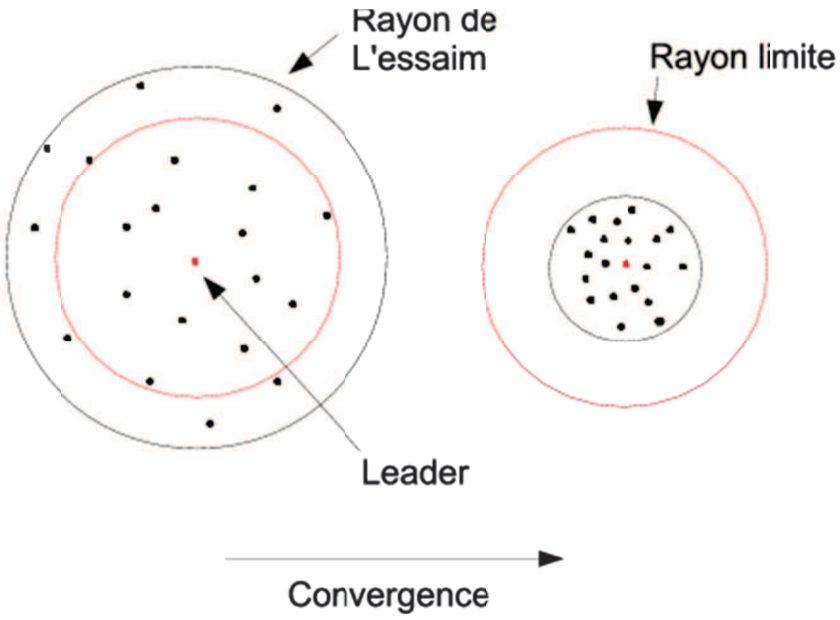

Figure 25 : Exemple d'arrêt sur le rayon de l'essaim

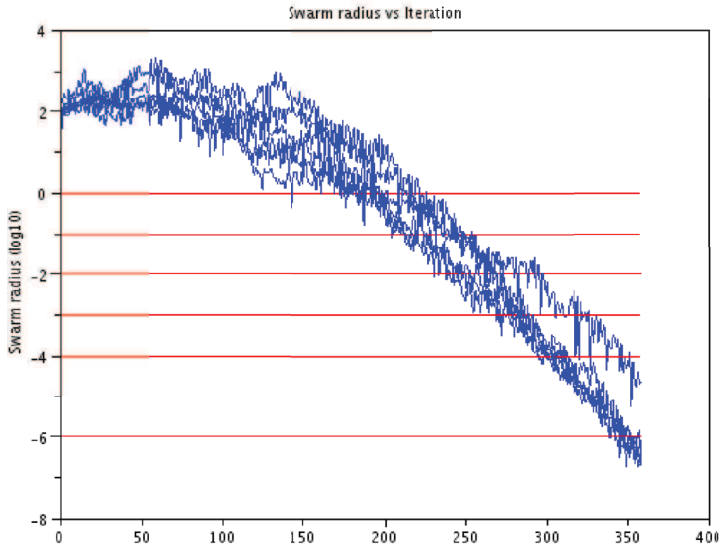

Figure 26 : Exemple d'arrêt sur le rayon de l'essaim, graphique de contrôle de la PSO Toolbox

Accélération de convergence par saut de l'essaim :

Cette modification est basée sur deux idées inspirées d'un film de science-fiction Battlestar Galactica ainsi que du jeu video Starcraft.

Dans le jeu vidéo est présent un porte-nef (Carrier) qui possède la possibilité de lancer sur une carte des engins de combat avec pour but la recherche et la destruction d'ennemis. Dans le cas présent ce seront des éclaireurs (Raptor) qui auront uniquement pour but la recherche d'une meilleure valeur de la fonction objectif. Le carrier est ici assimilé à la particule leader de l'essaim et les raptors sont lancés, suivant une probabilité, à une vitesse plus rapide que celui-ci dans des directions définies aléatoirement.

Dans le film de science-fiction, les vaisseaux, constitués en flotte, ont la possibilité d'effectuer des sauts instantanés dans l'espace (FTL jumps) suivant un vecteur de saut unique. L'utilisation de ce vecteur de saut permet alors de conserver la géométrie de la flotte et donc d'éventuelles collisions. Si l'on considère que le Carrier saute à l'emplacement du meilleur Raptor, cela permet de définir un vecteur de saut. Le reste de l'essaim saute donc dans l'espace suivant ce vecteur. La géométrie de l'essaim est alors conservée permettant de s'échapper d'un minimum local tout en conservant une bonne diversité de celui-ci.

Cette amélioration devrait permettre d'accélérer la convergence du processus vers un résultat éventuellement éloigné de la position de l'essaim mais aussi améliorer la qualité du résultat.

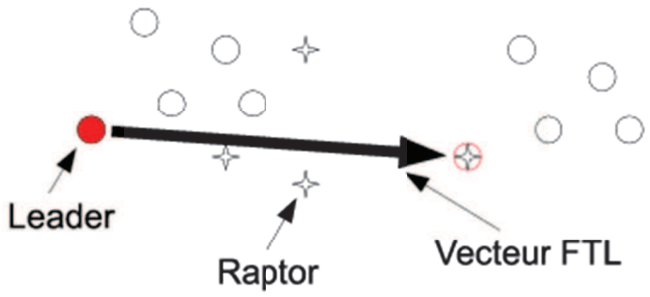

Figure 27 : Schéma de fonctionnement du saut d'essaim

Tests de validation des modifications :

Les tests de validation de ces améliorations ont été extraits de la revue bibliographique réalisée par Molga et Smutnicki [46] et sont destinés à l'évaluation des performances des algorithmes d'optimisation sur des problèmes multimodaux, multidimensionnels et avec la possibilité d'être en présence d'un grand nombre de minima locaux. Chaque test est répété 100 fois pour chaque programme d'optimisation.

Les différentes fonctions de tests ainsi que les résultats sont disponibles dans l'annexe A.

Conclusion sur les améliorations apportées :

Les deux modifications proposées n'apportent pas les mêmes réponses aux attentes :

- le Radius : pour les optimisations à long terme, c'est à dire quand le nombre d'itérations maximal est élevé, le Radius permet d'économiser un nombre important d'appels à la fonction objectif sans pour autant pénaliser la valeur finale.
- le BSG-Starcraft : telle que présentée cette modification n'apporte pas réellement d'amélioration à la performance de l'algorithme initial et entraine un surcout proportionnel à la probabilité de lancement des Raptors. Celle-ci est donc mise de côté pour la suite.

4.5 Robustesse des résultats

L'optimisation robuste est une classe d'optimisation où non seulement le résultat est important mais aussi sa capacité à rester stable suivant diverses incertitudes. Une définition de robuste est donnée par le Larousse comme étant :

Qui résiste bien aux causes d'agression, d'altération et à l'usage prolongé. [38]

Deux possibilités sont offertes pour déterminer la robustesse d'un résultat :

• réaliser une étude avec un modèle déterministe et réaliser une étude de sensibilité. Ceci est donc une méthode à posteriori;

• ou inclure directement les incertitudes dans le modèle et utiliser un procédé d'optimisation adapté.

Dans le cadre de cette thèse, la seconde option sera retenue afin de limiter les coûts de calculs supplémentaires et éventuellement la nécessité de renouveler le processus d'optimisation si l'analyse de sensibilité n'est pas assez concluante si le premier cas est retenu.

4.5.1 Robustesse à posteriori par analyse de sensibilité

4.5.1.1 Criblage

Les méthodes de criblage sont utilisées pour l'analyse de modèles comportant un nombre important de variables d'entrées. Ceux-ci sont relativement complexes à analyser mais il se peut que seules certaines entrées aient une influence importante. Ces méthodes ont surtout un but qualitatif par l'identification rapide des entrées importantes.

Parmi les méthodes existantes, il est possible de citer celle de Morris qui permet une interprétation visuelle des résultats afin de classer les entrées en trois catégories:

• à effets négligeables;

• à effets linéaires et sans interactions;

• à effets non-linéaires avec ou sans interactions.

La méthode de Morris [48] consiste à discrétiser un espace de travail de dimension p égal aux nombre d'inconnues d'entrées suivant une variable Δ. Un point est alors sélectionné et un calcul d'effet élémentaire est effectué autour de celui-ci pour chaque variable :

$$d_{X_i} = \frac{f(p_i) - f(p_{i-1})}{\Delta} \tag{61}$$

Ainsi la construction du vecteur d nécessite $p+1$ évaluations de la fonction objectif. Cette méthode est répétée R fois afin d'obtenir une collection de vecteurs d. Le nombre de recours à la fonction objectif est donc de $R(p+1)$. On détermine ensuite la moyenne m_i et l'écart-type σ_i des vecteurs d_{X_i} qui sont alors reportés sur un graphique $f(m_i, \sigma_i)$. Une valeur importante de moyenne indique une sensibilité du modèle à la variation de l'entrée alors qu'un écart-type va indiquer la présence d'interactions et-ou d'un effet non-linéaire de la variable car la méthode ne permet pas de distinguer les deux.

Cette méthode permet ainsi de sélectionner uniquement les entrées intéressantes pour effectuer des études plus approfondies.

4.5.1.2 Analyse locale

L'analyse locale permet d'obtenir une information sur l'effet d'une variation d'un paramètre autour d'un point de fonctionnement. Cette information s'obtient en calculant les

dérivées partielles $\frac{\partial f}{\partial X_i}$ que l'on peut normaliser afin de les comparer entre elles : $\frac{X_i}{f(X)}\frac{\partial f}{\partial X_i}$. L'usage classique veut que l'on étudie l'influence des paramètres pour des variations de l'ordre de 5 et 10%. Le classement des résultats permet alors seulement d'observer l'importance de certains paramètres autour d'un point de fonctionnement. Cette analyse est surtout intéressante dans le cadre d'un système linéaire.

Une prise en compte des dérivées secondes permet d'obtenir des informations sur les interactions entre paramètres et donc d'avoir une vue plus précise de la stabilité du système mais avec un cout de calcul plus important [8]. Comme précédemment, l'observation des variations des paramètres peut se faire pour des écarts de 5 et 10%.

4.5.1.3 Analyse globale

L'analyse globale se différencie de l'analyse locale par l'étude de l'influence des variations des paramètres d'entrées sur les variations des paramètres de sortie [62]. Une mesure simple de l'influence peut être déterminée par l'écriture d'un indice de sensibilité avec X_i une variable d'entrée et Y une variable de sortie :

$$SRC_i = \frac{V(X_i)}{V(Y)} \tag{62}$$

Pour la prise en compte des interactions et une mesure plus précise, les indices de Sobol [65] autorisent l'étude des interactions d'un indice 2 (interaction simple) jusqu'à l'indice $2^p - 1$. Ces indices s'écrivent sous la forme suivante :

$$\begin{cases} S_{1..p} = \frac{V_{1..p}}{V} \\ V_{1..p} = V - \sum_{i=1}^{p} V_i - \sum_{1\le i<j\le p} V_{ij} - \ldots - \sum_{1\le i_1<\ldots<i_{p-1}\le p} V_{i_1\ldots i_{p-1}} \end{cases} \tag{63}$$

L'interprétation des résultats est assez facile car les indices sont compris entre 0 et 1 et la somme totale est de 1. Plus un indice est proche de l'unité, plus la combinaison des variations sur ces entrées influence la sortie. Cependant dans le cas où le nombre de paramètres devient important, un autre indice peut être mis en place, l'indice de sensibilité totale qui est la somme des indices de sensibilité relatifs à la variable i :

$$S_{T_i} = \sum_{k\#i} S_k \tag{64}$$

4.5.2 Robustesse à priori par fonction objectif

Dantzig [16] en 1955 est l'un des premiers à traiter directement de la prise en charge d'incertitudes dans des problèmes linéaires. Les applications touchaient notamment le domaine de la planification. En 1973, Soyster [66] publie un article sur la résolution de systèmes linéaires

inexacts : les fonctions de contraintes sont alors soumises à des incertitudes. Ces 10 dernières années il est possible de citer les travaux de Ben-Tal, El Ghaoui, Lebret, Nemirovski et Bertsimas [?, 21, 4] sur le traitement de la robustesse et le coût associé, notamment dans le cadre de systèmes linéaires.

Cependant, dans une approche plus généraliste, les systèmes mécatroniques ne sont pas linéaires et soumis à des incertitudes aussi bien sur le modèle que, éventuellement, sur les contraintes.

En 1952, Markowitz [44] introduit une fonction objectif sous forme d'une combinaison d'espérance et de variance pour l'optimisation de gestion de portefeuille d'actions. L'espérance traduit alors le revenu attendu des actions et la variance la volatilité de celles-ci. Le but de l'optimisation est alors de maximiser le revenu tout en minimisant la volatilité. Cependant ce système est critiquable du fait de l'hypothèse que les différentes variables suivent une loi de Gauss. La moyenne et l'écart-type peuvent aussi être utilisés afin d'établir une fonction objectif, toujours sous l'hypothèse que celle-ci suit une loi de Gauss.

L'utilisation de la loi de Weibull [72] peut être une solution au problème lié à l'hypothèse de la présence d'une distribution gaussienne. En effet cette loi se compose de trois paramètres et permet d'approximer une très grande variété de distributions, gaussienne comprise. Cette loi se présente sous la forme suivante de densité de probabilité puis de fonction de répartition :

$$\begin{cases} f(x) = \frac{\beta}{\eta}\left(\frac{x-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{x-\gamma}{\eta}\right)^{\beta}}, \forall x \geq \gamma \; sinon \; f(x) = 0 \\ F(x) = 1 - e^{-\left(\frac{x-\gamma}{\eta}\right)^{\beta}}, \forall x \geq \gamma \; sinon \; F(x) = 0 \end{cases} \tag{65}$$

L'optimisation se déroule alors en étudiant l'écart par rapport à γ qui traduit une valeur cible puis en maximisant β le facteur de forme et η le facteur d'échelle.

L'introduction de données statistiques dans un processus d'optimisation nécessite le besoin d'effectuer un échantillonnage statistique. En effet il est quasi impossible d'effectuer un nombre très important (plus de 500) de calculs pour reconstruire une distribution statistique. L'échantillonnage permet alors de réduire de façon importante ceux-ci moyennant l'acceptation d'une marge d'erreur. Considérons une population (ou ensemble de produit fabriqués) relativement importants ($?\,100$), en appliquant la formule de Cochran [12] en posant les hypothèses d'un intervalle de confiance de 95% et en acceptant une erreur sur la moyenne et le coefficient de variation de respectivement 3 et 5%

$$n = \frac{T_{1-\alpha/2}^2 \, cv^2}{d^2} \tag{66}$$

Ici $\alpha = 0.05$, cv est le pourcentage sur le coefficient de variation et d le pourcentage sur la moyenne, ce qui conduit à un résultat arrondi de $n = 11$. Les valeurs présentées ici peuvent apparaître comme étant larges mais elles sont à adapter par l'utilisateur suivant le problème à traiter (cout de calcul et précision recherchée).

4.6 Données d'optimisation et données de validation

Une fois le modèle à traiter établi et le processus d'optimisation choisi, il est nécessaire de se poser la question des données nécessaires à l'obtention d'un résultat. Ces données sont appelées données d'optimisation. Afin d'obtenir un résultat le plus pertinent possible, il convient de répondre à la question suivante :

Les données introduites sont-elles nécessaires et suffisantes pour la détermination des paramètres du modèle ?

De plus dans le cadre d'une optimisation robuste, cette question doit être complétée par :

L'éventail de données disponibles est-il représentatif du fonctionnement du système ?

Il n'existe pas réellement de démarche pour répondre à ces questions si ce n'est la bonne connaissance du comportement du système réel et du modèle établi.

Les données de validation permettent de vérifier que les paramètres déterminés dans le cadre de l'optimisation sont utilisables pour la prédiction de comportement du système et cela pour une plage d'entrée. Cette vérification permet donc de s'assurer de la validité du modèle, mais aussi par un choix judicieux des données de validation, de déterminer la plage d'utilisation du modèle.

4.7 Conclusion

De façon générale un processus d'optimisation se déroule en plusieurs étapes:

- établissement d'une fonction objectif ;
- définitions des fonctions de contraintes ;
- choix de la méthode d'optimisation.

Le choix de la méthode d'optimisation peut être l'élément le plus critique car il est soumis à des contraintes de temps et / ou à l'adaptation de celle-ci par rapport au systèmes, contraintes et à la fonction objectif.

Les données d'optimisation ainsi que celle de validation sont aussi des éléments à ne pas négliger, car mis à part la qualité du modèle, ce sont ces données qui sont à l'origine des résultats et qui autoriseront la mise en production d'un modèle.

Dans le cadre du système mécatronique étudié, l'usage de méta-heuristique parait intéressant :

- le système étudié est non-linéaire
- les paramètres le constituant sont nombreux et interagissent.

De par sa simplicité d'implantation, on privilégiera, dans le chapitre suivant, l'usage du PSO et de son amélioration Radius afin de réaliser les identifications de paramètres ainsi que différentes optimisations.

5 Simulation

Sommaire

Ce chapitre traite de l'implantation et du choix d'un modèle numérique de l'actionneur ainsi que de son identification. La création du modèle numérique se fait en deux étapes :

- sélection des modèles analytiques possibles ;
- évaluation de la pertinence de ces modèles par rapport à l'expérience suite à une procédure d'identification.

Une fois un modèle sélectionné et identifié, le composant modélisé peut alors être inséré dans une modélisation plus large de l'ensemble du système mécatronique.

5.1 Construction du modèle

L'actionneur, ici de type piézoélectrique, ne peut être découplé de son pré-actionneur. Dans le cas présent, le pré-actionneur est un amplificateur linéaire de tension. Il sera considéré comme parfait, sauf si les résultats obtenus ne sont pas satisfaisants. La création du modèle de l'actionneur se fait en utilisant le langage Modelica qui permet une modélisation efficace et simple d'utilisation.

5.1.1 Modèle du pré-actionneur

Suivant l'hypothèse d'amplificateur linéaire parfait, le pré-actionneur se modélise de façon simple avec les composants électriques standards disponibles dans la bibliothèque de Modelica. On ajoutera cependant quelques contraintes sur les tensions d'entrées de l'amplificateur, ainsi qu'une résistance en série, afin de modéliser l'impédance de celui-ci.

Nom de la variable	Symbole
Tension d'entrée max	$V_{in\,max}$
Tension d'entrée min	$V_{in\,min}$
Résistance d'impédance	R
Coefficient d'amplification	G

Table 12 : Paramètres génériques de l'amplificateur

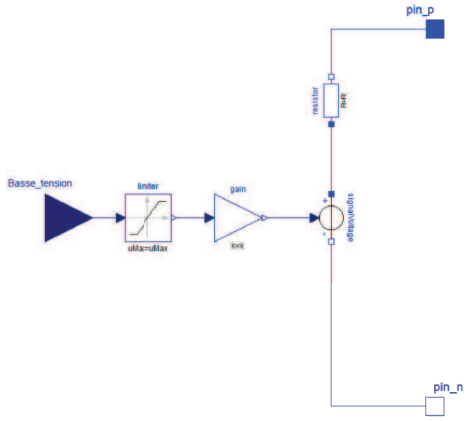

Figure 28 : Schéma Modelica de l'amplificateur linéaire

L'amplificateur est doté de deux connecteurs de type *electrical* pour en faire un élément connectable à un autre composant électrique. Le connecteur est en fait un bus qui permet la transmission bi-directionnelle de la tension et de l'intensité.

Les paramètres à identifier sont donc au nombre de trois. Les paramètres concernant les tensions d'entrée sont issus de la fiche technique du fabricant et facilement vérifiables par une simple expérience. L'impédance de l'amplificateur est plus difficile à maîtriser car dépendante de la fréquence de la forme d'onde à générer.

5.1.2 Modèle de l'actionneur

Dans un premier temps, on pose l'hypothèse que l'actionneur est linéaire. C'est à dire que l'on ne va pas chercher à prendre en compte l'effet d'hystérésis de celui-ci. Cet effet pourra être pris en compte si les résultats de l'identification ne sont pas jugés suffisants. Le modèle de l'actionneur est composé de trois sous-groupes : la masse d'inertie, le piézo amplifié et l'arbre en frottement.

5.1.2.1 Modèle du sous-groupe piézo amplifié

Le modèle de départ est le premier modèle présenté dans le chapitre 3, suivant le schéma et l'équation suivante :

$$\begin{bmatrix} F \\ V \end{bmatrix} = \begin{bmatrix} K_1 & -K_1 a_1 \\ -K_1 a_1 & K_1 a_1^2 + \frac{1}{C_1} \end{bmatrix} \cdot \begin{bmatrix} x \\ Q \end{bmatrix} \tag{67}$$

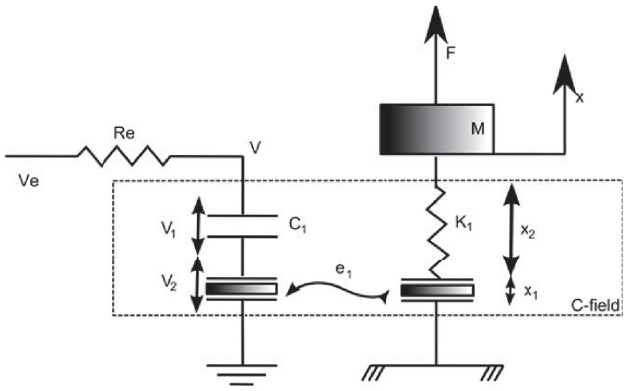

Figure 29 : Schéma de l'actionneur piézoélectrique pour le modèle 1

Nom de la variable	Valeur	Condition
a_1	x/Q	$F = 0$
C_1	Q/V	$F = 0$
K_1	$F/(a_1 Q)$	$x = 0$

Table 13 : Variables pour le modèle 1

L'équation est alors retransformée en langage Modelica par :

```
model piezo_amplifie

// Declaration des interfaces mecaniques et electriques
Modelica.Mechanics.Translational.Interfaces.Flange_a    flange_a ;
Modelica.Mechanics.Translational.Interfaces.Flange_b    flange_b ;
Modelica.Electrical.Analog.Interfaces.PositivePin    pin_p ;
Modelica.Electrical.Analog.Interfaces.NegativePin    pin_n ;

// Declaration des parametres fixes du modele
parameter Real K1=4.9e5; // K1=F/(a1*Q)
parameter Real a=0.3; // a1=x/Q
parameter Real C=0.25e-6; // C1=Q/V

// Declarations des variables du modele
Real Q(start=0);
Real F;
Real v;
Real x(start=0);
Real i;
```

```
        // Debut du modele mathematique
Equation

        // Traduction de la matrice piezoelectrique generalisee
        // attention au signe de x et F pour cause de changement de point de vue par rapport au
        modele analytique
F=K*(-1*x)-K*a*Q
v=-K*a*(-1*x)+( K*a^2+1/C)*Q;
i=der(Q);

        // Equations des connecteurs mecaniques
-1*x=flange_b.s-flange_a.s;
flange_a.f=-F;
flange_b.f=F;

        // Equations des connecteurs electriques
v=pin_p.v-pin_n.v;
pin_p.i=i;
pin_p.i+pin_n.i=0;

        // Fin du modele
end piezo_sans_masse;
```

Cette écriture, sous forme de code, permet de générer un composant doté de 4 bornes : deux mécaniques et deux électriques. Le modèle du piézo amplifié est alors un composant d'interaction électromécanique.

Une variante à ce modèle est aussi créée : on ajoute un terme mécanique d'amortissement dans les équations. En effet, l'armature et le matériau piezo ne sont pas parfaitement rigides, et des pertes visqueuses peuvent exister. Cette ajout se traduit par la modification d'équation suivante :

$$\begin{bmatrix} F \\ V \end{bmatrix} = \begin{bmatrix} K_1 & Am & -K_1 a_1 \\ -K_1 a_1 & 0 & K_1 a_1^2 + \frac{1}{C_1} \end{bmatrix} . \begin{bmatrix} x \\ \dot{x} \\ Q \end{bmatrix} \quad (68)$$

Cette modification sera aussi comparée à une expérience.

5.1.2.2 Modèle de la masse d'inertie

La masse d'inertie est considérée comme une masse ponctuelle qui n'est soumise à aucun type de frottement. Elle suit donc l'équation suivante :

$$\sum \vec{F}_{ext} = m.\vec{a} \quad (69)$$

La traduction par Modelica de cette équation est le composant Mass avec le code suivant :

```
model Mass "Sliding mass with inertia"
parameter SI.Mass m(min=0, start=1) ;
parameter StateSelect stateSelect=StateSelect.default ;
extends Translational.Interfaces.PartialRigid(L=0,s(start=0,stateSelect=stateSelect));
SI.Velocity v(start=0, stateSelect=stateSelect) ;
SI.Acceleration a(start=0) ;

Equation
v = der(s);
a = der(v);
m*a = flange_a.f + flange_b.f;
end Mass;
```

5.1.2.3 Modèle de l'arbre en frottement

L'arbre en frottement est plus délicat à modéliser. En effet, il s'agit d'une masse ponctuelle, comme la masse d'inertie, à laquelle on ajoute une force de frottement.

Différents modèles ont été établis dans le chapitre 3, qui sont séparables en deux catégories :

- les modèles statiques tels que Coulomb, augmentés des effets de type fluidiques ou bien Stribeck;
- les modèles dynamiques, comme Bliman et Sorinne, Dahl, LuGre, ...

Pour les modèles statiques, Modelica fournit directement une implémentation avec MassWithStopAndFriction qui est entièrement configurable.

Pour les modèles dynamiques, il est alors nécessaire de créer un composant personnalisé pour les différents modèles de frottement.

Par exemple, pour un modèle de Dahl suivant l'équation :

$$\frac{dF_r}{dt} = \sigma_0 v \left(1 - \frac{F_r}{F_c}.sign(v)\right)^{\alpha} \quad (70)$$

Le code associé est alors :

```
model Mass_dahl "Sliding mass with inertia"
parameter Modelica.SIunits.Mass m(min=0, start=1) ;
parameter StateSelect stateSelect=StateSelect.default;
extends.Modelica.Mechanics.Translational.Interfaces.PartialRigid(L=0,s(start=0,
stateSelect=stateSelect));
Modelica.SIunits.Velocity v(start=0, stateSelect=stateSelect) ;
Modelica.SIunits.Acceleration a(start=0) ;
parameter Real fc(start=1);
parameter Real sigma0(start=1);
parameter Real n(start=1);
Real f;
Real df;

Equation
df=der(f);
df= v*sigma0*(1-f*sign(v)/fc)^n;
v = der(s);
a = der(v);
m*a = flange_a.f + flange_b.f - f;
end Mass_dahl;
```

Dans cette application, seront testés les modèles statiques ainsi que les modèles de Dahl et LuGre pour les dynamiques.

5.2 Applications

Afin de réaliser les différentes applications, un banc de test est créé. Il comporte les éléments suivants :

- un actionneur slip & stick inertiel (SPA35XS, Cedrat)
- un amplificateur de tension (CA45, Cedrat)
- un générateur de forme d'onde programmable qui est aussi un système de commande / contrôle (ADWIN GOLD II, Adwin)
- un capteur d'intensité (CT-0.1P, LEM)
- un pont diviseur de tension à haute impédance
- un capteur laser de déplacement micrométrique (LC-2430 & LC-2450, Keyence)
- un bornier d'acquisition (USB 1616 HS BNC, Measurement Computing)

5.2.1 Identification inverse

5.2.1.1 Identification des paramètres du sous-modèle piezo amplifié

Dans le modèle présenté, le composant piézoélectrique est monobloc et sa masse est reportée vers les bornes du composant. Dans notre cas, la masse du piézo est remise au centre du composant, ce qui permet d'insérer un nouveau mode de vibration dans le modèle. Ce choix de modélisation nécessite alors de séparer en deux parties identiques le modèle piézoélectrique. Cela modifie les valeurs des paramètres de raideurs et de capacité par une mise, respectivement, en série et parallèle.

Pour cette identification, l'actionneur est démonté et l'arbre mis en encastrement avec le bâti. Le capteur de déplacement laser est alors pointé sur la masse d'inertie.

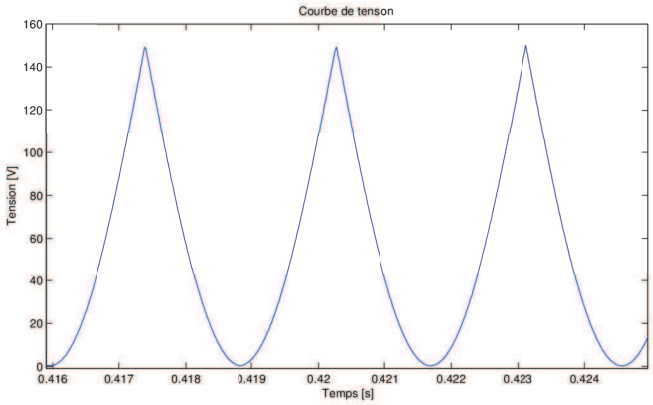

Figure 30 : Tension appliquée à l'actionneur

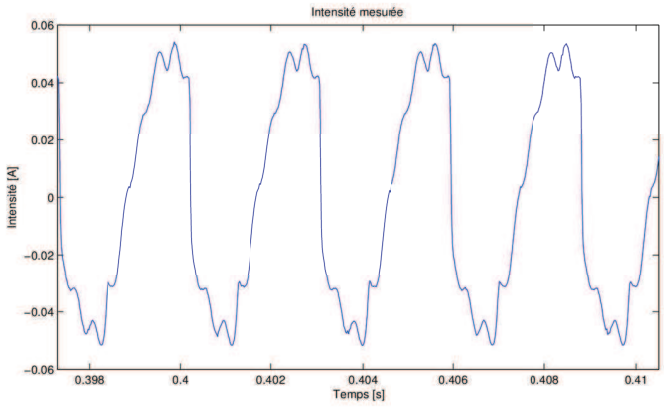

Figure 31 : Intensité mesurée

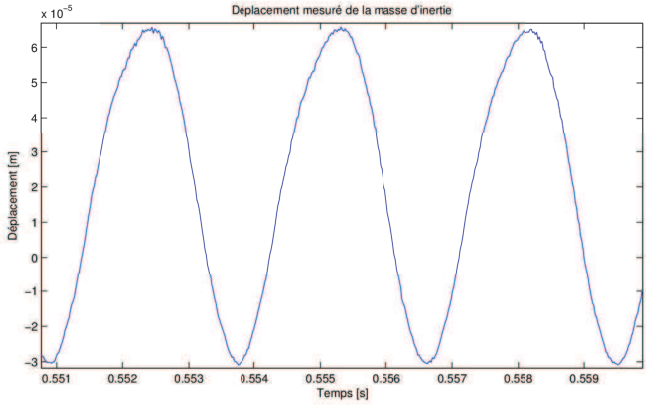

Figure 32 : Déplacement mesuré de la masse d'inertie

La masse d'inertie est de $40gr$ suite à une mesure avec une balance. De même, la capacité de l'élément piézoélectrique est déterminée par un multimètre à $250nF$.

Le critère de performance f d'un modèle est évalué par sa capacité à reproduire le déplacement x de la masse d'inertie et la courbe d'intensité i de façon unitaire. Ce qui revient à écrire les équations suivantes:

$$\begin{cases} f_i = \frac{\|i_{mesure} - i_{simulation}\|}{\|i_{mesure}\|} \\ f_x = \frac{\|x_{mesure} - x_{simulation}\|}{\|x_{mesure}\|} \\ f = f_i + f_x \end{cases} \quad (71)$$

Le premier modèle est testé en utilisant les valeurs du constructeur ainsi que la table 14 des paramètres du modèle. Il reste donc uniquement à identifier la valeur de la résistance d'impédance de l'amplificateur.

Nom de la variable	Équation	Valeur
a_1	x/Q	$1.46m/C$
C_1	Q/V	$250nF$
K_1	$F/(a_1 Q)$	$4.9e^5 N/m$

Table 14 : Valeurs des variables suivant modèle 1

L'identification de cet unique paramètre peut se faire directement par un mapping sur une plage de valeur. Cependant, on préférera faire une recherche par l'utilisation d'un PSO en une dimension (Table 15).

Nom	Valeur
Inertie [initiale,finale]	[0.9,0.4]
Itération max.	20
Facteur de connaissance [global,personnel]	[2,2]
Nombre de particules	20
Rayon limite	$1e^{-3}$
Nombre d'itération sous Rayon	10
Plage de lancer de R [max,min]	[1000,0]
Vitesse [max,min]	± 0.5x plage de lancer

Table 15 : Paramètres du PSO

Les résultats obtenus, présentés en Table 16, ne sont pas satisfaisants par un écart relativement important sur l'erreur de déplacement, et encore plus important sur l'intensité.

Nom	Valeur
Résistance	517Ω
Erreur sur l'intensité	26.5%
Erreur sur le déplacement	13.3 %
Erreur globale	39.8 %

Table 16 : Résultats pour la recherche d'une valeur de R

Afin d'obtenir de meilleurs résultats, on rend variable les valeurs de résistance, de raideur de l'actionneur, ainsi que le coefficient de couplage du piézo. Une nouvelle identification est réalisée par un PSO dont les paramètres sont définis en Table 17.

Nom	Valeur
Inertie [initiale,finale]	[0.9,0.4]
Itération max.	20
Facteur de connaissance [global,personnel]	[2,2]
Nombre de particules	20
Rayon limite	1e-3
Nombre d'itération sous Rayon	10
Plage de lancer R [max,min]	[1000,0]
Plage de lancer a [max,min]	[2,0]
Plage de lancer K [max,min]	[4.9e5 x 1.2,4.9e5 x 0.8]
Vitesse X [max,min]	± 0.5x plage de lancer X

Table 17 : Paramètres du PSO

Les résultats obtenus, en Table 18, sont meilleurs que précédemment. On note tout de même une forte erreur sur l'intensité par une sous-estimation dans le modèle par rapport à la mesure.

Nom	Valeur
Résistance	622Ω
Coefficient de couplage	1.44
Raideur	429541
Erreur sur l'intensité	30.25 %
Erreur sur le déplacement	7.21 %
Erreur globale	37.46%

Table 18 : Résultats pour la recherche pour 3 paramètres

Les écarts étant encore relativement importants, on teste l'ajout d'un terme d'amortissement de la partie mécanique du système. Une identification est alors relancée (Table 19).

Nom	Valeur
Inertie [initiale,finale]	[0.9,0.4]
Itération max.	20
Facteur de connaissance [global,personnel]	[2,2]
Nombre de particules	20
Rayon limite	1e-3
Nombre d'itération sous Rayon	10
Plage de lancer R [max,min]	[1000,0]
Plage de lancer a [max,min]	[2,0]
Plage de lancer K [max,min]	[4.9e5 x 1.2, 4.9e5 x 0.8]
Plage de lancer Am [max,min]	[0,100]
Vitesse X [max,min]	± 0.5x plage de lancer X

Table 19 : Paramètres du PSO

Nom	Valeur
Résistance	217Ω
Coefficient de couplage	1.4
Raideur	414360
Amortissement	44.78
Erreur sur l'intensité	22.13 %
Erreur sur le déplacement	4.89 %
Erreur globale	27.02%

Table 20 : Résultats pour la recherche pour 4 paramètres

Cet ajout d'un amortisseur est intéressant car il permet d'améliorer à la fois la précision du modèle en déplacement en se plaçant sous la barre des 5% et aussi la précision sur l'intensité (Table 20).

On retiendra alors cette modification comme modèle pour le sous-système masse d'inertie et piézo amplifié. Cependant on retiendra qu'il existe une erreur de l'ordre de 20% qui sous-estime l'intensité traversant réellement le composant d'interaction.

5.2.1.2 Identification des paramètres du sous-modèle arbre

Cette identification peut se faire à partir de mesures différentes :

- soit en fixant au bâti la masse d'inertie et en effectuant une mesure de déplacement de l'arbre;
- soit en mesurant le déplacement de l'arbre avec la masse d'inertie mobile;
- ou bien en effectuant la mesure du déplacement de la masse d'inertie.

La première solution, qui consiste à d'effectuer une mesure en bloquant la masse d'inertie peut apparaitre comme étant à privilégier car elle permet d'éliminer du modèle l'aspect dynamique de la masse. Cependant cette mesure ne peut se faire que sur une petite portion de l'arbre frottant (de l'ordre de $50\mu m$) alors que la course totale de celui-ci peut aller jusqu'à $4mm$. Sur cette distance, il est possible que le frottement varie du fait, par exemple, de défauts de forme en fabrication. La Figure 33 montre un déplacement de l'arbre. On remarque deux modes de frottement différents par la rupture de pente moyenne indiquée par les deux lignes.

Les solutions 2 et 3 sont à retenir. Sachant que la partie masse d'inertie et piézo amplifié ont été identifiés précédemment, la seconde solution est choisie.

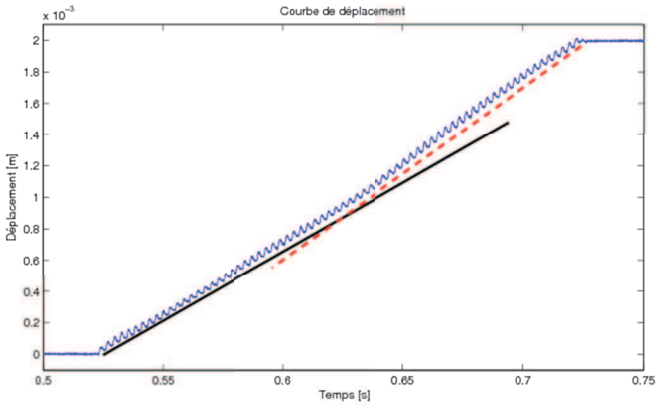

Figure 33 : Courbe de déplacement mesurée

Compte-tenu de la modification du mode de friction, on étudiera seulement la première partie de la courbe de déplacement.

La composante électrique ayant aussi été déterminée par l'identification précédente, la fonction objectif est alors uniquement l'écart entre courbe théorique et courbe expérimentale de déplacement.

Pour tous les modèles d'arbre, la masse de celui-ci est considérée comme ponctuelle et à pour valeur $4gr$. Pour les modèles de frottements dits statiques, on utilise le composant Modelica MassWithStopAndFriction , et pour les frottements dynamiques sont créés des composants spéciaux.

Identification des paramètres du frottement statique

Pour les modèles de type statique, sont testées les combinaisons suivantes :

- Coulomb
- Coulomb + Stribeck
- Coulomb + frottement fluide
- Coulomb + frottement fluide + Stribeck

Grâce à une simple expérience, on peut s'attendre à un effort de frottement maximal de $10N$ pour faire bouger l'arbre de l'actionneur. Cette limitation permet de ne pas utiliser un PSO, pour l'identification du modèle simple de Coulomb : un simple mapping de 0 à 10N par pas de 0.1 est suffisant (Fig. 34). Ce mapping permet de déterminer une force de frottement de $6.7N$ pour un écart de 2%.

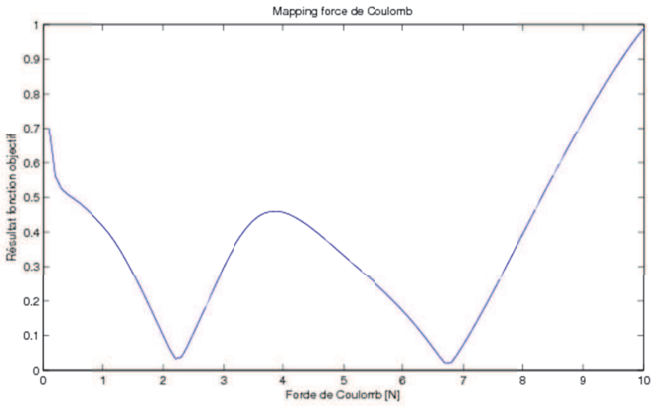

Figure 34 : Résultat du mapping de la force de Coulomb

Pour les autres modèles, on utilisera un PSO notamment à cause des interactions entre les paramètres (notamment pour Stribeck). Les paramètres du PSO sont communs et définis dans la Table 21.

Nom	Valeur
Inertie [initiale,finale]	[0.9, 0.4]
Itération max.	20
Facteur de connaissance [global,personnel]	[2,2]
Nombre de particules	20
Rayon limite	1e-3
Nombre d'itération sous Rayon	10
Plage de lancer F_c [max,min]	[10,0[
Plage de lancer f_v [max,min]	[100,0]
Plage de lancer F_s [max,min]	[10,0]
Plage de lancer v_s [max,min]	[100,0]
Vitesse X [max,min]	± 0.5x plage de lancer X

Table 21 : Paramètres du PSO pour l'identification des paramètres statiques

Les résultats sont inscrits dans la Table 22. On note que le frottement de Coulomb est le paramètre le plus important suivi par la force de frottement fluide. Le modèle de Stribeck n'apporte, dans ce cas, qu'une contribution faible aux résultats.

Modèle	F_c	f_v	F_s	v_s	Ecart
	$[N]$	$[N/m.s]$	$[N]$	$[m.s^{-1}]$	$[\%]$
Coulomb	6.7				1.97
Coulomb + frottement fluide	5.73	15.44			1.33
Coulomb + Stribeck	6.75		$1.8e^{-4}$	$2.5e^{-5}$	1.52
Coulomb + frottement fluide + Stribeck	5.43	17.29	0.2	0.67	1.32

Table 22 : Résultats pour les paramètres des modèles statiques

Identification des paramètres du frottement dynamique

Deux modèles dynamiques sont testés :

- Dahl
- LuGre

Le modèle de Dahl comporte 3 paramètres, et celui de LuGre en comporte 6. Ces modèles sont constitués d'équations comportant des interactions fortes entre les paramètres qui nécessitent, encore une fois, l'usage d'un PSO.

Nom	Valeur
Inertie [initiale,finale]	[0.9, 0.4]
Itération max.	20
Facteur de connaissance [global,personnel]	[2,2]
Nombre de particules	20
Rayon limite	1e-3
Nombre d'itération sous Rayon	10
Plage de lancer F_c [max,min]	[10,0[
Plage de lancer f_v [max,min]	[100,0]
Plage de lancer F_s [max,min]	[10,0]
Plage de lancer v_s [max,min]	[100,0]
Vitesse X [max,min]	± 0.5x plage de lancer X

Table 23 : Paramètres du PSO pour l'identification des paramètres dynamiques

Les résultats sont inscrits dans la Table 24. On retrouve des performances comparables pour les deux modèles, avec un léger avantage pour le modèle de LuGre.

Modèle	F_c	σ_0	σ_1	σ_2	n	v_s	F_s	Écart
	$[N]$	$[N/m]$	$[N/m.s]$	$[N/m.s]$		$[m.s^{-1}]$	$[N]$	$[\%]$
Dahl	8.59	2503162			1.75			0.86 %
LuGre	5.61	1216509	148.44	3.27		0.47	1.33	0.85 %

Table 24 : Résultats pour les paramètres des modèles dynamiques

Conclusion sur l'identification et robustesse du modèle de frottement retenu

On peut constater que les résultats d'identification des modèles de frottement sont comparables. Afin de les différencier, il est nécessaire d'en faire une comparaison qualitative (Fig. 35).

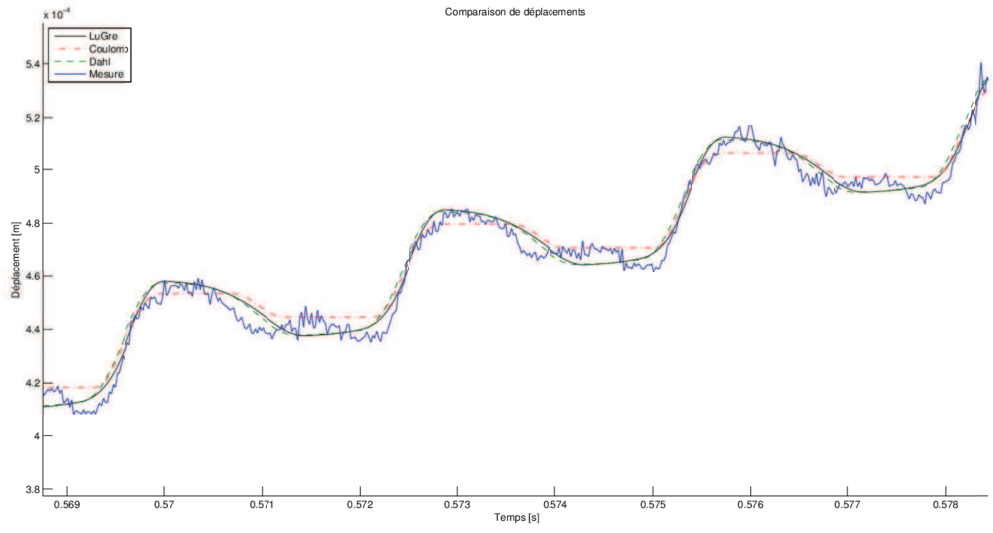

Figure 35 : Comparaison de courbe de déplacements

Les modèles statiques donnent des courbes que l'on peut qualifier de carrées comparées à la courbe expérimentale. Les modèles dynamiques ne souffrent pas de ce défaut.

Le modèle de LuGre apparait comme étant le modèle le plus performant. Cependant,

cette performance peut être mise en relation avec un nombre important de paramètres. Le modèle de Dahl permet d'obtenir un résultat comparable avec seulement 3 paramètres. Ce modèle est retenu pour effectuer une analyse de robustesse.

On effectue alors une analyse de robustesse locale, c'est à dire en effectuant des variations de ± 5 et $\pm 10\%$ pour chaque paramètre (Table 25).

F_c	-10%	-5%	$+5\%$	$+10\%$
Résultat [%]	9.92	5.31	5.81	11.9
σ_0	-10%	-5%	$+5\%$	$+10\%$
Résultat [%]	1.16	0.96	0.98	1.31
n	-10%	-5%	$+5\%$	$+10\%$
Résultat [%]	1.53	1.07	1.06	1.48

Table 25 : Analyse de robustesse locale du modèle de Dahl

A partir de cette analyse, il est possible de voir que la valeur du paramètre de la force de Coulomb possède une forte importance sur le résultat, comparé aux autres variables. C'est donc un paramètre critique pour le modèle.

5.2.2 Optimisation du signal de commande :

Dans la partie précédente, pour l'identification, a été utilisé un signal de commande standard fourni par le constructeur. Ce signal est basé sur l'utilisation de quart de sinus. D'autres types de signaux sont utilisés pour la commande de systèmes piézoélectriques :

- Jiang, dans [90], utilise des signaux triangulaires et trapézoïdaux pour la commande d'un actionneur inertiel slip & stick. Il propose aussi de remplacer la rampe de montée en tension par une courbe parabolique. Il conclue que la montée parabolique entraine dans tous les cas une amélioration des performances de l'actionneur.
- Yang, dans [86], propose une étude complète de l'usage de signaux triangulaires sur des actionneurs de classe identique à celui utilisé. Il y montre l'importance de la fréquence et du rapport cyclique sur le comportement de l'actionneur.
- Ha, dans [27], utilise des signaux triangulaires puis remplace les rampes de montée et de descente par des courbes paraboliques. Comme pour Jiang, il conclut que ces signaux sont plus performants que le signal triangulaire.

Ces différents signaux vont être étudiés dans le cadre de différentes optimisations et comparés au signal de référence du constructeur. On introduit aussi un nouveau type de signal : les signaux libres. La pertinence de ces signaux pourra être comparée par rapport aux autres, mais aussi par rapport à la vérification expérimentale.

5.2.2.1 Introduction au concept de signal libre

L'idée maîtresse de ce nouveau signal est que celui-ci soit non pas imposé dans sa forme par l'utilisateur ou le fabricant mais que ce soit l'actionneur qui décide de sa forme ainsi que de sa fréquence.

Afin de réaliser ce signal, la forme d'onde est réalisée par l'interpolation par spline d'un certain nombre de points (Fig. 36). Ce nombre de points pouvant soit être déterminé par l'actionneur dans le cadre du processus d'optimisation, soit imposé par l'utilisateur.

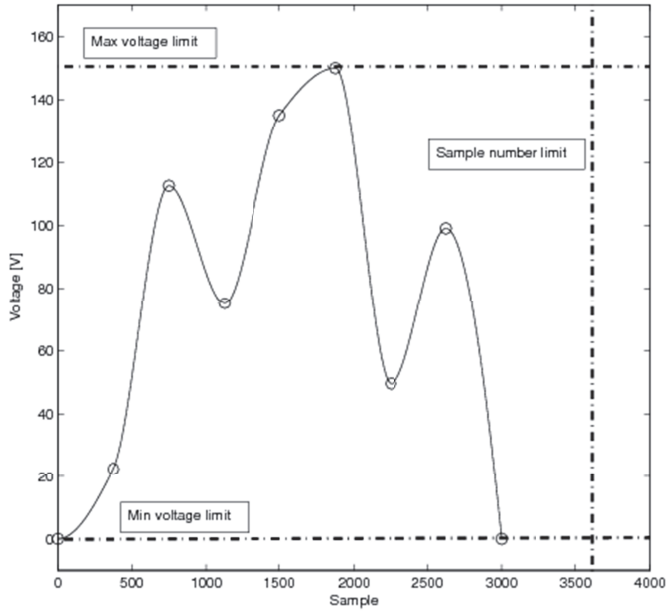

Figure 36 : Exemple de signal libre

5.2.2.2 Optimisation en vitesse

Cette optimisation a pour but d'améliorer la vitesse de déplacement de l'actionneur sur une distance de $2mm$. Pour cette optimisation, la tension maximale de la forme d'onde finale est fixée à $150V$. On imposera aussi un nombre de 5 points, équirépartis sur le temps de la période, pour la description de la spline. Pour toutes les formes d'ondes, la période doit commencer et finir à tension nulle.

Signal	Nombre de paramètres	Détail des paramètres
Carré	2	rapport cyclique + fréquence
Triangle	2	temps de montée + fréquence
Spline	6	tension des points + fréquence

Table 26 : Table des paramètres des signaux

Signal	Rapport cyclique	Fréquence	Vitesse simulée	Vitesse réelle	Écart
	[%]	[Hz]	[m/s]	[m/s]	[%]
Carré	33	381	1.3e-2	$\begin{cases} m = 1.87e^{-2} \\ \sigma = 2.26e^{-4} \end{cases}$	44
Dents de scie		508	1.89e-2	$\begin{cases} m = 6.8e^{-3} \\ \sigma = 9.36e^{-5} \end{cases}$	64
Triangle	87	521	2.04e-2	$\begin{cases} m = 1.14e^{-2} \\ \sigma = 1.08e^{-4} \end{cases}$	44
Spline		941	3.12e-2	X	X
Constructeur		400		$\begin{cases} m = 1.04e^{-2} \\ \sigma = 1.49e^{-4} \end{cases}$	-

Table 27 : Table des résultats pour l'optimisation en vitesse

Point n°	1	2	3	4	5
Tension unitaire	0	0.6	0.84	1	1

Table 28 : Table de résultat de la spline pour l'optimisation en vitesse

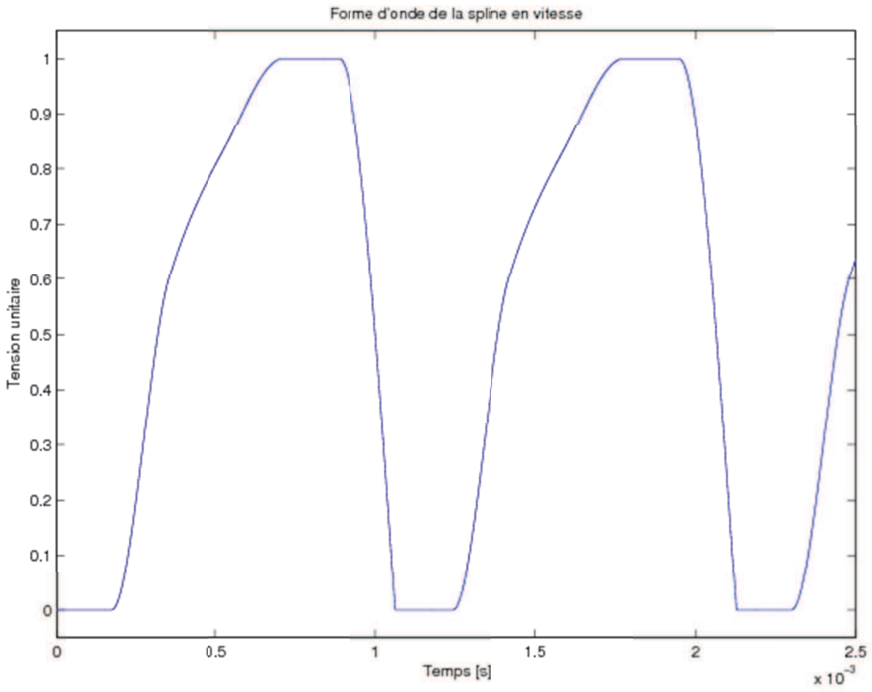

Figure 37 : Forme d'onde de la spline pour l'optimisation en vitesse

D'un point de vue numérique, cette optimisation des signaux donne des résultats tout à fait intéressants en permettant à l'actionneur de se déplacer plus rapidement qu'avec le signal de référence du constructeur. On note que la spline (Fig. 37) permet d'obtenir le meilleur résultat de tous les signaux testés.

Cependant, la confrontation à l'expérience ne donne pas forcément les résultats attendus : la spline n'est pas capable de déplacer l'actionneur dans la direction voulue (indiqué par un X dans la table de résultat), et pour les autres signaux les écarts sont très importants entre simulation et réalité.

5.2.2.3 Optimisation énergétique

L'optimisation énergétique de l'actionneur peut être un point important de conception, surtout si l'on s'adresse à des produits de type embarqué ou à faible consommation énergétique. Pour réaliser cette optimisation, on insère dans le modèle une sonde de tension ainsi qu'une sonde de courant. A l'aide du produit de ces deux grandeurs, on obtient la puissance instantanée que l'on intégrera afin d'obtenir l'énergie nécessaire à un déplacement de l'actionneur.

On définit la fonction objectif comme étant la minimisation de la quantité d'énergie requise sur le déplacement effectué. La fonction objectif a pour unité des joules par mètre $[J/m]$. Une fonction de contrainte est insérée : la vitesse minimale de déplacement doit être de 10^{-3} m/s.

Les signaux de l'optimisation précédente sont de nouveaux utilisés.

Signal	Rapport cyclique	Fréquence	Tension	Énergie simulée	Énergie réelle	Écart
	[%]	[Hz]	[V]	[J/m]	[J/m]	[%]
Carré	73	519	28	93.71	X	X
Dents de scie		458	27	44.46	X	X
Triangle	78	481	81	23.11	X	X
Spline		363		24.34	X	X
Constructeur		400	150		$\begin{cases} m = 60.9 \\ \sigma = 6.9e^{-1} \end{cases}$	-

Table 29 : Table des résultats pour l'optimisation énergétique

Point n°	1	2	3	4	5
Tension unitaire	0.25	0.78	0.4	0.47	0.49

Table 30 : Table de résultat de la spline pour l'optimisation énergétique

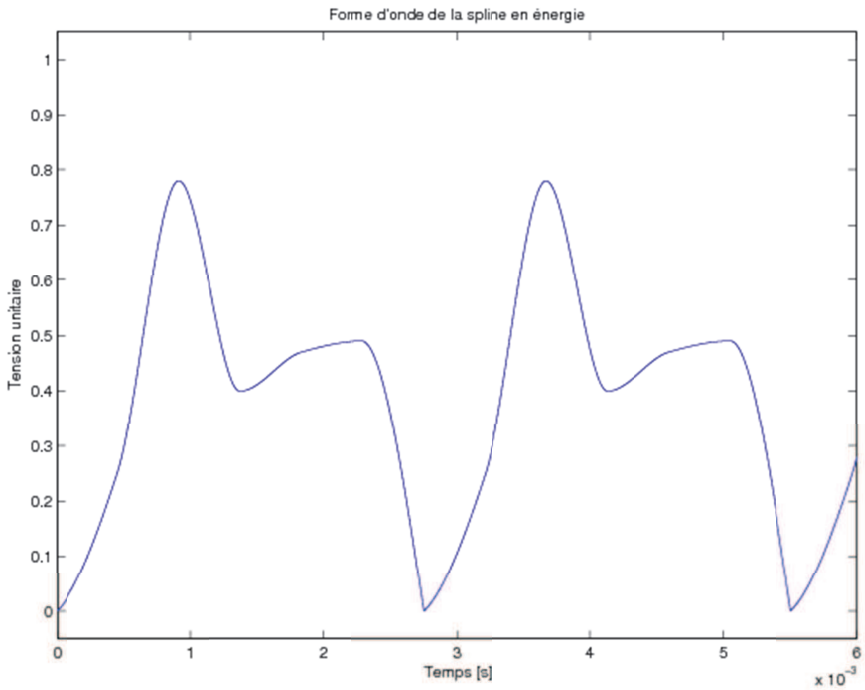

Figure 38 : Forme d'onde de la spline pour l'optimisation énergétique

Deux signaux ressortent de cette optimisation : le triangle, ainsi que la spline (Fig. 38). Mais, comme pour l'optimisation précédente, la vérification des résultats par l'expérience conduit, ici, à un échec complet. L'explication de cet échec provient soit d'une fréquence de signal non adaptée, soit d'un niveau de tension trop faible pour le mouvement réel de l'actionneur.

5.3 Analyse fréquentielle des signaux

Dans les différents cas d'optimisation, on peut constater que les signaux classiques ont une fréquence proche de $500Hz$. Pour le signal libre, il n'est pas évident de conclure aussi rapidement. On peut constater que les fréquences de cette onde sont relativement éloignées de ces $500Hz$. Une analyse par FFT permet cependant d'accéder aux fréquences contenues dans celui-ci.

Dans le cas de l'optimisation en vitesse (Fig. 39), on note une fondamentale à $940Hz$ ainsi que son harmonique à $1880Hz$.

Pour l'optimisation énergétique (Fig. 40), la fondamentale est à $360Hz$ et son harmonique à $720Hz$

On réalise alors une analyse fréquentielle de l'actionneur par l'injection d'un sinus d'amplitude de $150V$.

Dans le premier cas, on relève l'amplitude de la vitesse de déplacement de l'arbre et on note un pic de réponse (Fig. 41) à $1640Hz$ avec des limites à $-3dB$ de 1040 et $2210Hz$. La spline contient donc des fréquences proches ou comprises dans cette bande.

Dans le second cas, on relève l'énergie électrique consommée par unité de distance parcourue par l'arbre. On obtient alors une réponse (Fig. 42) comportant un minimum de consommation pour une fréquence de $720Hz$ avec des limites à $-3dB$ de 300 et $840Hz$. On note aussi la présence d'un minimum local à $320Hz$. Ce sont bien les deux fréquences présentes dans la spline optimisée en énergie.

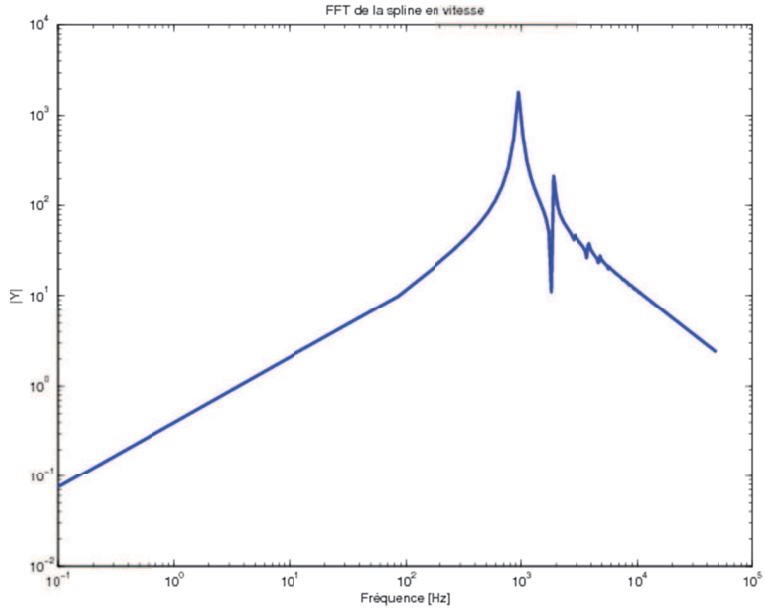

Figure 39 : FFT de la spline en vitesse

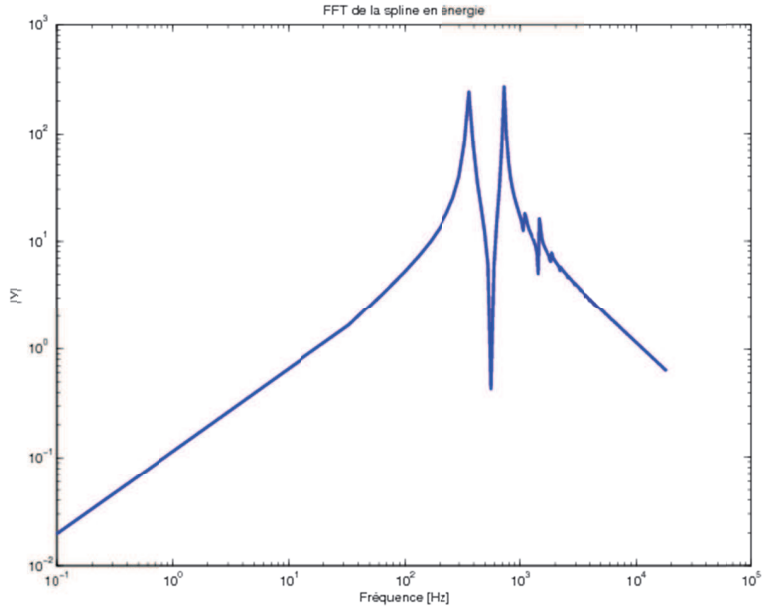

Figure 40 : FFT de la spline en énergie

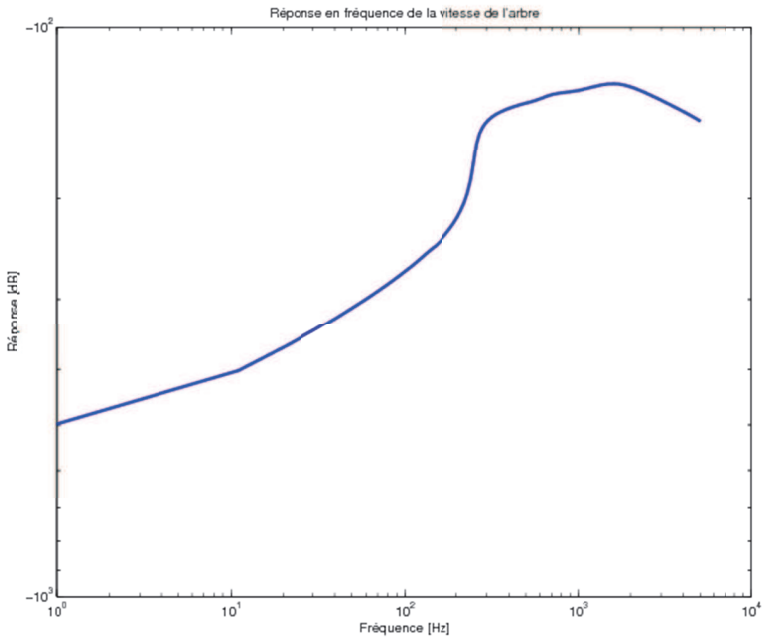

Figure 41 : Réponse en fréquence de la vitesse de l'arbre

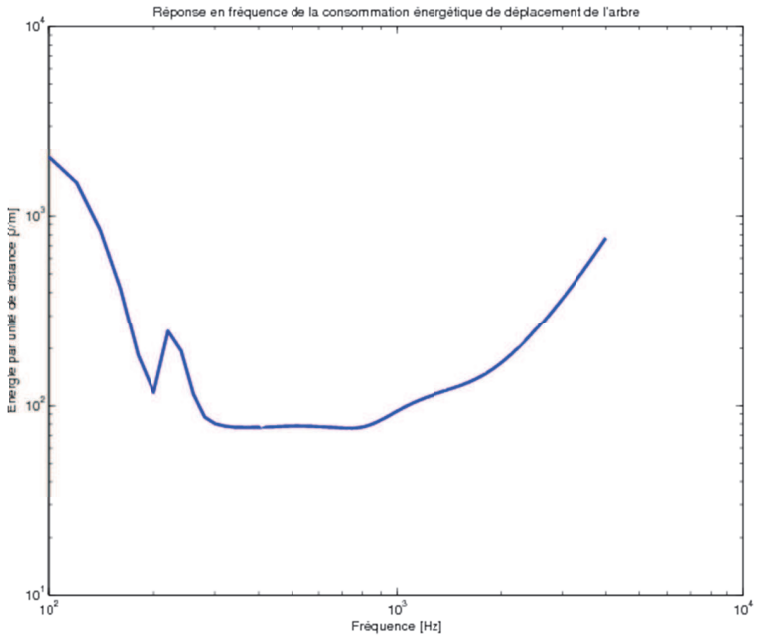

Figure 42 : Réponse en fréquence de l'énergie par unité de distance pour la mise en mouvement de l'arbre

5.4 Conclusion

Ce chapitre a permis de mettre en œuvre un algorithme d'optimisation méta-heuristique pour deux applications principales de la conception mécatronique : l'identification inverse de paramètres pour déterminer les paramètres d'un modèle et sa pertinence, et l'optimisation de paramètres en vue de l'amélioration des performances.

Cependant, les résultats obtenus par simulation numérique ne se vérifient pas bien ou très mal par la confrontation à l'expérience. Cela peut provenir d'un modèle qui n'est pas assez représentatif des phénomènes en jeu. Mais ce modèle est capable de reproduire un phénomène réel donné, on peut alors parler de limitations de modèle.

Afin de pallier ces difficultés, deux solutions sont possibles : soit refaire un modèle ayant de faibles limitations, soit remplacer le modèle numérique par un prototype dans une boucle de simulation. C'est ce que propose la technique du Hardware in the Loop. L'intérêt de cette technique peut aussi être renforcé par le fait d'une utilisation en temps réel des systèmes, ce qui est ici un avantage en considérant un temps de simulation de l'ordre de la seconde, comparé à une réaction de l'ordre du dixième de seconde dans la réalité.

La technique du Hardware in the Loop est appliquée dans les deux chapitres suivant afin d'obtenir une simulation hybride réalisant un système en boucle fermée, puis réaliser l'optimisation expérimentale du sous-système actionneur.

6 Hybridation

Sommaire

Le chapitre précédent a présenté l'identification inverse de paramètres dans un modèle d'actionneur puis diverses optimisations sur le signal de commande de celui-ci ont été réalisées. De ce chapitre est ressorti les difficultés de construction de modèles ainsi que les limites d'utilisation de ceux-ci, notamment dans le cadre des optimisations de signaux de commande. Afin de palier à des carences de modélisations des composants, il est possible de remplacer un modèle par un composant (ou système) réel dans une modélisation plus large. C'est le concept du Hardware in the Loop ainsi que du Software in the Loop. Ce chapitre traite en particulier du Hardware in the Loop dans le cadre d'une modélisation globale d'un système mécatronique.

6.1 Introduction au HiL / SiL

En reprenant le modèle d'actionneur présenté dans le chapitre précédent et en l'insérant dans une boucle fermée d'asservissement, nous obtenons alors un MIL soit un Model In the Loop. Cela correspond de fait à la simulation complète d'un système mécatronique avec les différentes interactions existantes.

Cependant il peut s'avérer que l'une des parties du système mécatronique n'ait pas une modélisation suffisamment fine pour reproduire de façon fidèle un comportement réel.

L'idée du Hardware in the loop (HIL) est de remplacer cette partie modélisée défaillante par un système réel et de l'interfacer directement avec le reste du modèle numérique. Cet interfaçage impose alors que le modèle numérique doive réagir en temps réel avec le système réel.

Dans un autre sens, le Software in the loop (SIL) consiste à implanter un code dans son support matériel définitif et de le tester par l'intermédiaire d'une simulation du système qu'il doit, par exemple, contrôler.

Ce type de simulation mixte matériel / logiciel permet d'obtenir une souplesse intéressante pour le développement des produits. Il est ainsi possible de tester de façon réelle des sous-systèmes dès que ceux-ci sont disponibles et de pouvoir alors apporter d'éventuelles modifications sur les autres sous-systèmes.

Le but de l'application, présentée ici, est de palier à une modélisation incomplète des sous-groupes actionneur et pré-actionneur d'un système mécatronique en remplaçant les modèles par des composants réels instrumentés. Cette application s'inscrit donc dans un cadre HIL.

Le remplacement des modèles par des composants réels entraine alors une modification forte de la partie simulée. En effet les simulations s'inscrivent généralement dans un cadre hors-temps réel : une simulation peut s'exécuter plus rapidement ou plus lentement que le temps contenu dans la simulation. Or dans un cadre HIL, la simulation doit se dérouler en temps réel afin d'assurer une commande et un contrôle correct de l'actionneur.

La nécessité de travail en temps réel impose une réécriture des modèles : on passe d'équations continues à des équations discrètes soumises à des temps d'échantillonnage. Il se pose alors des questions quant au traitement des dérivations et intégrations. Une étude des différentes possibilités peut être trouvée dans [91], avec notamment le calcul des erreurs par rapport à une intégration ou dérivation analytique.

Exemple:

Soit un réservoir cylindrique de section S dont on connait un débit de remplissage $Q_{in}(t)$ et un débit de sortie $Q_{out}(t)$, la hauteur $h(t)$ de fluide est donnée par l'équation suivante :

$$h(t) = \int_0^t \frac{Q_{in} - Q_{out}}{S} dt \quad (72)$$

Une transposition de cette équation sous forme discrète et d'intégration régressive peut alors s'écrire, en posant T_e un temps d'échantillonnage pertinent :

$$h(nT_e) = h((n-1)T_e) + \frac{1}{S}[Q_{in}((n-1)T_e) - Q_{out}((n-1)T_e)]T_e \quad (73)$$

Cette équation est alors directement exploitable informatiquement et peut-être calculée en temps réel.

Lors de l'utilisation de système HIL se pose aussi le problème du bruit. Ce bruit, issu des mesures, peut entrainer des problèmes de divergence lors des simulations. Ce problème est surtout lié à l'opération de dérivation : lors de la présence d'un bruit l'opération de dérivation entraine une amplification de celui-ci de 20dB/octave. L'intégration, étant l'opération contraire, entraine une atténuation de 20dB/octave (filtre passe bas). On préférera donc une écriture intégrale des équations par rapport aux variables alimentées par des mesures.

6.2 Matériel nécessaire

Le matériel utilisé pour effectuer du Hardware et/ou Software in the Loop dépend de différents besoins et contraintes :

• si le modèle numérique est lourd et/ou possède des constantes de temps grandes :

Il est alors possible de réaliser la simulation sur un ordinateur classique en utilisant un langage type Modelica ou bien encore Simulink. Cela permet de ne pas avoir à modifier en profondeur le modèle et d'utiliser des cartes d'entrées / sorties standards.

• si le modèle numérique est lourd et/ou possède des constantes de temps moyennes :

L'utilisation d'un ordinateur est encore possible mais il peut être envisageable de travailler avec un environnement de calcul de type temps réel. Les cartes d'entrées sorties sont généralement capables de fonctionner avec ce mode.

• si le modèle est lourd et/ou avec des constantes de temps restreintes :

Deux solutions sont à envisager : soit simplifier le modèle afin de récupérer une partie de la puissance de calcul au prix d'une baisse de la précision du modèle, soit de changer de plateforme de calcul pour des puces de type DSP ou encore FPGA. Les puces DSP (Digital Signal Processing) peuvent gérer plusieurs processus mathématiques mais uniquement l'un après l'autre ce qui rend ces processeurs de type temps réel soft (on ne maitrise pas exactement le temps nécessaire à l'exécution d'un processus). Les puces FPGA (Field Programmable Gate

Array) ont l'avantage de pouvoir gérer, avec un timing parfaitement maitrisé donc du temps réel hard , des processus parallèles en contrepartie d'une programmation pouvant être plus difficiles qu'un DSP.

Il revient alors à effectuer un choix qui est un compromis en qualité de simulation, coût de développement et coût de matériel.

Les éléments d'entrées et sorties sont des composants indispensables pour la mise en œuvre d'un système HiL. Pour les éléments de sorties, une étude des limites de fonctionnement du système réel doit être réalisée sous peine d'endommager le composant réel. De plus la mise en place d'un banc de test doit aussi s'accompagner de sécurités poussées car le système peut avoir des réactions non prévues.

6.3 Application

Dans l'application choisie, qui est de réaliser un montage HiL du système de régulation de hauteur de fluide, est à disposition un actionneur piézoélectrique ainsi que son pré-actionneur, le reste du système étant à l'état de modèle.

Une analyse rapide du système réel est réalisée :

- le signal de commande de l'actionneur possède des fréquences de l'ordre de $500Hz$. Afin de réaliser ce signal, on considère qu'il est essentiel de générer la forme d'onde avec un échantillonnage d'au moins $5KHz$;
- le pré-amplificateur possède une tension d'entrée qui doit être comprise entre deux valeurs. Cela devient ainsi une contrainte pour l'interface de sortie;
- le reste du modèle doit être capable d'être simulé à la même fréquence que la génération de la forme d'onde.

De fait il peut apparaitre que les contraintes de gestion du système réel ainsi que la difficulté de simulation sont relativement faibles pour être acceptables pour un ordinateur classique. Cependant, le choix est fait d'utiliser un système de type ADWIN qui embarque un processeur de type DSP pour permettre une évolution du banc suite à une possibilité de miniaturisation et donc une augmentation de fréquences de contrôle.

Le premier travail à réaliser est donc la traduction du modèle vers le système temps-réel.

6.3.1 Traduction du modèle

Pour rappel, la partie fluidique de la simulation est composée d'un réservoir se remplissant avec un débit incertain et se vide par une valve et deux tuyaux. La valve étant mise en mouvement par le moteur piézoélectrique.

Trois variables constituent les frontières de la partie fluidique :

- la position de la valve qui est issue de la mesure de la position de l'arbre de l'actionneur;

• le débit de remplissage incertain du réservoir qui est réalisé par la mesure d'une tension ;

• la hauteur estimée de fluide contenu dans le réservoir qui est uniquement une variable informatique.

On va alors chercher à remanier le modèle de façon à ne pas avoir de dérivations par rapport aux variables issues de mesure.

Afin de simplifier le modèle, on pose les hypothèses suivantes :

• l'écoulement du fluide est laminaire :

Cela permet de ne gérer qu'un seul cas d'écoulement et donc d'ignorer régimes transitoires et turbulents;

• les efforts générés par la vanne sur l'actionneur sont considérés comme négligeables :

Cette hypothèse permet de ne pas créer de prototype mécanique de la vanne;

• l'écoulement en sortie du réservoir se fait sans pertes;

• le dénivelé du système fluidique est égal à la longueur des tuyaux;

• la vanne possède le même diamètre disponible pour le passage de fluide que les tuyaux;

Les équations du système sont alors les suivantes :

• Pour le réservoir :

$$h(t) = \frac{1}{S}\int_0^t (Q_{in} - Q_{out})dt \tag{74}$$

• Pour un tuyau :

$$\begin{cases} \Delta p = \lambda . \frac{L}{D} . \frac{\rho}{2} . v^2 \\ \lambda = \frac{64}{R_e} \\ R_e = \frac{|v|D}{\vartheta} \end{cases} \tag{75}$$

• pour la vanne :

$$\begin{cases} A_{inter} = 2R^2 \cos^{-1}\left(\frac{d}{2R}\right) - \frac{1}{2} d(4R^2 - d^2)^{\frac{1}{2}} \\ A_{max} = \pi R^2 \\ A = \frac{A_{max} - A_{inter}}{A_{max}} \end{cases} \tag{76}$$

$$\begin{cases} K_v = K_{vS}.A \\ \Delta p = \rho.\left(\frac{q}{K_v}\right)^2 \end{cases} \quad (77)$$

Comme la somme des pertes de charges du circuit est égale à la pression issue de la hauteur de fluide, il est possible d'obtenir une relation entre vitesse de fluide et la hauteur du fluide dans le réservoir

$$\frac{32\vartheta L}{D^2}v + \frac{\pi^2 D^4}{16K_v^2}v^2 = g(h+L) \quad (78)$$

Cette équation est donc simple à résoudre et linéaire, on ne garde que la solution positive. Elle présente aussi l'avantage de réaliser un estimateur de hauteur de fluide et donc de réaliser la boucle de rétroaction.

$$\hat{h} = \frac{1}{g}\left[\frac{32\vartheta L}{D^2}v + \frac{\pi^2 D^4}{16K_v^2}v^2\right] - L \quad (79)$$

Cette écriture analytique du modèle permet alors l'écriture du modèle embarqué. Seule l'équation du réservoir nécessite une adaptation car elle contient une intégrale, les autres équations étant directement linéaires, il n'est pas nécessaire de travailler dessus. L'adaptation est donc la suivante, en posant T_e le temps d'échantillonnage et de fonctionnement du système temps-réel et en insérant toutes les équations du modèle dans une boucle infinie :

$$\begin{cases} Initialisation\ des\ variables \\ While(1) \\ \\ h = volume/S_{reservoir} \\ K_v = K_{vs}.Taux\ \ d'ouverture \\ dp_{totale} = \rho g(L+h) \\ dp_{tuyau} = \frac{32L\rho\vartheta}{D^2}v \\ dp_{valve} = \rho\left(\frac{v.section}{K_v}\right)^2 \\ \\ \Delta = \left(\frac{32L\vartheta}{D^2}\right)^2 - 4.\frac{section^2}{K_v^2}.(-g(L+h)) \\ v = \frac{1}{2\frac{section^2}{K_v^2}}\left[-\frac{32L\vartheta}{D^2} \pm \sqrt{\Delta}\right] \\ \\ q_{out} = v_+.section \\ volume = volume + q_{in}.Te - q_{out}.Te \\ \ldots \\ end \end{cases} \quad (80)$$

Le modèle est implanté dans un ADWIN Gold II en utilisant une fréquence d'échantillonnage de 30KHz.

6.3.2 Test de performance du modèle embarqué en temps réel

Ce test de performance est nécessaire pour valider les points suivants :

• le système temps réel étant de type soft, il convient de vérifier que le temps d'échantillonnage intervenant dans le calcul de l'intégrale du réservoir est respecté;

• vérifier que des erreurs d'arrondis de calcul n'entrainent pas une divergence des résultats;

• enfin l'influence des bruits de mesure sur les résultats ne doit pas être trop importante.

Pour réaliser ces tests, le même code est implanté dans un logiciel de calcul numérique puis les résultats sont comparés à ceux du système embarqué.

Ce test consiste au test de vidage du réservoir sans que la fonction de remplissage soit active, et avec la vanne ouverte au maximum. On relève alors la courbe de hauteur de fluide dans celui-ci et l'on cherche le moment où le niveau de fluide devient nul. Cela permet de vérifier que les temps d'écoulement entre le système temps réel et la simulation numérique sont identiques.

Les conditions initiales du test sont reportées dans la Table 31.

Variable	Valeur
Pression atm du réservoir	0 Pa
Pression atm de la sortie	0 Pa
Volume initial de fluide	1e-6 m^3
Débit de remplissage	0 m^3
Densité du fluide	1000 kg/m^3
Viscosité cinématique du fluide @ 20°C	1e-6 m^2/s
Vitesse initiale du fluide	0 m/s
Longueur de tuyau	1.3 m
Diamètre de tuyau	1.1e-3 m
Dénivellation	1.3 m
Surface de réservoir	7.854e-5 m^3
Caractéristique de la vanne	0.0036 $m^3/h\ et\ bar$
Gravité	9.81 m/s^{-2}

Table 31 : Table des paramètres pour validation du système temps réel

Le pas de calcul choisi pour la simulation numérique est de $1s$. Le système en temps réel est réglé pour avec un pas de temps à $30KHz$. Le résultat donné par la simulation donne un temps de vidage de $887s$. Le système en temps réel, en faisant une acquisition de l'état de ses calculs toutes les secondes, donne un résultat de $884.4s$. L'écart est donc de 0.3%.

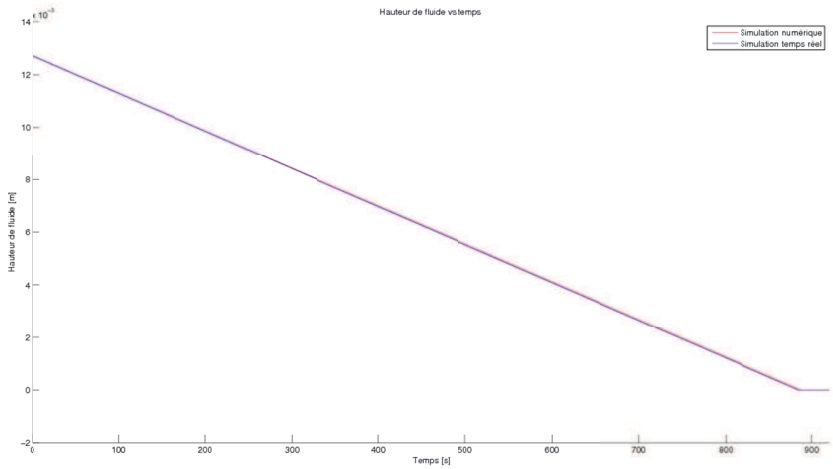

Figure 43 : Comparatif entre le système temps réel et la simulation numérique

Les résultats entre le système temps réel et par calcul classique sont identiques. Les seuls problèmes rencontrés lors du travail avec un signal bruité sont qu'il est possible que ce

signal soit égal à zéro ou encore prennent des valeurs négatives pouvant alors générer des divisions par zéro ou encore rendre des résultats physiquement irréalistes

6.3.3 Création d'un macro-modèle d'estimation de position de l'actionneur

Dans la boucle d'asservissement établie précédemment, la position de l'actionneur est directement issue de l'utilisation du capteur de position laser. Dans l'optique de ne pas instrumenter l'actionneur, il peut être envisageable d'utiliser un estimateur de position de celui-ci. Deux solutions sont possibles :

- insérer un modèle de l'actionneur :

Cette solution permet d'obtenir de bons résultats avec une bonne précision mais requière une puissance de calcul non négligeable. Pour le système ADWIN, cette solution est peu réalisable.

- créer un macro-modèle de l'actionneur :

La création de ce macro-modèle peut se faire en utilisant par exemple des réseaux de neurones mais aussi par l'utilisation de distribution statistique. Dans le cas présent, on réalise ce macro-modèle par l'analyse statistique du déplacement effectué par période de signal de commande. Les résultats se présentent donc sous la forme d'une distribution statistique, ce qui requière un coût mathématique faible.

La loi de distribution retenue est celle de Weibull car elle permet de décrire une grande variété de distributions. Afin d'établir les paramètres de la loi, il est nécessaire d'effectuer une succession de mesures de déplacement pour constituer une base de mesure conséquente. Le signal de commande retenu est celui du constructeur.

L'actionneur fonctionnant dans un sens positif et un sens négatif, on obtient alors deux graphiques de représentation (Fig. 44 & 45) de la distribution de Weibull.

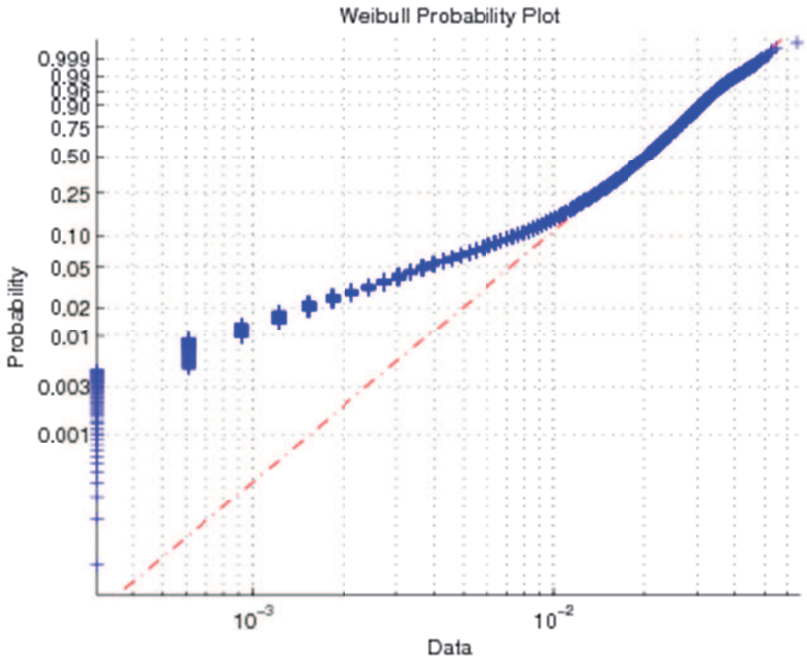

Figure 44 : Graphique de Weibull pour le sens positif

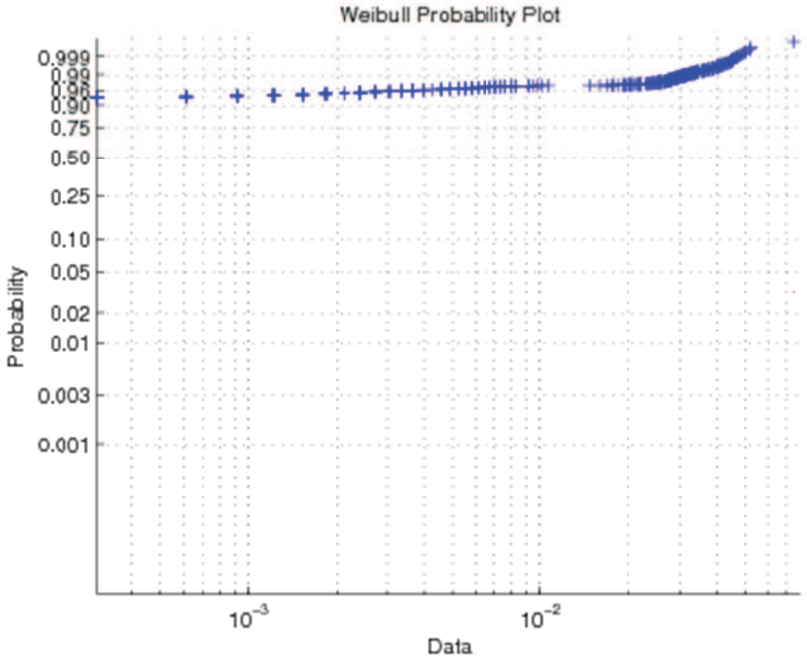

Figure 45 : Graphique de Weibull pour le sens négatif

De ces deux graphiques de distribution, il est possible de tirer les conclusions suivantes :

• les graphiques des deux sens ne possèdent pas les mêmes caractéristiques, il est donc nécessaire d'établir un modèle pour chaque sens de déplacement.

• dans chaque graphique, on peut remarquer que les données ne forment pas une

droite. Cela correspond au fait qu'il n'existe pas une unique distribution permettant de décrire le déplacement. Deux droites semblent convenir pour approximer la courbe obtenue et donc générer deux distributions.

L'absence de distribution unique pour décrire un déplacement est un facteur limitant. En effet, pour obtenir un estimateur correct de position, il est nécessaire de répertorier les zones où l'une des deux lois décrivant un déplacement est active. Une solution à ce problème peut être de déterminer un signal permettant d'obtenir une distribution unique. En l'état actuel, la présence d'un capteur de position s'avère nécessaire pour l'estimateur de position.

6.3.5 Application

Dans cet exemple, on cherche à maintenir un niveau de hauteur de fluide dans le réservoir. Un contrôleur de type tout ou rien est implanté afin de réaliser la boucle d'asservissement introduisant alors une zone morte autour du niveau souhaité pour ne pas entrainer de fonctionnement intempestif de l'actionneur. D'autres types de correcteur / contrôleurs sont envisageables pour obtenir un fonctionnement optimisé de l'actionneur.

On utilise le capteur laser comme source pour la simulation en temps réel du sous-système fluidique. La simulation du remplissage du réservoir se fait par l'utilisation d'un GBF fournissant un signal carré. Afin de palier à des problèmes de calcul liés à la fermeture totale de la vanne, on considère que la vanne possède une ouverture de fuite minimale.

La Figure 46 présente un exemple de fonctionnement du système : la courbe rouge présente un écart de hauteur de fluide par rapport à une position souhaitée, la courbe en pointillé la position de l'actionneur et la courbe pleine le débit de remplissage du réservoir. Le système en temps-réel, qui est donc utilisé pour la commande / contrôle de l'actionneur mais aussi en charge de la simulation de la partie fluidique, est réglé pour fonctionner à 30*KHz*.

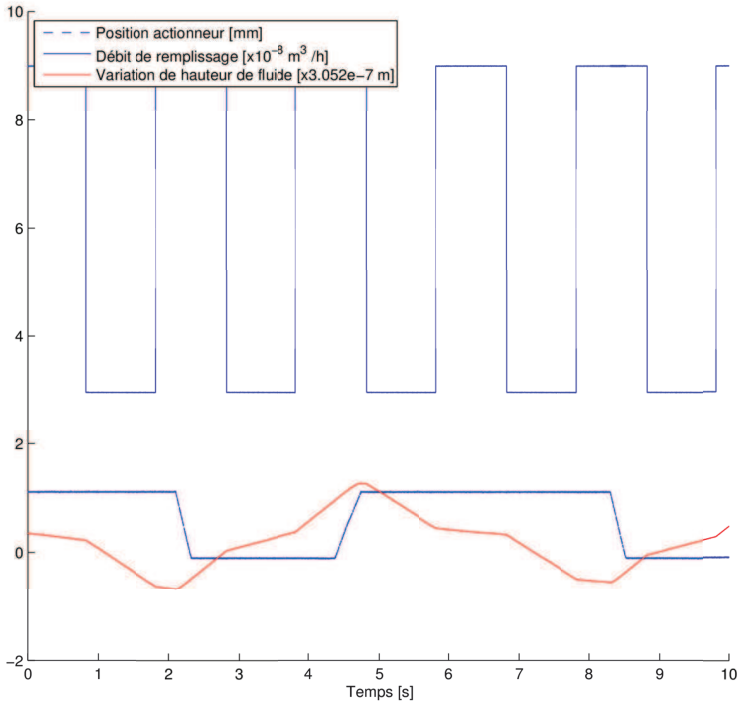

Figure 46 : Exemple de comportement du système HIL

On peut constater que le système HIL répond au cahier des charges à savoir que la hauteur de fluide est régulée dans le temps. Le comportement tout ou rien de la commande étant validé, il peut être envisageable d'obtenir une régulation plus appropriée

6.4 Conclusion

Le principe du HiL / Sil permet ainsi de pouvoir utiliser des composants réels dans une boucle d'asservissement en simulation. Ce genre de système permet ainsi de pouvoir réduire le temps de développement, par exemple, des organes de contrôle de sous-systèmes. Ce type de montage expérimental permet aussi de pouvoir recréer des environnements de tests extrêmes et d'observer / corriger rapidement la conception des éléments encore non prototypés. Enfin le HiL / SiL permet de ne pas réaliser une étude théorique poussée d'un composant afin d'en établir un modèle complexe. Le coût de réalisation du prototype étant alors à mettre en balance avec le coût de création d'un modèle. Le chapitre suivant va aller plus loin dans le HiL / SiL en proposant cette méthode pour l'optimisation des performances du groupe pré-actionneur / actionneur.

7 Optimisation expérimentale

Contenu

Ce chapitre traite de l'optimisation expérimentale des systèmes. L'optimisation expérimentale désigne un processus d'optimisation dont la fonction objectif, au sens mathématique du terme, est remplacée par une mesure sur un système ou sous-système réel.

Ce type d'optimisation est intéressant pour plusieurs aspects :

- le système étudié peut être utilisé dans des conditions d'utilisation réelles, c'est à dire qu'il sera soumis aux mêmes phénomènes (climatiques, incertitudes, ...) qu'en phase d'exploitation,
- il n'est pas nécessaire de passer par une phase de modélisation qui peut être difficile à réaliser de par la complexité du système et du coût de cette modélisation,
- dans le cadre des systèmes ayant un temps d'action plus rapide que le temps de simulation de celui-ci, l'optimisation expérimentale permet d'obtenir des résultats directement exploitables plus rapidement que par simulation.

Cependant, ces avantages sont à mettre en opposition avec les points suivants :

- il est nécessaire d'avoir à sa disposition un prototype du système mais qui est aussi requis lors de la validation du modèle,
- le système est un système réel qui peut s'avérer dangereux pour son environnement (utilisateur, installations, ...), ainsi que pour son intégrité. Il est donc obligatoire d'avoir une connaissance des limites de celui-ci et de mettre en place des systèmes de sécurité performants,
- le système doit être instrumenté et contrôlé de manière pertinente par rapport au but de cette optimisation : le coût des capteurs, ainsi que du système de contrôle adapté, peuvent devenir non négligeables,
- le résultat de la fonction objectif étant issu d'une mesure, celui-ci est donc considéré comme bruité, rendant certains processus d'optimisation inopérants (par exemple les processus à base de gradient). De plus, le système mis en œuvre étant réel et donc soumis à d'éventuels phénomènes de rodage et d'usure, le résultat de la fonction objectif peut apparaitre comme variant dans le temps.

L'optimisation expérimentale a été utilisée, avec succès, dans les exemples suivants :

- à partir d'algorithmes génétiques et d'un montage de type HiL, Wozniak [83] propose de définir les paramètres d'un correcteur PI pour la commande d'un moteur électrique à aimants permanents. Il souligne que cette méthode peut être plus rapide et efficace que la mise en place d'un modèle puis son optimisation.
- en utilisant une optimisation par simplex et une mesure en sortie de ligne de production, Kvale [36] optimise la taille de microsphères destinées aux industries pharmaceutiques et cosmétiques. Cette optimisation permet alors de compenser d'éventuelles variations de caractéristiques des produits d'entrée en jouant sur des paramètres tels que les débits d'entrée des composants chimiques.
- en utilisant un PSO, Kao [32] réalise l'optimisation d'un système mécanique en jouant sur les paramètres d'un correcteur PID. Il y souligne aussi l'intérêt de cette méthode par rapport à des difficultés éventuelles de réalisation de modèle pour des systèmes complexes.
- à l'aide d'une fonction d'optimisation contenue dans Matlab (FMINCON), Rogers [61] propose d'optimiser la commande d'un actionneur pas à pas par le choix du temps d'envoi

des pulses de commande.

L'optimisation expérimentale peut alors être une alternative à une optimisation par simulation et cela dans des domaines aussi variés que la chimie, l'automatique ou encore la commande des systèmes.

Ce chapitre va alors appliquer le procédé d'optimisation expérimentale sur un actionneur piézoélectrique réel et comparer les résultats obtenus à ceux d'une simulation informatique.

7.1 Mise en place d'un banc d'optimisation

Le banc d'optimisation mis en place a pour but d'effectuer diverses optimisations sur un sous-système actionneur accompagné d'un pré-actionneur. L'actionneur est le même que celui étudié précédemment. Le banc de test se compose d'un ordinateur ayant pour tâche de mettre en œuvre le processus d'optimisation retenu, ainsi que d'assurer une communication avec l'organe de commande et de contrôle, et avec l'utilisateur. Dû à la rapidité de mouvement de l'actionneur étudié, la commande et le contrôle ne peuvent s'effectuer directement par un ordinateur équipé d'entrées / sorties. Un système embarqué est retenu pour réaliser cette tâche : un ADWIN [1]. Ce système possède diverses E/S ainsi qu'une puce programmable. Les différentes E/S du système à optimiser seront reliées à cet ADWIN, et un bornier d'acquisition de données sera aussi utilisé en complément. Différents capteurs seront utilisés pour ces optimisations : un capteur de déplacement laser, une sonde de tension, ainsi qu'un capteur d'intensité. Le système embarqué sera reprogrammé suivant le ou les buts des optimisations. Le processus d'optimisation, devant à la fois être robuste (insensible au bruit) et pouvant suivre des minima variant dans le temps, sera un essaim de particules modifié, tel que présenté dans la Particle Swarm Optimization Toolbox de Scilab.

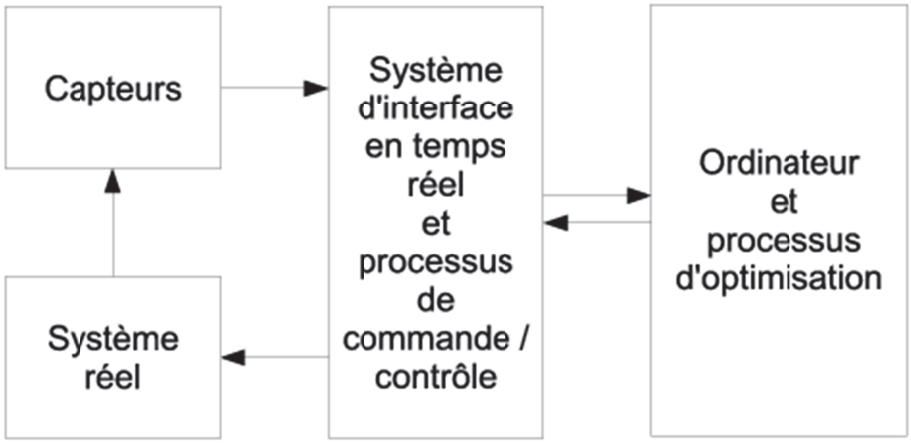

Figure 47 : Architecture générale d'un banc d'optimisation expérimental

7.2 Optimisations

Les optimisations réalisées sont identiques à celles de la partie numérique. Celles-ci sont réalisées de façon expérimentale puis les résultats seront réinjectés dans la simulation numérique pour comparaison. Toutes les optimisations sont robustes et la robustesse réalisée se fait à priori.

7.2.1 Optimisation robuste en vitesse

Le but de cette optimisation est de maximiser la vitesse de déplacement de l'actionneur par l'utilisation d'un signal adapté. Comme pour l'optimisation numérique, quatre signaux sont testés : le carré, la dent de scie, le triangle, et enfin la spline à 5 points.

Dans un premier temps, les quatre signaux sont testés sur l'actionneur à vide puis on chargera l'actionneur avec une masse en utilisant la spline.

Un bornier d'acquisition est aussi inséré dans le banc de test afin d'assurer l'acquisition de la tension générée ainsi que de la courbe de déplacement de l'actionneur par un capteur laser.

Comme le capteur utilisé est un capteur de déplacement et non un capteur de vitesse, la mesure de la vitesse est remplacée par la mesure du temps nécessaire au parcours d'une distance de $2mm$. Cependant, compte-tenu du fonctionnement de type pas à pas du système et de la méthode de commande de l'actionneur (retour à zéro de la tension obligatoire) la mesure de temps est réalisée sur la courbe de tension. L'acquisition se fait sur 2 secondes à $50KHz$.

La fonction objectif peut se définir de la façon suivante :

- dans le cas où l'actionneur arrive à réaliser le déplacement :

$$f = t_{mesure} \tag{81}$$

- dans le cas où le déplacement n'est pas satisfaisant :

une pénalisation se met en place par la formule suivante :

$$f = 10 + (1 - distance_{parcourue}) \tag{82}$$

- enfin si les données d'entrées ne respectent pas les contraintes matérielles :

$$f = 1e9 \tag{83}$$

Les contraintes matérielles sont définies ainsi :

- les différents points composants la spline doivent être compris entre 0 et 150V.

Pour des raisons pratiques, cette contrainte est remplacée par une tension normée comprise entre 0 et 1.

• la fréquence de la spline doit être positive. Encore une fois cette contrainte est remplacée par un nombre d'échantillons à générer par période par l'ADWIN entre 0 et 6000 .

En considérant que cette optimisation doit être robuste, la stratégie suivante est adoptée pour la modification de l'algorithme d'optimisation :

• lors de l'évaluation de l'essaim, la fonction objectif est directement évaluée (fonctionnement normal)

• lors de la sélection du $Fbest$, la fonction objectif est évaluée successivement 14 fois. La meilleure et plus mauvaise mesures sont rejetées, et donc 12 mesures gardées. De ces mesures restantes, est établie une nouvelle fonction objectif définie par la moyenne et l'écart-type:

$$f_{bis} = \bar{f} + \hat{\sigma}(f) \tag{84}$$

L'algorithme de référence sera ici un PSO classique dont les paramètres sont définis dans la Table 32.

Nom de la variable	Symbole	Valeur
Paramètre d'inertie initial	ω_{ini}	0.9
Paramètre d'inertie final	ω_{fin}	0.6
Nombre de particules	N	20
Vitesse de variation des samples	V_s	+/- 600
Vitesse de variation de tension	$V_{tension}$	+/- 0.1
Plage de lancer des samples		500 – 6000
Nombre d'itérations max	$itmax$	300
Paramètre de connaissance C1	C_1	2
Paramètre de connaissance C2	C_2	2

Table 32 : Table des paramètres du PSO pour l'optimisation en vitesse

Signal	Rapport cyclique	Fréquence	Vitesse réelle	Vitesse simulée	Écart
	[%]	[Hz]	[m/s]	[m/s]	[%]
Carré	37	500	$\begin{cases} m = 1.47e^{-2} \\ \sigma = 2.52e^{-4} \end{cases}$	$-4.8e^{-2}$	
Dents de scie		704	$\begin{cases} m = 1.81e^{-2} \\ \sigma = 1.09e^{-4} \end{cases}$	$1.52e^{-2}$	16
Triangle	88	680	$\begin{cases} m = 1.89e^{-2} \\ \sigma = 1.72e^{-4} \end{cases}$	$2.29e^{-2}$	21
Spline		417	$\begin{cases} m = 2.08e^{-2} \\ \sigma = 3.73e^{-4} \end{cases}$	$1.9e^{-2}$	8.6
Constructeur			$\begin{cases} m = 1.04e^{-2} \\ \sigma = 1.49e^{-4} \end{cases}$		

Table 33 : Table des résultats pour l'optimisation en vitesse

L'actionneur est testé dans des cas de charges différents : à vide puis sous charge de 20 et 40 grammes. La charge étant placée sur l'arbre.

Nom de la variable	Unité	A vide	sous 20g	sous 40g
Point 1 spline	V	0.668	0.247	0.234
Point 2 spline	V	0.004	0.985	0.928
Point 3 spline	V	0.259	0.629	0.985
Point 4 spline	V	0.51	0.017	0.071
Point 5 spline	V	0.992	0.283	0.487
Fréquence spline	Hz	418	497	466
Temps moyen	s	$4.13e^{-2}$	$4.88e^{-2}$	$5.53e^{-2}$
Ecart type	s	$4.22e^{-5}$	$8.02e^{-4}$	$5.51e^{-4}$
Temps de référence	s	$8.8e^{-2}$	$1.14e^{-1}$	$1.2e^{-1}$
Ecart type	s	$1.3e^{-3}$	$1.4e^{-3}$	$9.25e^{-4}$
Gain de temps	%	53	57	54

Table 34 : Table de résultats

Le fait de travailler sur des données expérimentales entraine un bruitage du résultat de la fonction objectif ainsi qu'une non décroissance locale de celle-ci. La Figure 48 montre ces différents défauts.

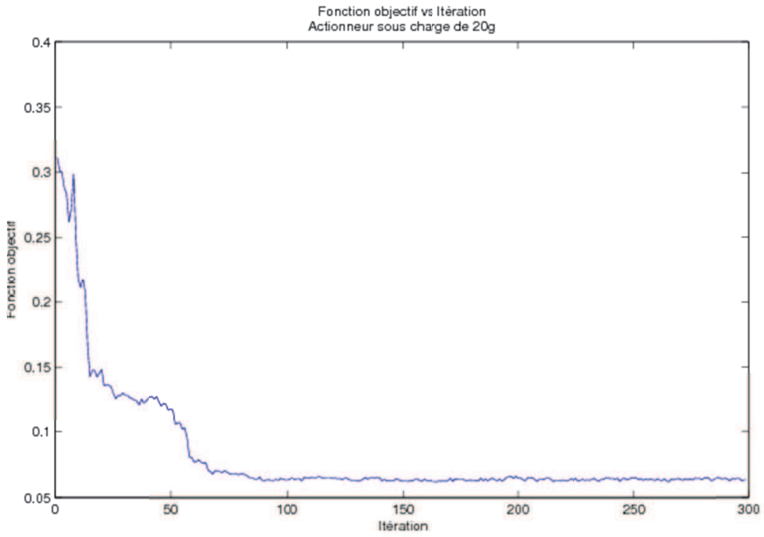

Figure 48 : Résultat de la fonction objectif pour l'optimisation en vitesse, actionneur chargé à 20g

La spline finale pour l'optimisation en vitesse à vide est présentée par la Figure 49

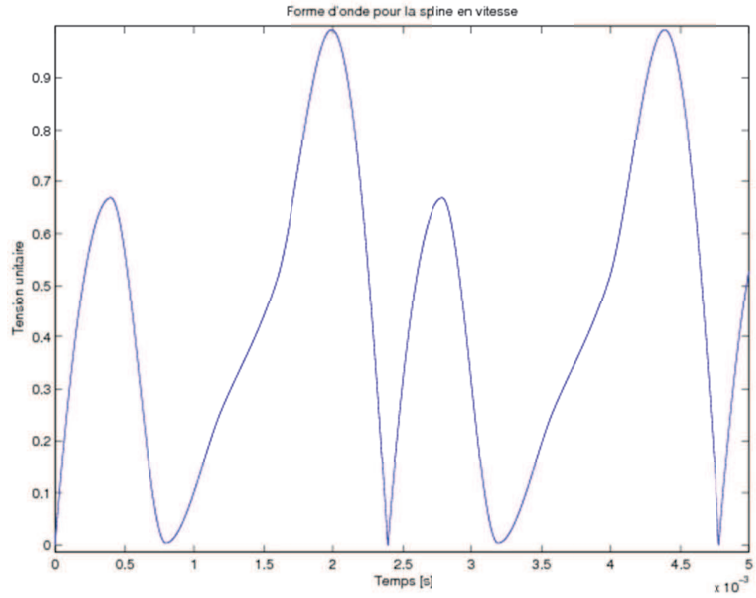

Figure 49 : Forme d'onde de la spline pour l'optimisation en vitesse, actionneur à vide

7.2.2 Optimisation énergétique robuste

Le but de cette optimisation est de réduire la quantité d'énergie circulant dans l'actionneur au cours d'un déplacement. Deux capteurs supplémentaires sont donc utilisés pour cette mesure : un capteur de tension ainsi qu'un capteur d'intensité reliés au bornier d'acquisition afin d'effectuer une mesure synchrone.

Comme pour l'optimisation numérique, quatre signaux sont testés : le carré, la dent de scie, le triangle ainsi que la spline 5. On n'utilisera, pour cette optimisation, qu'un actionneur non chargé.

L'écriture de la fonction objectif se fait comme pour les autres optimisations :

- si l'actionneur effectue un déplacement total satisfaisant :

$$P(nTe) = U(nTe).I(nTe)$$
$$f(x) = \sum_{n=1}^{t_{final}/Te} |P(nTe)|.Te \quad (85)$$

avec Te le temps d'échantillonnage de l'interface d'acquisition, t_{final} le temps final d'acquisition, $U(nTe)$ et $I(nTe)$ les valeurs de tension et d'intensité au temps nTe. Ici Te est de 1/50000.

- si l'actionneur ne parvient pas à réaliser le déplacement de façon satisfaisante :

$$f(x) = 1e9 - (cible - deplacement) \quad (86)$$

- dans le cas où les paramètres d'entrées ne sont pas dans un domaine admissible :

$$f(x) = 1e16 \quad (87)$$

Le processus d'optimisation utilisé est un PSO qui est modifié afin d'obtenir une robustesse par le calcul de la moyenne de 12 évaluations de la meilleure particule ou d'un candidat à ce poste. Les paramètres du PSO sont consignés dans la Table 35.

Les résultats obtenus montrent que la spline (Table 37 & Fig. 51) est le signal permettant une mise en mouvement de l'actionneur avec le moins d'énergie possible (Table 36).

Cependant l'injection des différents signaux optimisés dans la simulation montre des écarts important avec l'expérience mais conserve le classement de ces différents signaux (Table 36).

Cette optimisation apparait comme étant plus difficile à mettre en œuvre que la précédente car le résultat de la fonction objectif montre d'importantes non-décroissances locales (Fig. 50).

Nom de la variable	Symbole	Valeur
Paramètre d'inertie initial	ω_{ini}	0.9
Paramètre d'inertie final	ω_{fin}	0.4
Nombre de particules	N	20
Dimension du problème	D	suivant signal
Vitesse de variation de fréquence	V_f	+/- 400
Vitesse de variation de rapport cyclique	V_α	+/- 0.1
Plage de lancer en fréquence	f	1000 – 4000
Plage de lancer rapport cyclique	α	0.1 – 0.9
Nombre d'itérations max	$itmax$	200
Radius	R	10
Compteur Radius		10
Nombre de Raptors		20
Coefficient d'accélération des Raptors		2
Probabilité de lancement		0.1
Paramètre de connaissance C1	C_1	2
Paramètre de connaissance C2	C_2	2

Table 35 : Table des paramètres du PSO

Signal	Rapport cyclique	Fréquence	Tension	Énergie réelle	Énergie simulée	Écart
	[%]	[Hz]	[V]	[J/m]	[J/m]	[%]
Carré	20	227	136.5	$\begin{cases} m = 103 \\ \sigma = 0.61 \end{cases}$	292	183
Dents de scie		568	135.5	$\begin{cases} m = 55.88 \\ \sigma = 0.58 \end{cases}$	75.65	36
Triangle	68	625	143.1	$\begin{cases} m = 18.45 \\ \sigma = 0.19 \end{cases}$	41.24	127
Spline		391		$\begin{cases} m = 13.67 \\ \sigma = 0.14 \end{cases}$	30.88	126
Constructeur		400		$\begin{cases} m = 41.1 \\ \sigma = 1.04 \end{cases}$		

Table 36 : Table des résultats pour l'optimisation énergétique

Point n°	1	2	3	4	5
Tension unitaire	0.09	0.33	0.72	0.17	0.12

Table 37 : Table de résultat de la spline pour l'optimisation énergétique

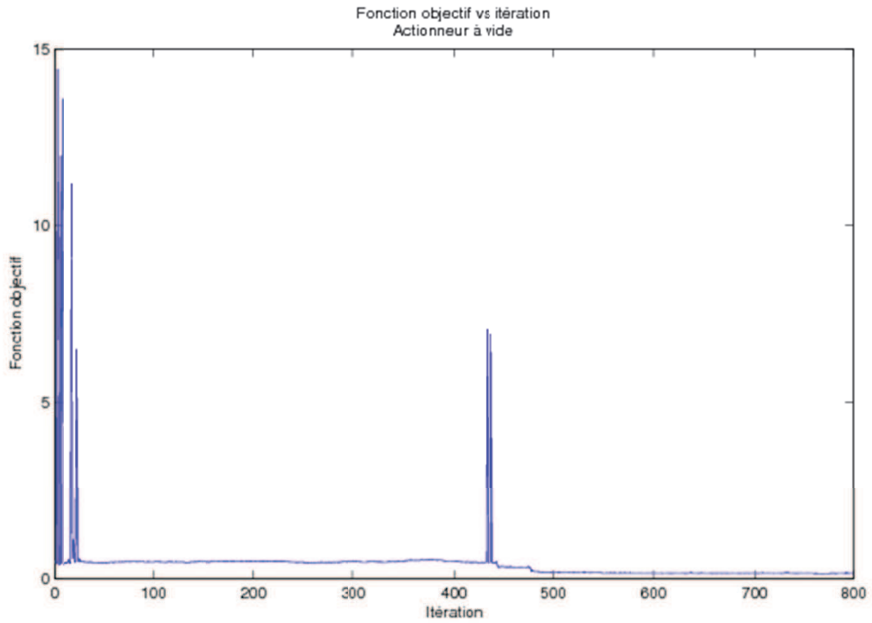

Figure 50 : Exemple de courbe de fonction objectif pour un actionneur à vide en optimisation énergétique avec spline

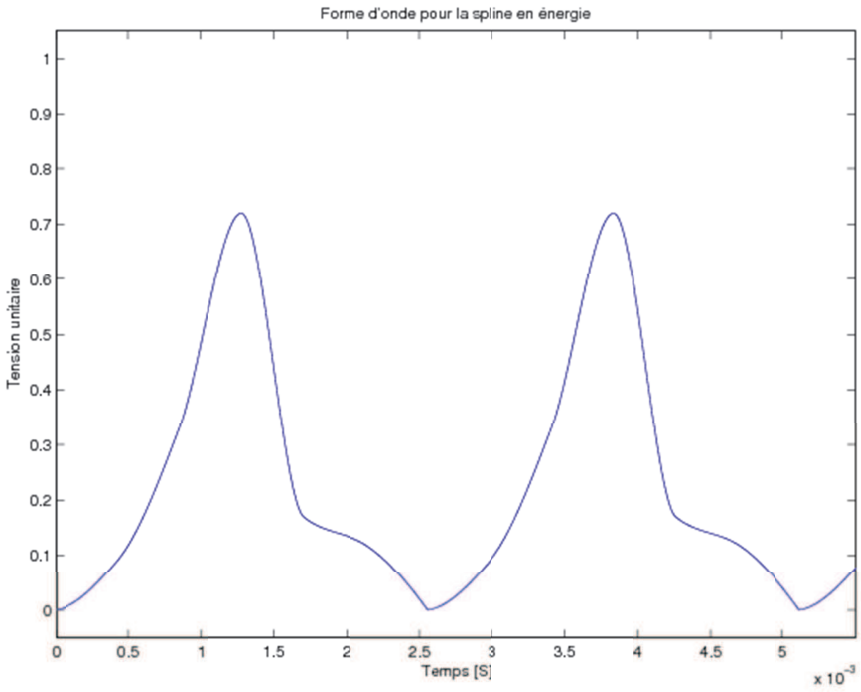

Figure 51 : Forme d'onde de la spline pour l'optimisation en énergie, actionneur à vide

7.2.3 Optimisation robuste pour l'estimateur de position

Le but de cette optimisation est de faire en sorte que l'actionneur, suive une loi statistique unique de déplacement. Cette unicité permet alors de pouvoir établir un macro-modèle de l'actionneur afin de réaliser l'estimateur de position proposé dans le chapitre précédent.

En utilisant le signal du constructeur, comme le montrent les figures du chapitre précédent (Fig. 44 & 45), l'actionneur suit deux distributions de type Weibull. L'optimisation a alors pour but de faire que le graphique de Weibull suive une droite unique.

A la vue des résultats précédents, on s'intéresse uniquement au signal de type spline à 5 points.

En reprenant les fonctions de contraintes des optimisations précédentes, ainsi que celle de la spline, la fonction objectif se traduit par la norme de l'écart entre les points constituant le graphique de Weibull et une régression linéaire de ceux-ci.

Cependant, l'usage direct des mesures sur le graphique de Weibull entraine la présence de points de mesures solitaires qui perturbent le résultat de la fonction objectif. Ces points sont alors filtrés pour ne garder que ceux qui sont multiples.

Le PSO utilisé est le même que celui de l'optimisation en vitesse (Table 32).

Nom de la variable	Valeur
Point 1 spline	0.32
Point 2 spline	0.45
Point 3 spline	1
Point 4 spline	0.13
Point 5 spline	0.08
Fréquence spline	1740
Ecart moyen	14%
Ecart type	$3.59e^{-2}$
Delta	-2.25
Beta	256
Eta	2.29

Table 38 : Table de résultats pour l'estimateur

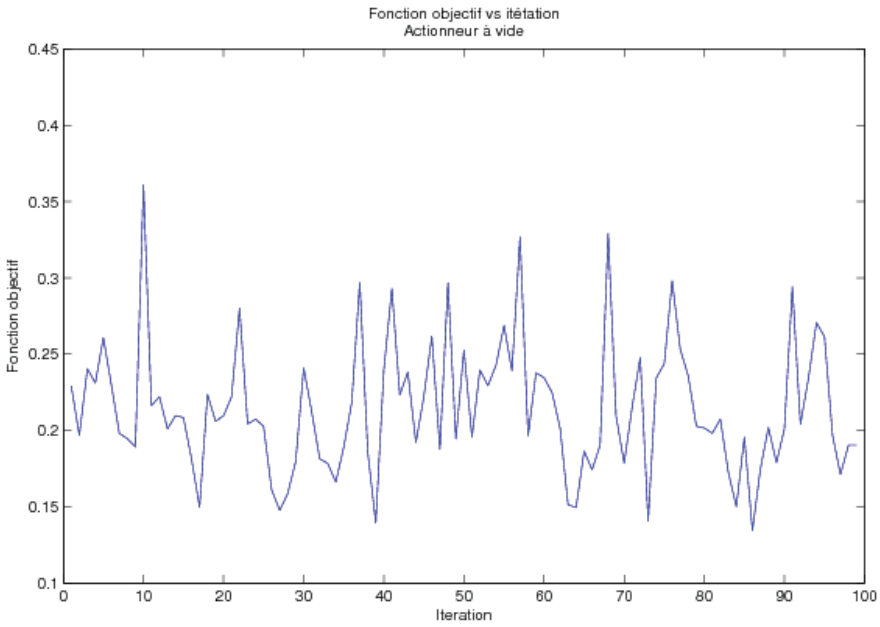

Figure 52 : Exemple de courbe de fonction objectif pour un actionneur chargé sous 20g en optimisation pour estimation de position avec spline

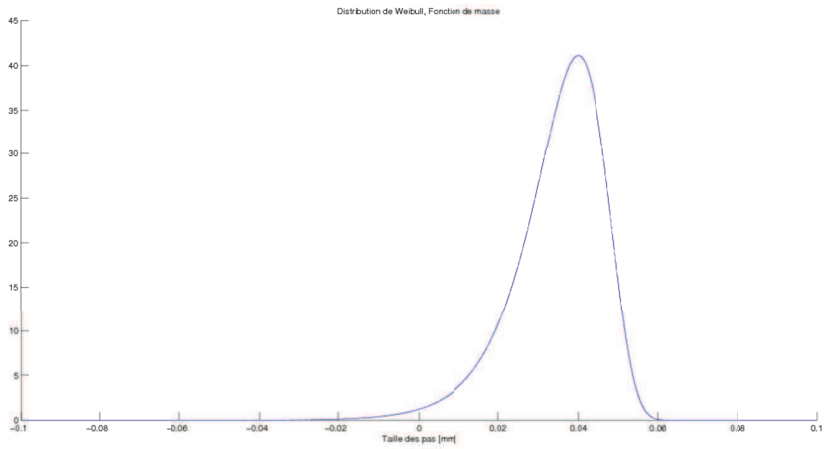

Figure 53 : Fonction de masse de la distribution de Weibull obtenue

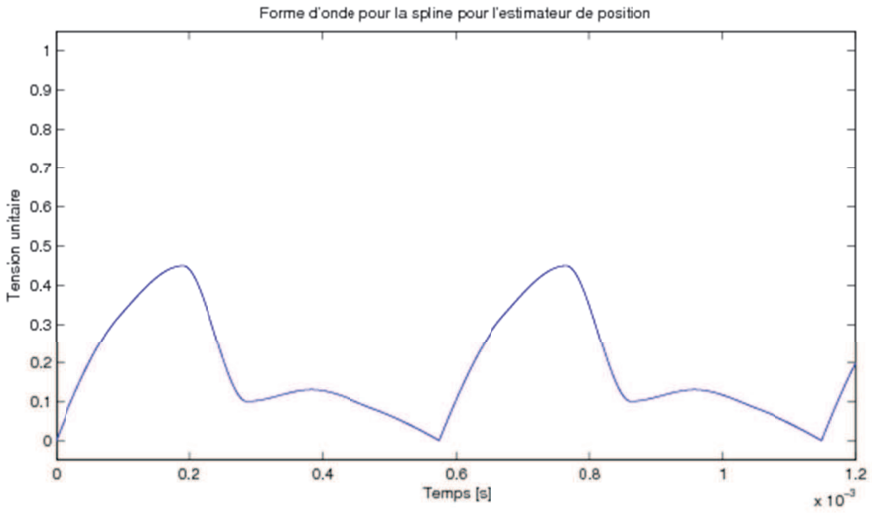

Figure 54 : Forme d'onde de la spline pour l'estimateur de position

Bien que la valeur du β soit élevée (Table 38), la majorité des valeurs de taille de pas sont contenues entre 20 et $40\mu m$, ce qui apparait comme étant insuffisant pour réaliser un estimateur efficace. L'idéal étant d'obtenir une distribution ayant un dirac comme fonction de masse, contrairement à la distribution obtenue par l'optimisation (Fig. 53).

Cette optimisation s'avère très difficile, voire impossible, sur cet actionneur car on ne note pas de convergence de la fonction objectif : sa courbe est uniquement constituée d'un bruit (Fig. 52).

La spline de commande obtenue est présentée dans la Figure 54.

7.3 Analyse fréquentielle des signaux

Comme pour la partie concernant l'optimisation numérique, une analyse fréquentielle des signaux est réalisée. On s'intéresse ici uniquement à la spline compte tenu de ces résultats par rapport aux autres signaux employés.

Une analyse FFT est de nouveau réalisée (Fig. 55 & 56 & 57) sur les splines. La spline en vitesse possède une fondamentale à $418Hz$ ainsi qu'une harmonique à $836Hz$, la spline en énergie a sa fondamentale à $391Hz$ et son harmonique à $782Hz$ et la spline pour Weibull a une fondamentale de $1740Hz$ et une harmonique de $3480Hz$.

On retrouve alors des résultats équivalents pour la spline en énergie par rapport aux résultats de simulation mais pas pour la spline en vitesse avec une fréquence plus basse. Pour la spline de Weibull, il est difficile de conclure car l'optimisation n'a pas permis d'obtenir un résultat satisfaisant.

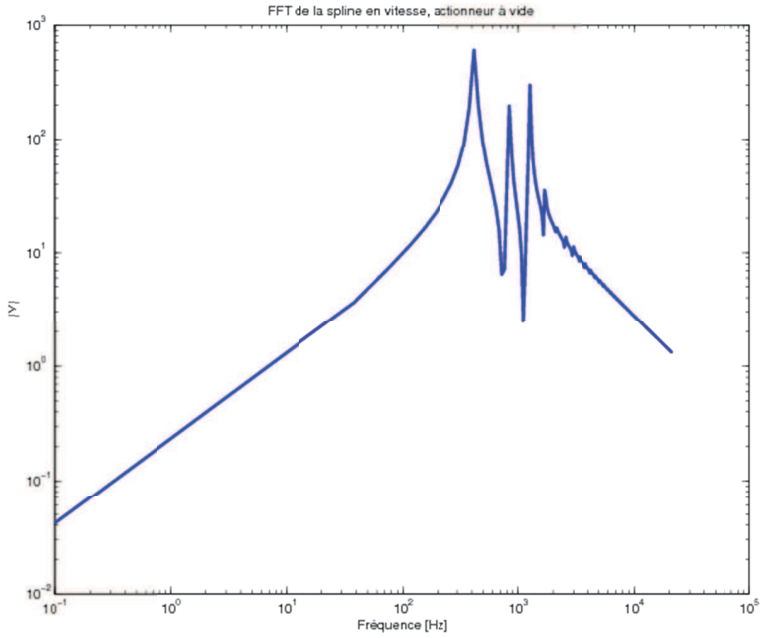

Figure 55 : FFT de la spline en vitesse pour l'actionneur à vide

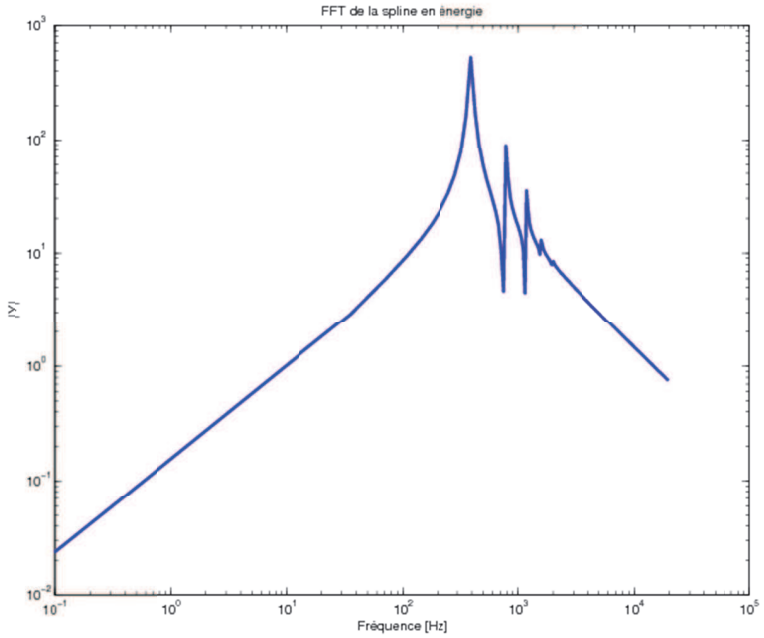

Figure 56 : FFT de la spline en énergie

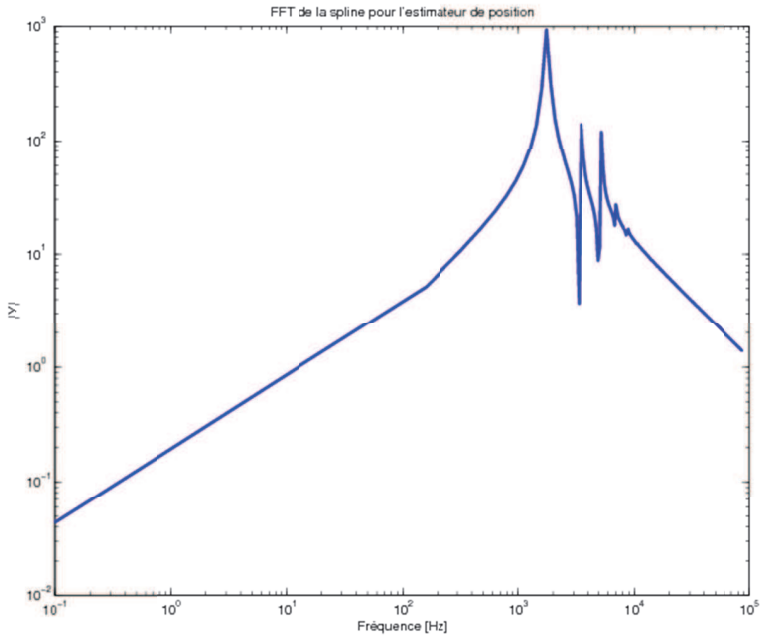

Figure 57 : FFT de la spline pour l'estimateur de position

7.4 Conclusion

Dans les cas d'optimisation énergétique ainsi que de vitesse, l'usage du signal libre apparait comme étant plus efficace que le signal particulier du constructeur, mais aussi des signaux canoniques (carré, triangle et dent de scie). Cela confirme les résultats de la partie numérique pure par un autre moyen. La réintroduction des résultats des signaux expérimentaux dans la simulation donne des résultats plus proches que l'inverse. Ces résultats peuvent confirmer le fait que les limitations du modèle sont liées à une modélisation incomplète du pré-actionneur. Malgré le succès des optimisations en vitesse et énergétiques, l'optimisation pour l'estimateur de position, par une représentation utilisant une distribution de Weibull, n'est pas assez performante pour pouvoir utiliser ce méta-modèle. Il est donc bien nécessaire d'inclure un capteur de position dans la boucle d'asservissement.

Conclusion

Le but de cette thèse était de fournir des pistes de résolutions pour des problèmes de caractérisation, d'identification de paramètres ainsi que d'optimisation de système mécatroniques. Ces voies de résolutions passent alors par la mise en place de modèles numériques, de banc d'expériences aussi bien numériques, physiques par l'expérimentation sur composants réels qu'hybride, en alliant efficacement le numérique et l'expérimental.

Le chapitre premier a permis, par l'introduction de définitions, de cadrer le domaine d'étude de la thèse : les systèmes mécatroniques. Il a aussi permis d'observer les méthodes de conception et développement de ces systèmes. Celles-ci sont conçues de façon à réduire le délai de mise sur le marché par la prise en compte précoce de validations sur prototype, ainsi que par l'intégration des interactions entre composants du système. Ces méthodes introduisent alors les concepts de modélisation système, banc de tests, ainsi que l'hybridation entre simulation et expérimentation.

Le chapitre 2 permet de répondre aux besoins définis précédemment, en effectuant une revue des langages et outils de simulation couramment disponibles, dans un but de modélisation de systèmes mécatroniques. Il ressort que le langage Modelica peut être considéré comme langage de prédilection de par sa simplicité d'utilisation, sa clarté, ses performances ainsi par des environnements logiciels multiples et potentiellement libres (le langage en lui-même l'étant). Une seconde revue est réalisée mais concernant le matériel supportant la simulation. Diverses solutions allant de l'ordinateur à la puce dédiée, sont abordées. Si l'ordinateur reste l'outil de simulation de prédilection, l'utilisation de puces dédiées est aussi envisageable pour le calcul en temps-réel, ainsi que pour la commande et le contrôle des composants en banc de test et donc l'hybridation entre simulation et expérience.

Le chapitre 3 a présenté une démarche pour la caractérisation des paramètres physiques des systèmes mécatroniques ainsi que des différents moyens existants pour l'identification de ceux-ci. Cette démarche se fait suivant trois points : une observation globale du système permettant l'identification des sous-systèmes qui permet alors, par une recherche de type bibliographique, d'identifier les modèles inhérents aux différents composants les constituant. Il est alors possible d'établir la liste des paramètres à identifier et de pouvoir développer des bancs et protocoles de test pouvant conduire à la détermination des valeurs de ces paramètres. Il s'agit alors d'identification, elles peuvent être directes, par lecture de graphique, par modèle inverse si cette détermination passe par l'inversion d'une équation, ou bien uniquement inverse si il est nécessaire d'établir une procédure d'identification inverse.

Le chapitre 4 détaille la méthode d'identification inverse des paramètres d'un modèle. Ce genre de procédure s'adresse principalement lorsqu'il n'est pas possible de déterminer par des moyens simples les valeurs de paramètres de modèles car, généralement, non-linéaires. La détermination des valeurs se fait en approchant au mieux un relevé expérimental par le résultat d'une simulation numérique à l'aide d'un algorithme d'optimisation. Il existe donc une convergence entre les problèmes d'optimisation et ceux d'identification inverse. Une revue d'algorithme d'optimisation est effectuée et un focus est réalisé sur l'un d'entre eux, afin d'y

apporter des modifications et de les valider, dans un but de réduction du coût de calcul. Cet algorithme, l'essaim de particules, est retenu afin de réaliser les identifications inverses et optimisation sur le sous-système pré-actionneur / actionneur du système mécatronique servant d'exemple.

Le chapitre 5 est l'application de la construction d'un modèle numérique sur le sous-système pré-actionneur / actionneur. Le modèle numérique est créé en utilisant le langage Modelica afin de pouvoir prendre en compte une interaction mécanique / électrique due à la présence d'un composant piézoélectrique. L'identification inverse, en utilisant les essaims de particules, a permis de créer et de déterminer les valeurs des paramètres de ce sous-système avec des résultats satisfaisants. Ce modèle étant validé, sont réalisées des optimisations sur la vitesse de déplacement et l'énergie consommée en travaillant sur les paramètres de différents signaux de commande. Dans ce chapitre, est introduit un nouveau signal de commande uniquement basé sur l'utilisation de spline, et d'un nombre réduit de points pour le définir. On parle alors de *signal libre*. Ce signal permet d'obtenir les meilleurs résultats des optimisations numériques. Cependant, les différents résultats ne se vérifient pas ou peu expérimentalement. Il existe alors un problème de limitation de modèle.

Le chapitre 6 présente le concept d'hybridation d'un processus de simulation numérique avec l'expérimental. La partie expérimentale est constituée du sous-système pré-actionneur / actionneur, et la partie numérique traite le reste du système. Le choix est fait d'utiliser un système embarqué capable de réaliser les mesures et la simulation en temps-réel. Cette implémentation a nécessité une modification mineure du modèle, afin de l'adapter au nouveau support de simulation. Dans cette simulation hybride, il est requis de connaitre la position de l'actionneur pour le fonctionnement de la chaîne d'asservissement. Deux solutions ont été envisagées pour obtenir un estimateur de position : l'utilisation du modèle de l'actionneur et un méta-modèle statistique. La mise en place du modèle étant peu aisée, l'étude d'un méta-modèle est réalisée. Cependant, celui-ci ne permet pas, de façon simple et pertinente, de décrire le fonctionnement de l'actionneur. L'utilisation d'un estimateur n'est donc pas directement envisageable et cela entraîne la nécessité d'utiliser un capteur de position.

Le dernier chapitre reprend les buts des optimisations réalisées dans le chapitre 5, mais en remplaçant la simulation numérique par le système réel. L'algorithme d'optimisation, ainsi que ses paramètres, sont conservés. Etant donné que le résultat de la fonction objectif est issu d'une mesure physique, et que l'actionneur effectue un déplacement rapidement, on effectue alors des optimisations robustes à priori. Comme pour les résultats numériques, le signal libre apparait comme étant le plus performant, aussi bien pour la vitesse que pour l'énergie consommée. Une autre optimisation a été réalisée dans le but de fournir une unique distribution de Weibull pour l'estimateur de position, mais les résultats ne sont pas assez pertinents pour que l'estimateur soit performant, confirmant ainsi la nécessité d'utiliser un capteur de position sur cet actionneur.

Dans cette thèse, la modélisation choisie a montré ses limites dans le cadre des différentes optimisations présentées. Ces défauts peuvent être, en partie, compensés par l'usage de l'expérimentation et par la mise en œuvre de signal de commande efficace.

Cependant, une modélisation plus poussée pourrait permettre de se passer d'un prototype à un stade préliminaire et de ne pas devoir réaliser une optimisation expérimentale.

De même, le choix de l'algorithme utilisé dans cette thèse n'est pas forcément celui le

plus pertinent en termes de coût de calcul / performance. Une étude plus poussée sur les différents algorithmes, surtout méta-heuristiques, est à effectuer. Un travail similaire est à réaliser sur les différents moyens matériels de calcul pour obtenir le meilleur rapport coût de développement / temps de calcul.

Références

[1] Adwin. *Adwin Gold user manual*. 2011.

[2] F. Al-Bender and V. Lampaert and J. Swevers. The Generalized Maxwell-Slip Model : A Novel Model for Friction Simulation and Compensation. *IEEE Transactions on Automatic Control*, 50(2), 2005.

[3] J. Backus. The history of Fortran I, II and III. *IEEE annals of the History of computing*, 20, 1998.

[4] D. Bertsimas and D.B. Brown and C.Caramanis. Theory and applications of Robust Optimization. 2007.

[5] P-A. Bliman and M. Sorine. Easy-to-use realistic dry friction models for automatic control. *Proceedings of 3rd European Control Conference, Rome, Italy*, 1995.

[6] R. Bouc. Modele mathematique d'hysteresis. *Acustica*, 24:16-25, 1971.

[7] Breguet, Jean-Marc. *Actionneurs "stick and slip" pour micro-manipulateurs*. PhD thesis, Lausanne, 1998.

[8] K. Breitung. Asymptotic approximation for multi-normal integrals. *Journal of Engineering Mechanics ASCE*, 110:357-366, 1984.

[9] C.G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms,Journal of the Institute of Mathematics and Its Applications. *Journal of the Institute of Mathematics and Its Applications*, 6:76-90, 1970.

[10] Canudas de Wit, C. and Olsson, H. and Astrom, K.J. and Lischinsky, P. A new model for control of systems with friction. *Automatic Control, IEEE Transactions on*, 40(3):419 -425, 1995.

[11] Particle Swarm Central. *http://www.particleswarm.info/index.html*.

[12] W.G. Cochran. *Sampling Techniques*. Wiley, 1977.

[13] IFAC Technical Committees. Scope on Mechatronics Systems. *http://tc.ifac-control.org/4/2/scope*, 2005.

[14] C.A. Coulomb. *Theorie des machines simples*. 1821.

[15] P. Dahl. A solid friction model. Technical report, The Aerospace Corporation, El

Segundo, CA, 1968.

[16] G.B. Dantzig. Linear Programming under Uncertainty. *Management Science*, 1:197-206, 1955.

[17] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politechnico di Miliano, 1992.

[18] H. Elmqvist. *A Structured Model Language for Large Continuous Systems*. PhD thesis, Lund Institute of Technology, 1978.

[19] S. Fabricius and E. Badreddin. Modelica Library for Hybrid Simulation of Mass Flow in Process Plants. *Modelica'2002*, 2002.

[20] R. Fletcher. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13:317-322, 1970.

[21] L. El Ghaoui and F. Oustry and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9:33-52, 1998.

[22] T. Gingold. *GHDL user guide*. http://ghdl.free.fr, 2010.

[23] D.E. Goldberg. *Genetic Algortihms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[24] D. Goldfarb. A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24:23-26, 1970.

[25] Giuliano Grossi and Federico Pedersini. FPGA implementation of a stochastic neural network for monotonic pseudo-Boolean optimization. *Neural Networks*, 21(6):872 - 879, 2008. Computational and Biological Inspired Neural Networks, selected papers from ICANN 2007, 17th International Conference on Artificial Neural Networks (ICANN).

[26] The Kronos Group. *OpenCl references pages*. 2011.

[27] Jih-Lian Ha and Ying-Shieh Kung and Rong-Fong Fung and Shao-Chien Hsien. A comparison of fitness functions for the identification of a piezoelectric hysteretic actuator based on the real-coded genetic algorithm. *Sensors and Actuators A: Physical*, 132(2):643 - 650, 2006.

[28] J.H. Holland. *Adaptation In Natural And Artificial Systems*. University of Michigan Press, 1975.

[29] R. Hooke and T.A. Jeeves. Direct search solution of numerical and statistical

problems. *Journal of the Association for Computing Machinery*, 8:212-229, 1961.

[30] Mohammad Al Janaideh and Subhash Rakheja and Chun-Yi Su. A generalized Prandtl–Ishlinskii model for characterizing the hysteresis and saturation nonlinearities of smart actuators. *Smart Materials and Structures*, 18(4):045001, 2009.

[31] A. Jardin. *Contribution à une méthodologie de dimensionnement des systèmes mécatroniques : Analyse structurelle et couplage à l'optimisation dynamique*. PhD thesis, INSA Lyon, 2010.

[32] Chih-Cheng Kao and Chin-Wen Chuang and Rong-Fong Fung. The self-tuning PID control in a slider-crank mechanism system by applying particle swarm optimization approach. *Mechatronics*, 16(8):513 - 522, 2006.

[33] J. Kennedy and R. Eberhart. Particle Swarm Optimization. *Internationnal Conference on Neural Networks*, 1995.

[34] B. Kernighan and D. Ritchie. *The C programming language*. Masson, 1983.

[35] S. Kirkpatrick and C.D. Gelatt and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671-680, 1983.

[36] Svein Kvale and Bjorn Haugseter and Odd A. Asbjornsen and Tore Omtveit. On-line optimisation of a particle production process. *Computers & Chemical Engineering*, 25(2-3):465 - 471, 2001.

[37] J. Laffite. *Aide à la conception et au dimensionnement énergétique et dynamique de systèmes mécatroniques par une apporche inverse : application aux chaînes de transmission automobiless*. PhD thesis, INSA, Lyon, 2004.

[38] Larousse. robuste. *http://www.larousse.fr/encyclopedie/nom-commun-autre/robuste/88776*, 2011.

[39] Larousse. simuler. *http://www.larousse.fr/encyclopedie/nom-commun-nom/simulation/91990*, 2011.

[40] H. Ledgard. *Reference manual for the ADA programming language*. Springer - Verlag, 1983.

[41] Cheng-Jian Lin and Hung-Ming Tsai. FPGA implementation of a wavelet neural network with particle swarm optimization learning. *Mathematical and Computer Modelling*, 47(9-10):982 - 996, 2008.

[42] Chiu-Feng Lin and Chyuan-Yow Tseng and Tsai-Wen Tseng. A

hardware-in-the-loop dynamics simulator for motorcycle rapid controller prototyping. *Control Engineering Practice*, 14(12):1467 - 1476, 2006.

[43] R. Longbottom. Dhrystone benchmark results. *http://www.roylongbottom.org.uk*, 2010.

[44] H. Markowitz. Portfolio Selection. *The Journal of Finance*, 7:77-91, 1952.

[45] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44:335-341, 1947.

[46] M. Molga and C. Smutnicki. Test functions for optimization needs. *http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf*.

[47] T. Mori. Mecha-tronics. Technical report, Memo 21.131.01, Yaskawa Internal Trademarck Application, 1969.

[48] M. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33:161-174, 1991.

[49] N. Kyura, H. Oho. Mechatronics - an industrial perspective. *IEEE ASME Transactions on Mechatronics*, 1:10-15, 1996.

[50] L.W Nagel and D.O. Pederson. SPICE (Simulation Program with Integrated Circuit Emphasis). Technical report, University of California, Berkeley, 1973.

[51] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308-313, 1965.

[52] Ngatchou, P. and Anahita Zarei and El-Sharkawi, M.A. Pareto Multi Objective Optimization. *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*, pages 84 -91, 2005.

[53] H.M. Paynter. Analysis and design of engineering systems. *MIT Press*, 1961.

[54] Michael D. Petersheim and Sean N. Brennan. Scaling of hybrid-electric vehicle powertrain components for Hardware-in-the-loop simulation. *Mechatronics*, 19(7):1078 - 1090, 2009. Special Issue on Hardware-in-the-loop simulation.

[55] Övünç Polat and Tülay YIldIrIm. FPGA implementation of a General Regression Neural Network: An embedded pattern classification system. *Digital Signal Processing*, 20(3):881 - 886, 2010.

[56] F. Preisach. Uber die magnetische Nachwirkung. *Zeitschrift fur Physik*,

94:277-302, 1935.

[57] R.W. Daniel, J.R. Hewit. Editorial. *Mechatronics*, 1:1-2, 1991.

[58] A. Rasheed and D. Robinson. Multiscale modelling of urban climate. *Eleventh International IBPSA Conference*, 2009.

[59] Osborne Reynolds. *Collected Papers on Mechanical and Physical Subjects*. Cambridge University Press, 1869 - 1900.

[60] J.M. Rodriguez-Fortun and J. Orus and F. Buil and J.A. Castellanos. General Bond Graph model for piezoelectric actuators and methodology for experimental identification. *Mechatronics*, 20(2):303 - 314, 2010.

[61] John R. Rogers and Kevin Craig. On-hardware optimization of stepper-motor system dynamics. *Mechatronics*, 15(3):291 - 316, 2005.

[62] A. Satelli and M. Ratto and T. Andrs and F. Campolongo and J. Cariboni and D. Gatelli and M. Saisana and S. Tarantola. *Global sensitivity analysis, The primer*. Willey, 2008.

[63] D.F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24:647-656, 1970.

[64] A. Siemers and D. Fritzin and I. Nakhimovski. General meta-model based co-simulations applied to mechanical systems. *Simulation Modelling Practice and Theory*, 17:612-624, 2009.

[65] I.M. Sobol. Sensitivity analysis for non linear mathematical models. *Mathematical Modelling and Computer Experiments*, 1:407-414, 1993.

[66] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research*, 21:1154-1157, 1973.

[67] Richard Stribeck. Die Wesentlichen Eigenschaften der Gleit-und Rollenlager. *Z. Verein. Deut. Ing.*, 46:1341-1348, 1902.

[68] Swevers, J. and Al-Bender, F. and Ganseman, C.G. and Projogo, T. An integrated friction model structure with improved presliding behavior for accurate friction compensation. *Automatic Control, IEEE Transactions on*, 45(4):675 -686, 2000.

[69] Top500.org. Top 500 supercalculators. *http://www.top500.org/list/2010/11/100*, 2010.

[70] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel*

Machines. PhD thesis, Rice University, Houston, Texas, 1989.

[71] Hoon Wee and Yoon Young Kim and Haeil Jung and Gwang Nam Lee. Nonlinear rate-dependent stick-slip phenomena: modeling and parameter estimation. *International Journal of Solids and Structures*, 38(8):1415 - 1431, 2001.

[72] W. Weibull. A statistical distribution function of wide applicability. *Journal of applied Mechanics Transactions ASME*, 18:293-297, 1951.

[73] *AMESim user manual*. LMS Imagine.Lab, 2010.

[74] Reinhold P. Weicker. DHRYSTONE Benchmark Program. *CACM*, 27(10):1013, 1984.

[75] *Dymola user manual*. Dassault Systemes AB, 2010.

[76] IEEE Standard Verilog Hardware Description Language.

[77] IEEE Standard VHDL Language Reference Manual.

[78] ISO / CEI 14882:1998 : The C++ language.

[79] *JModelica user manual*. http://www.jmodelica.org, 2011.

[80] *Matlab user manual*. The Mathworks, 2011.

[81] Y.K. Wen. Method for random vibration of hysteretic systems. *Journal of Engineering Mechanics ASCE*, 102:249-263, 1976.

[82] Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification v. 3.2.

[83] Piotr Wozniak. Preferences in multi-objective evolutionary optimisation of electric motor speed control with hardware in the loop. *Applied Soft Computing*, 11(1):49 - 55, 2011.

[84] www.dspaceinc.com. *dSpace devices*.

[85] www.opal-rt.com. *Opal-RT devices*.

[86] Chia-Feng Yang and Shyr-Long Jeng and Wei-Hua Chieng. Motion behavior of triangular waveform excitation input in an operating impact drive mechanism. *Sensors and Actuators A: Physical*, 166(1):66 - 77, 2011.

[87] A. Zenati. *Modelisation et simulation de microsystemes multi domaines à signaux mixtes : vers le prototypage virtuel d'un microsysteùe autonome*. PhD thesis, Université Joseph Fourier, 2007.

[88] Mécatronique - vocabulaire.

[89] *OpenModelica user manual*. http://www.openmodelica.org, 2011.

[90] T.Y. Jiang and T.Y. Ng and K.Y. Lam. Optimization of a piezoelectric ceramic actuator. *Sensors and Actuators A: Physical*, 84(1-2):81 - 94, 2000.

[91] Quarteroni, Alfio and Sacco, Riccardo and Saleri, Fausto. *Méthodes numériques pour le calcul scientifique*. Springer-France, Paris, FR, 2000.

[92] *Quartus II*. Altera, 2011.

[93] *Scilab user manual*. Digiteo, 2011.

[94] *20Sim 4.1 user manual*. Controllab Products B.V., 2009.

[95] *Simplorer user manual*. Ansoft, 2011.

[96] *SimulationX user manual*. ITI Gmbh, 2010.

[97] *Xilink ISE user manual*. Xilink, 2011.

Annexes

Tests de validation des modifications du PSO

Fonction de Rastrigin :

La fonction de Rastrigin est définie par l'équation suivante :

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i)] \quad (88)$$

La solution se situe au zéro de $\Re^n$ et le résultat de la fonction est aussi zéro. Le test se fait en dimension 20. Les paramètres du PSO sont situés dans la Table 39 et les résultats dans la Table 40.

Nom des paramètres	Valeur
Position inf. $\Re^{20}$	-5.12
Position sup. $\Re^{20}$	5.12
Vitesse inf. $\Re^{20}$	-0.512
Vitesse sup. $\Re^{20}$	0.512
Radius	1e-3
Compteur pour Radius	10
Nombre de Raptor	20
Probabilité de lancement	0.1
Coefficient d'accélération	2
Max. iteration	400
Nombre de particules	20

Table 39 : Paramètres des PSO

	Inertial	Radius	BSG-Starcraft	BSG-Starcraft Radius
Valeur finale moyenne	54.03	57.36	62.29	62.76
Ecart type	16.58	15.61	19.7	18.43
Itération finale moyenne	400	274.07	400	286.8
Ecart type		19.97		23.14

Table 40 : Résultats pour la fonction de Rastrigin

Première fonction de De Jong :

La première fonction de De jong est définie par l'équation suivante :

$$f(x) = \sum_{i=1}^{n} [x_i^2] \quad (89)$$

La solution es située en zéro de $\Re^n$ et le résultat de la fonction est aussi zéro. Le test se fait en dimension 20. Les paramètres du PSO sont définis dans la Table 41 et les résultats dans la Table 42:

Nom des paramètres	Valeur
Position inf. $\Re^{20}$	-5.12
Position sup. $\Re^{20}$	5.12
Vitesse inf. $\Re^{20}$	-0.512
Vitesse sup. $\Re^{20}$	0.512
Radius	1e-3
Compteur pour Radius	10
Nombre de Raptor	20
Probabilité de lancement	0.1
coefficient d'accélération	2
Max. iteration	400
Nombre de particules	20

Table 41 : Paramètres des PSO

	Inertial	Radius	BSG-Starcraft	BSG-Starcraft Radius
Valeur finale moyenne	0.033	0.025	0.02	0.027
Ecart type	0.087	0.062	0.037	0.046
Itération finale moyenne	400	284.86	400	297.13
Ecart type		19.33		18.31

Table 42 : Résultats pour la première fonction de De Jong

Fonction d'Ackley :

La fonction d'Ackley est définie par l'équation suivante :

$$\begin{cases} a = 30 \\ b = 0.2 \\ c = 2\pi \\ f(x) = -a.\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + \exp(1) \end{cases} \quad (90)$$

La solution est située en zéro de $\Re^n$ et à pour valeur zéro. Le test se fait en dimension 20, les paramètres et résultats des PSO étant définis dans les Tables 43 et 44:

Nom des paramètres	Valeur
Position inf. $\Re^{20}$	-32.768
Position sup. $\Re^{20}$	32.768
Vitesse inf. $\Re^{20}$	-3.2768
Vitesse sup. $\Re^{20}$	3.2768
Radius	1e-3
Compteur pour Radius	10
Nombre de Raptor	20
Probabilité de lancement	0.1
Coefficient d'accélération	2
Max. iteration	400
Nombre de particules	20

Table 43 : Paramètres des PSO

	Inertial	Radius	BSG-Starcraft	BSG-Starcraft Radius
Valeur finale moyenne	5.56	4.93	4.99	4.73
Ecart type	2.62	2.77	3.1	2.81
Itération finale moyenne	400	308.9	400	317.1
Ecart type		28.8		26.9

Table 44 : Résultats pour la fonction d'Ackley

Fonction Drop wave :

La fonction Drop wave modélise la géométrie d'une surface liquide soumise à la chute d'un goutte et peut être définie par :

$$f(x_1, x_2) = -\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2} \quad (91)$$

La solution se situe en zéro de $\Re^2$ et le résultat de la fonction est -1. Le problème est donc de dimension 2. Les paramètres et résultats des PSO sont contenus dans les Tables 45 et 46:

Nom de paramètres	Valeur
Position inf. $\Re^2$	-5.12
Position sup. $\Re^2$	5.12
Vitesse inf. $\Re^2$	-0.512
Vitesse sup. $\Re^2$	0.512
Radius	1e-3
Compteur pour Radius	10
Nombre de Raptor	20
Probabilité de lancement	0.1
Coefficient d'accélération	2
Max. iteration	800
Nombre de particules	20

Table 45 : Paramètres des PSO

	Inertial	Radius	BSG-Starcraft	BSG-Starcraft Radius
Valeur finale moyenne	-0.9949	-0.9943	-0.9923	-0.9988
Ecart type	0.0174	0.0183	0.0208	0.0246
Itération finale moyenne	400	244.48	400	272.19
Ecart type		57.69		69.19

Table 46 : Résultats pour la fonction Drop wave