Systèmes mécatroniques à paramètres variables : analyse du comportement et approche du tolérancement

Manel Zerelli

THÈSE

Présentée par Manel ZERELLI

Pour l'obtention du

GRADE DE DOCTEUR

École Doctorale : **École Centrale Paris (ED287)**

Spécialité : **Sciences de l'Ingénieur**

Laboratoire d'accueil : **LISMMA (EA2336)** SUPMECA Toulon

SUJET

Systèmes mécatroniques à paramètres variables
Analyse du comportement et approche du tolérancement

Soutenue le 31 mars 2014

Devant un jury composé de :

M. SORIANO Thierry	**Professeur des Universités SUPMECA**	**Directeur de thèse**
M. MEIZEIL Dominique	**Professeur des Universités ENSIL**	**Rapporteur**
Mme. PRELLE Christine	**Professeur des Universités Université de Technologie de Compiègne**	**Rapporteur**
M. FATHALLAH Raouf	**Professeur des universités ENISO Tunisie**	**Examinateur**
M. BRAC Jean	**Directeur de recherches HdR IFPEN**	**Examinateur**
M. SEPPECHER Pierre	**Professeur des Universités Université de Toulon Var**	**Examinateur**
M. MARE Jean-Charles	**Professeur des Universités INSA Toulouse**	**Invité**
M. HADJ AMOR Hassen	**Docteur Société D2T**	**Invité**

2013 - 2014

Année 2014 École Doctorale Sciences pour l'Ingénieur – ED287

Numéro d'ordre : 2014ECAP0032 École Centrale Paris

Remerciements

Les travaux présentés dans ce mémoire ont été réalisés au Laboratoire d'Ingénierie des Systèmes Mécaniques et des MAtériaux (LISMMA) à l'institut SUPérieur de MECAnique de Toulon (SUPMECA).

Je voudrais remercier toutes les personnes qui m'ont soutenue, de près et de loin, en particulier:

Monsieur Thierry SORIANO, professeur au LISMMA pour m'avoir accueilli dans son laboratoire et soutenu tout au long de la thèse, qui a consacré son temps pour encadrer ces travaux. Merci pour ses conseils scientifiques, l'encouragement et la confiance qu'il m'a accordés.

Je remercie également chacun des membres de jury de thèse. Monsieur Dominique MEIZEL pour avoir être le rapporteur de ma thèse et pour toutes ses remarques instructives, Madame Christine PRELLE pour sa lecture attentive. Je remercie vivement Monsieur Raouf FATHALLAH pour sa confiance et son soutien aux moments de doute. Je remercie également Monsieur Jean BRAC, Monsieur Pierre SEPPECHER et Monsieur Jean-Charles MARE. Et je finis par Monsieur Hassen Jawhar HADJ AMOR pour ses conseils et son soutien.

Je tiens à remercier très sincèrement le personnel administratif et scientifique du LISMMA, en particulier : Madame Pascale AZOU-BRIARD, Madame Lyudmyla YUSHCHENKO, Monsieur Cedric ANTHIERENS, Monsieur Didier GROUX, Madame Sabine SELLIER, Madame Ingrid DESCAREGA, Madame Sabine SEGAL, Madame Lucette ARCHER, Monsieur Olivier TORREMOCHA, Monsieur Christian TOUSSAINT, pour leurs compétences, leur gentillesse et leur disponibilité.

Je remercie également Monsieur Alain RIVIERE et Monsieur Jean Yves CHOLEY pour avoir mis à ma disposition un financement pour ma thèse ainsi que Madame Christel COMPAGNON pour tous ses efforts pour tout ce qui est administratif.

Pour conclure, je remercie et je dédicace cette thèse à toute ma famille, d'abord ma mère et mon père pour leur support irremplaçable et inconditionnel et pour leur soutien sans faille et permanent malgré la distance qui nous sépare. Ils ont été présents pour écarter les doutes, soigner les blessures et partager les joies. Cette thèse est un peu la leur également. Je remercie mon mari Arbi pour son affection et son amour. Je remercie mon amie de toujours Amira pour tout ce qu'elle a fait pour que j'aboutisse. Je remercie également ma belle-mère pour son soutien et son aide. Je n'oublie pas mes frères Amine et Achraf. Merci d'être là tous les jours.

Je remercie tous ceux qui ont cru en moi et mon soutenue avec un mot, un sourire, un conseil. Je vous remercie mes amis, mes cousins et mes chères cousines.

Je dédie ce travail à Lina qui depuis ses premiers jours a entendu parler des variations, des inclusions, et Modelica fut l'un des premiers mots qu'elle a prononcé. Je le dédie aussi à

Alma Sarah ; elle est arrivée à la fin de ce travail et m'a aidée à le clôturer avec son magnifique sourire.

Sommaire

Table des figures

Introduction et problématique

1- Le cadre de l'ingénierie d'un système mécatronique

Les systèmes mécaniques utilisés dans l'industrie, les transports sont de plus en plus associés à des systèmes de pilotage ou de contrôle électronique. Initialement, il s'agissait d'une association ou juxtaposition de fonctions mécaniques et électriques telle qu'on en trouve en électromécanique ; ensuite un important mouvement d'intégration fonctionnelle et volumique s'est produit en particulier pour les systèmes embarqués et la robotique.

Le terme «mécatronique » a été introduit par un ingénieur de la compagnie japonaise «Yaskawa » en 1969. Le terme est apparu officiellement en France dans le Larousse en 2005.
Dans la littérature on trouve différentes définitions du système mécatronique dont :
« La mécatronique est l'intégration synergique de l'ingénierie mécanique avec l'électronique et le contrôle intelligent de calculateurs dans la conception et la fabrication de produits et processus industriels. » [Harashima, 1996]
Parmi les approches de conception liées à de grands domaines technologiques, la mécatronique est définie comme la combinaison synergique du génie mécanique, du génie électronique, de l'automatique et de l'informatique [Craig, 1999].

Isermann [Isermann, 2000] résume les définitions données à la mécatronique. Il estime que toutes les définitions sont d'accord pour dire que la mécatronique est un domaine interdisciplinaire dans lequel les disciplines suivantes agissent ensemble :

- Systèmes mécaniques (éléments mécaniques, machines, mécanique de précision).
- Systèmes électroniques (micro-électronique, électronique de puissance, capteurs et actionneurs).
- Technologie de l'information (théorie des systèmes, automatisation, génie logiciel, intelligence artificielle).

Dans la norme française [XP E 01 – 013] on trouve la définition suivante : « La mécatronique est 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. Cette démarche permet d'obtenir des performances supérieures aux solutions traditionnelles, de réaliser de nouvelles fonctionnalités, et de rendre les produits mécatroniques plus compacts. »

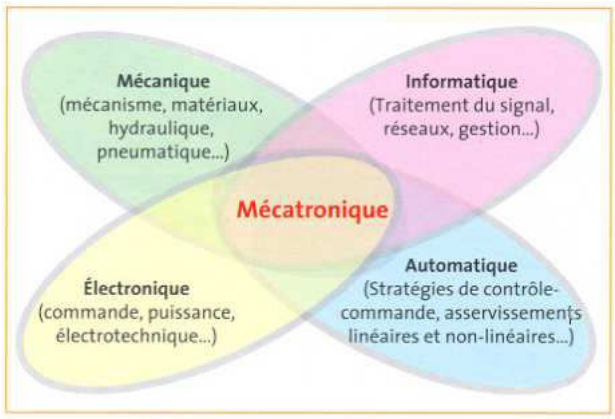


Figure 1 Le domaine interdisciplinaire de la mécatronique

Un système mécatronique est donc caractérisé par l'intégration et l'interaction des différents domaines physiques et technologiques (automatique, mécanique, électronique, électromécanique, optique, thermique, thermodynamique…). Cela rend nécessaire de prendre en compte l'ensemble du cycle de vie dès la phase de conception.

Un composant mécatronique constituant d'un système mécatronique présente un niveau partiel d'intégration mécatronique de point de vue fonctionnel et physique associant mécanique, électronique et permettant le traitement de l'information.

Au total on distingue trois types d'intégration :

- Intégration multi domaine
- Intégration physique 3D
- Intégration fonctionnelle

Nous allons les détailler ;

- **Intégration multidomaine**

La mécatronique est une démarche d'intégration qui combine la mécanique, l'électronique l'informatique, l'automatique, donc par définition multidomaines.

La réussite d'une application mécatronique passe par une vision globale d'un système impliquant différentes disciplines jusqu'à alors séparées, et par la recherche de solutions pluritechniques.
Il faut lors de la conception mécatronique introduire l'ensemble des lois multiphysiques d'un système dans un seul modèle. L'intégration multidomaine des produits mécatroniques implique des couplages forts entre des données complexes à gérer.

- **Intégration physique 3D**

La conception d'un système mécatronique nécessite une intégration des différents métiers ou domaines. Ainsi, la mécanique et l'électronique sont étroitement imbriquées. Il y a quelques années la mécanique et l'électronique étaient séparées, mais l'évolution technologique va dans le sens d'une intégration croissante : plutôt qu'un simple montage d'une électronique à côté d'une mécanique. La mécatronique permet désormais d'inclure l'électronique dans la mécanique, voire même de les fusionner.
On définit aussi l'intégration volumique par rapport aux contraintes de volume pour les systèmes embarqués par exemple où on s'efforce d'intégrer les différentes fonctions (mécanique, électrique et automatique) dans le même volume.

- **Intégration fonctionnelle**

L'intégration fonctionnelle consiste à augmenter le nombre de fonctions réalisées par un ou des composants au sein du même système. Cette intégration consiste à transférer des fonctions réalisées par plusieurs composants dans un seul. Il en résulte une diminution de l'encombrement et du nombre des composants à gérer. Le terme fonction n'a pas le même sens dans les différents domaines ; en mécanique par exemple on parle de fonctions guidage en translation ou en rotation auxquelles on associe des tolérances géométriques et des efforts supportés; en électronique on parle par exemple de gain associé à un gabarit fréquentiel sur une bande de fréquence ou à un temps de montée ; en automatique en parle de fonction de transfert avec des performances de marges de stabilité, de précision ou de temps de réponse. Enfin en informatique le terme fonction couvre un module avec ses entrées et sorties, ses paramètres. Les fonctions et leurs liens avec le temps, plus généralement ce que l'on appelle le comportement doit donc faire l'objet d'une grande attention.

Afin de mesurer les changements de vocabulaire et de culture scientifique et technique nécessaires et les effets attendus de cette démarche d'intégration, un chercheur comme D. Brun-Picard a développé dans [Brun-Picard, 2004] une démarche mécatronique en s'appuyant sur la conception concrète d'un robot « pick-and-place » à haute cadence. Il compare la démarche classique de conception (validation du cahier du charge, choix de l'architecture du robot et de son actionnement, choix des lois de mouvement, dimensionnement des organes mécaniques et des actionneurs, la conception, la fabrication et les essais) et la démarche mécatronique qui fait appel à un « plateau mécatronique » qui réunit les meilleurs spécialistes en construction mécanique, actionnement électrique, commande automatique et informatique industrielle. Cette dernière démarche consiste à avoir les points de vue des différents spécialistes (le mécanicien, l'électronicien, l'automaticien et de l'informaticien) pour faire un choix d'orientation de préconception en se basant sur la synthèse des points de vue. Cette approche permet dès la préconception d'avoir une solution nouvelle qui satisfait toutes les contraintes et apporte même un plus au système tout en réduisant les coûts et les délais.
On peut distinguer deux points dans son approche ;

- l'utilisation de concepts de recherche d'optimalité (commande, volume, masse etc...)et plus généralement de principes variationnels qui sont à la base des progrès constatés vers une solution optimisée.
- Le raisonnement sur des modèles intermédiaires de pré-dimensionnement en relâchant certaines contraintes, conduisant à raisonner sur des paramètres non pas fixes mais pouvant évoluer dans un intervalle.

Nous allons dans la suite nous centrer sur l'intégration fonctionnelle et particulièrement aux modèles de comportement des systèmes mécatroniques en précisant les fonctions supportées, leurs représentions et leur enchainement dans le temps aux cours du fonctionnement des différents modes d'un système.
Notre espace de représentation du comportement sera l'espace vectoriel d'état R^n. Les paramètres évolueront dans des intervalles de R. Les évènements initiant des changements de mode de fonctionnement seront associés à des variables booléennes.

2- Les fonctions et le comportement des systèmes mécatroniques multi modes

Bien que les commandes d'un système mécatronique soient souvent des commandes discrètes, le comportement global est quant à lui constitué de sauts entre des états continus. Ce type de comportement est celui des systèmes dynamiques hybrides (SDH). Il existe un grand nombre de formalismes qui permettent la modélisation des systèmes dynamiques hybrides. En effet, l'intérêt pour ces systèmes a connu un regain ces dernières années à la fois dans la communauté automaticienne et dans la communauté informaticienne. Parmi ces formalismes on peut citer les automates hybrides, les Bond Graph à commutations, les réseaux de Pétri hybrides, les Grafcets hybrides, etc. Les systèmes hybrides se retrouvent dans beaucoup de domaines, comme par exemple l'automobile, l'avionique, la robotique et la génétique.

L'étude des SDH s'intéresse principalement aux classes de problèmes qui n'ont pu être traités avec les approches traditionnelles basées sur une modélisation homogène. Il n'existe pas pour l'instant de théorie globale pour l'étude de ces systèmes, mais plutôt des approches basées sur l'extension de méthodes classiques issues des systèmes continus ou discrets, en vue de couvrir une gamme plus étendue d'applications.

Un système dynamique hybride est un système contenant des variables d'état continues/discrètes et des variables d'état booléennes en interaction.

Définition formelle d'un SDH :

Soit $x \in R^n$, $Q = \{1, 2,..., N\}$ et $u \in R^c$, où n, N et c sont donnés et soit A un sous-ensemble fermé de $R^n xQ$. L'ensemble X représente l'ensemble des états continus et l'ensemble Q représente l'ensemble des états discrets. L'ensemble U représente l'ensemble des commandes continues et la variable t représente le temps.

L'état d'un système est défini à l'instant t par la donnée du couple [x(t), q(t)], où $x(t) \in X$, $q(t) \in Q$ et $u(t) \in U$.

On appelle système hybride, un système dynamique décrit par les équations suivantes :

$$\frac{dx}{dt} = F\big(t, x(t), q(t), u(t)\big)$$

$$[x(t^+), q(t^+)] = G\big(t, x(t), q(t)\big) \quad \text{si} \quad [x(t), q(t)] \in A$$

La fonction F représente la dynamique continue et G représente la dynamique hybride du système.

Pour pouvoir modéliser une plus grande variété de dynamiques continues, le formalisme des automates temporisés a été étendu pour obtenir les automates hybrides.

- **Définition d'un automate hybride :**

Les automates hybrides sont un outil de modélisation des systèmes dynamiques. Cela représente les systèmes qui ont un comportement continu avec des événements discrets. Cet outil est utilisé dans le cas non variationnel. Il sera défini en détail dans le chapitre 3 d'inclusion différentielle impulsionnelle (paragraphe 1.3).

3- Etude des variations paramétriques et des conséquences sur le comportement

On observe d'après le paragraphe précédent, que le comportement d'un système mécatronique, en particulier si il est multimodes, peut être complexe.

On souhaite par ailleurs que la modélisation soit robuste par rapport à une variation de paramètres. En conception on doit être capable d'analyser les conséquences de variations des paramètres. Ces variations (ou appelées incertitudes dans certains cas) peuvent affecter les composants mécaniques ou électroniques.

Elles peuvent être la conséquence de la production, de la fabrication, de l'usage…

3.1 Classification des variations

Les incertitudes peuvent avoir différentes origines ce qui permet un premier classement [Ben Abdessalem, 2011]:

- incertitudes sur le modèle : le modèle est une description approximative des aspects multi-physique du système, plus le modèle est simplifié plus il est imparfait et laisse la place à l'introduction des incertitudes.
- incertitudes numériques : elles proviennent de la résolution numérique des équations et des formulations mathématiques.
- incertitude sur les paramètres : comme les paramètres matériaux (coefficient d'écrouissage, coefficient d'anisotropie, . . .), les paramètres géométriques (épaisseur initiale, longueur, . . .) et les paramètres liés au procédé (trajet de chargement, contact et frottement, . . .).

Si on utilise les équations différentielles pour la modélisation mathématique des systèmes dynamiques, le tolérancement se traduit alors par une variation du coefficient correspondant

au paramètre dans un intervalle. Cette variation peut être exprimée en pourcentage par rapport à une valeur nominale.

On peut classer également les incertitudes paramétriques en deux classes liées à la représentation mathématique de la variation:

- *Variation paramétrique qui n'est pas en fonction du temps*, c'est à dire $v \in [v_1, v_2]$ mais $v'(t) = 0$ (pour $t \in T_1 = [t_1, t_2]$ un intervalle du temps). L'incertitude est invariante avec le temps. On ne connaît pas la valeur initiale de l'incertitude. On a dans ce cas, un échantillon parmi n échantillons. Par exemple pour un système mécanique une évolution dans le temps n'est pas observable $v \in [v_1, v_2]$ on ne connait ni la valeur initiale ni la valeur finale de la variation comme par exemple lors de la production en série, la variabilité dépend de la production (on a n exemplaires).
- *Variation paramétrique qui évolue en fonction du temps*, c'est-à-dire $v \in [v_1, v_2]$ mais $v'(t) \neq 0$. La valeur initiale de l'incertitude est connue. Ceci correspond à l'étude d'un seul exemplaire dont les paramètres évoluent dans le temps $v'(t) \neq 0$. Ce cas peut se manifester lors de l'exploitation par exemple à cause de l'usure. La loi en fonction du temps est inconnue.

On peut également classer les variations selon leur niveau ;

On a commencé à s'intéresser à l'étude du problème des variations paramétriques après avoir constaté son impact sur les performances d'un système. On peut séparer les grandes et les petites variations suivant les cas d'étude. Cette notion est assez relative mais on pourra la simplifier avec des ordres de grandeur : par exemple dans le cas de petite variation on est dans un intervalle de variation de 1% à 20% par rapport à la valeur nominale. Dans le cas de problèmes avec une variation supérieure à 30% de la valeur nominale, on se place dans le cadre de grande variation.

Dans le cas de pré-dimensionnement, on traite des problèmes de grande variation et on veut assurer le fonctionnement du système en respectant les contraintes. L'objectif est le respect des spécifications du cahier des charges et la définition de l'architecture du système. Cela peut faire partie de la phase de conception du système par exemple. Il influence le choix de la technologie (composants) et du logiciel utilisés pour l'étude.

Dans le cas du tolérancement, on traite des incertitudes qui varient dans un intervalle de petite longueur. Notre objectif est de déterminer l'influence de cette variation sur le comportement global du système et de le maintenir dans une zone de fonctionnement désirée.

On peut également classer les variations selon la position dans le cycle de vie du système;

- En phase de conception, on fait face à des grandes variations. En pré-dimensionnement on a un cahier des charges et on doit satisfaire les contraintes.
- En phase de production, on fait face à des petites variations. On a n exemplaires, et on connait juste l'intervalle de variation.
- En phase d'exploitation, on fait face à des petites variations. On a un seul exemplaire dont les paramètres varient dans le temps. On connait la valeur initiale mais on ne connait pas la loi d'évolution.

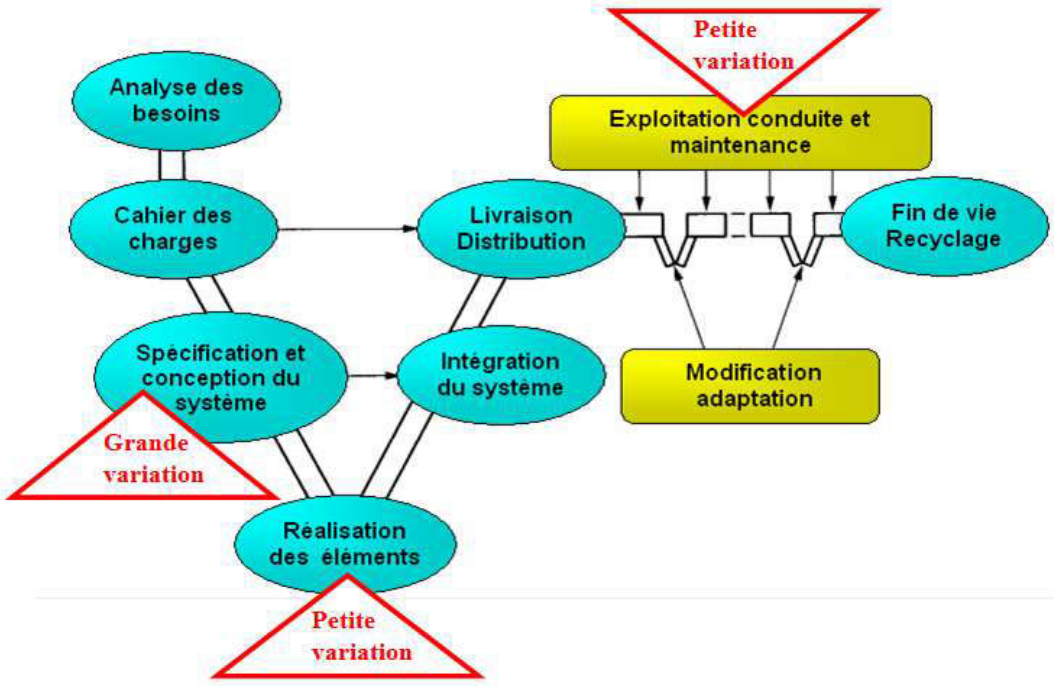


Figure 2 Cycle de vie du système et variations paramétriques

3.2 Méthodes d'approche pour l'étude des variations paramétriques

Dans la première partie de cette thèse, on suppose que la variation paramétrique est associée à un intervalle fixe. L'objectif est d'analyser l'influence de cette variation sur le fonctionnement

et le comportement du système mécatronique. Pour cela il faut trouver les méthodologies qui permettent l'étude des variations et d'exprimer leurs résultats dans l'espace d'état.

Dans la littérature on trouve différentes méthodes d'analyse des conséquences des variations paramétriques.

On peut les classer ;

<u>selon le domaine d'étude</u> :

- domaine temporel
- domaine fréquentiel

<u>selon la méthode d'approche</u> :

- Méthodes stochastiques
- Méthodes déterministes

<u>selon le système à étudier</u> :

- Système continu / hybride
- Système linéaire / non linéaire

<u>selon le schéma de connexion</u>

- En boucle fermée
- En boucle ouverte

<u>selon les critères du modèle</u>

- Modèle explicite
- Modèle implicite

<u>Selon le niveau</u>

- Grande variation
- Petite variation

Dans le cadre du modèle explicite, les variations paramétriques sont explicites. Elles sont généralement en entrée du système. Par contre, dans le modèle implicite, l'incertitude est masquée et il nous faut un outil de transformation de l'entrée pour pouvoir traiter les variations paramétriques.

Un premier apport de notre de travail est d'avoir considéré, analysé et trié les modèles existants de prises en compte de variations paramétriques puis d'avoir sélectionné ceux capables de s'adapter aux modèles de comportement de type automates hybrides associés aux systèmes mécatroniques.

Dans un premier temps on fait un état de l'art de différentes méthodes de traitement des variations paramétriques. Mais en considérant un système mécatronique le choix va se limiter à quelques méthodes précises. En effet, les méthodes stochastiques et probabilistes jusqu'à présent utilisées pour l'étude des incertitudes ne résolvent pas notre problème. Le résultat donné par ces méthodes ne représente que 5 à 20% du résultat des méthodes déterministes [Raczynski, 2006]. Il faut alors une méthode d'étude déterministe et qui fournit un résultat précis. L'inclusion différentielle est une méthode déterministe d'étude de variation. Raczynski a utilisé cette approche dans le domaine de l'automatique et de la simulation. On a adopté cette méthode pour les systèmes mécatroniques à comportement purement continu. On a développé l'algorithme de résolution et on a montré qu'on peut l'utiliser pour étudier les variations paramétriques.

Le comportement des systèmes dynamiques hybrides se caractérise par un fonctionnement continu et présentant des sauts discrets dans le temps. La méthode la plus adaptée selon nous pour l'étude des variations paramétriques des systèmes hybrides est l'inclusion différentielle impulsionnelle. Cette approche a été introduite par Aubin [Aubin, 2002]. On a repris cette approche et on a développé des algorithmes de résolution dans le cadre des systèmes mécatroniques hybrides.

A ce stade, l'objectif est de déterminer l'influence de ces incertitudes sur le fonctionnement et le comportement du système. On procède à une démarche d'analyse et de simulation. On dispose des intervalles de variation, et on procède à une étude du système soumis à ces variations et à une simulation des résultats pour obtenir le domaine atteignable.

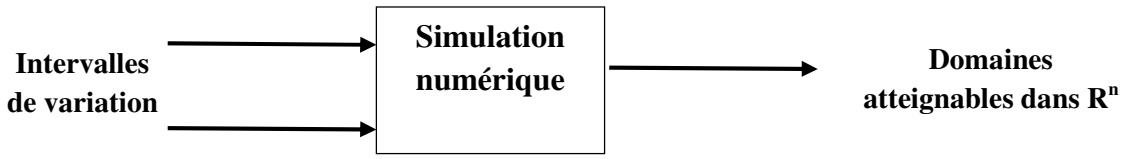


Figure 3 Démarche d'analyse et de simulation

Un second apport de notre travail a consisté à écrire des algorithmes et à choisir un outil de simulation numérique permettant une implémentation des modèles mathématiques sélectionnés précédemment pour construire et exhiber concrètement les domaines atteignables d'un modèle de comportement d'abord continu pur, puis hybride.

4- Maitrise des variations paramétriques et tolérancement sur le comportement

Dans la partie suivante on introduit le tolérancement mécatronique. On définit une propriété (spécification) d'un système mécatronique par un sous ensemble S de l'espace d'état auquel le vecteur d'état doit appartenir. (On l'appellera au chapitre 6 la zone de fonctionnement désiré).

Par ailleurs, on évalue également dans l'espace d'état le domaine atteignable par le vecteur d'état lorsque le temps et les paramètres évoluent.

On considère les relations entre ces deux ensembles. On choisit un outil métrique pour mesurer les distances entre eux. On peut alors grâce à cette métrique construire des stratégies de convergence entre spécification et domaine atteignable de telle sorte que ce dernier soit inclus dans le sous ensemble S.

Définition du tolérancement mécatronique

Nous posons la définition suivante ;

Le tolérancement du comportement d'un système mécatronique est l'activité du concepteur consistant à définir un sous ensemble S de l'espace d'état R^n, dans lequel le

vecteur d'état doit appartenir de sorte qu'une certaine propriété globale du système est respectée.

Les dimensions de ce sous ensemble S sont les résultats de distribution par le concepteur d'intervalles de R associés aux valeurs des paramètres physiques de nature différentes (mécanique, électrique, …).

Remarque : ces intervalles de paramètres peuvent être choisis en pré-conception, ou subis en production ou en exploitation par dérive dans le temps.

Le tolérancement peut être vu comme l'action effectuée pour cerner une variation dans un certain domaine ou intervalle en vue d'assurer une fonction ou un comportement. L'incertitude peut être définie comme une variation de la valeur nominale, elle est mesurée et souvent subie par le concepteur.

Le tolérancement concerne alors indirectement une variation d'une cotation géométrique, d'une dimension d'une pièce mécanique (jeu) ou de la valeur caractéristique d'un composant électronique comme par exemple la valeur de la résistance qui n'a pas une valeur fixe mais qui varie dans un intervalle dit intervalle de tolérancement.

Dans le cas du tolérancement mécanique on associe à une fonction une condition géométrique par exemple un jeu qui est distribué sur l'ensemble des pièces. Il faut alors, que le système assure ses fonctions en tenant compte du ce jeu. Selon Samper [Samper,2007], pour le domaine mécanique « Le tolérancement s'inscrit, tout naturellement dans une modélisation cinématique d'un assemblage en petits déplacements. Le mécanisme est vu, dans un premier temps, sous la forme d'un assemblage rigide sans défauts de formes, puis dans un second temps comme un assemblage élastique sans défauts de formes, puis comme un assemblage rigide avec défaut de forme. Quant à l'étude d'un assemblage élastique avec défauts de formes, nous esquissons seulement des pistes que nous comptons explorer.

Dans la mécanique des structures, nous utilisons les deux types les plus classiques d'analyses : l'analyse de structures élastiques en statique et l'analyse modale avec les hypothèses des petites perturbations. La première permet d'intégrer les déformations élastiques des composants à l'analyse de leur assemblage et de leur fonctionnement. La seconde ouvre le paramétrage traditionnel des géométries à la prise en compte des défauts de formes. ».

Dans le cas du tolérancement électronique, on travaille dans un certain gabarit temporel ou fréquentiel dans le domaine monodimensionnel. L'objectif est de réaliser un réglage en boucle fermée et d'assurer la stabilité ou un certain gain.

Dans le cadre du tolérancement mécatronique on travaille dans l'espace d'état R^n, le tolérancement est sur le comportement du système. Il est de nature différente car il peut avoir une origine électrique, mécanique, automatique. Le tolérancement global est distribué sur l'ensemble des composants. Ce qui rend l'étude du tolérancement mécatronique avec des variations paramétriques de natures différentes plus abstraite.
En synthèse sur le tolérancement on constate qu'on a donc deux points importants à résoudre :

- Modéliser les effets des variations paramétriques sur le comportement d'un système mécatronique et ce en adoptant une méthode théorique qui permet de modéliser l'effet de cette incertitude sur le fonctionnement des systèmes mécatroniques continus et hybrides et de choisir l'outil d'ingénierie adéquat à la simulation et l'évaluation des zones de l'espace d'état R^n quand les paramètres varient.
- Exprimer une propriété correspondante à un tolérancement en caractérisant des zones de l'espace d'état et vérifier que le comportement du système continuera de respecter cette propriété en vérifiant l'appartenance du vecteur d'état à ces zones.

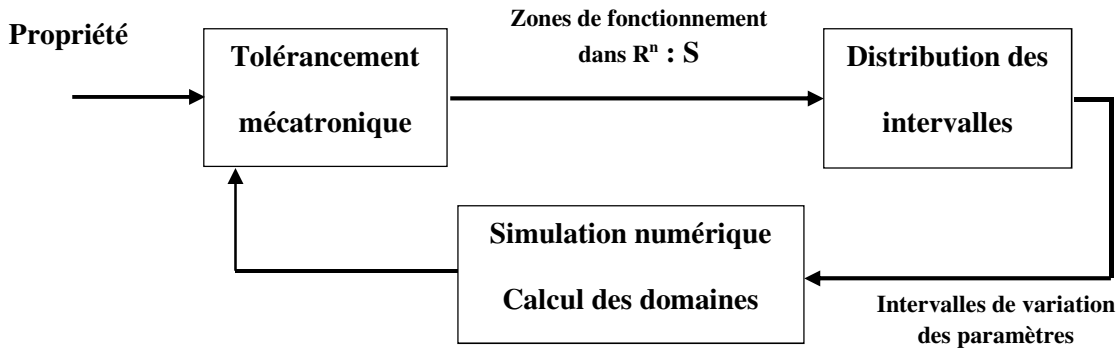


Figure 4 La démarche du tolérancement bouclé

Avec s un sous ensemble de l'espace d'état R^n

Organisation du document

Dans le premier chapitre on commence un état de l'art des méthodes existantes de prise en compte de variation suivant les critères énoncés plus haut dans la classification du paragraphe 3.2. On commence par une méthode non dynamique à savoir la méthode des plans d'expériences qui permet lorsque des expériences sont possibles, des analyses de sensibilité à des variations de paramètres pertinentes pour des réponses non nécessairement dépendantes du temps. Ensuite, on présente une méthode industrielle qui se base sur Openturns et Modelica. On introduit ensuite les méthodes dynamiques. On commence par les méthodes stochastiques. Puis, on définit les méthodes fréquentiels de H∞ et la méthode de μ-analyse qui permettent l'étude de variations paramétriques dans le domaine fréquentiel. Ensuite, on présente la méthode d'analyse par intervalle qui permet de cerner le résultat, de calculer et de manipuler des intervalles. Ces méthodes présentent des avantages et des limites mais elles ne seront pas retenues dans le cadre de ces travaux pour des raisons qu'on va expliquer dans le chapitre 1. On finit le chapitre 1 par présenter la méthode des inclusions différentielles et la méthode des inclusions différentielle impusionnelle.

Dans le deuxième chapitre nous choisissons d'abord d'étudier un système mécatronique purement continu avec des paramètres variables. Pour résoudre le problème d'influence de variation paramétrique on a choisi la méthode des inclusions différentielles. On détaille cette approche déterministe qui consiste à trouver l'ensemble possible des solutions suite à une variation des paramètres ; On reprend précisément son formalisme. C'est un outil mathématique développé par Raczynski [Raczynski, 2006] et utilisé dans la simulation des systèmes dynamiques. Ayant constaté qu'aucun algorithme complet ou schéma numérique de réalisation satisfaisant n'existent dans la littérature, on développe entièrement un algorithme de résolution d'une inclusion différentielle. On l'applique à un exemple de système.

Dans le troisième chapitre nous élargissons notre champ de recherches et nous nous intéressons à l'étude des systèmes hybrides qui comportent à la fois une évolution continue et discrète. Nous présentons les automates hybrides comme un outil qui permet la modélisation des systèmes dynamiques hybrides dont les paramètres sont fixes. Pour modéliser des systèmes hybrides munis d'incertitude (avec des paramètres variables) nous choisissons l'inclusion différentielle impulsionnelle. Nous identifions ses différents ensembles appliqués à

exemple mécatronique et nous développons entièrement des algorithmes de résolution d'inclusion différentielle impulsionnelle.

Au cours du quatrième chapitre on compare deux outils logiciels de simulations utilisables en ingénierie à savoir Mathematica et Modelica sur notre exemple mécatronique et nous présentons une implémentation de nos algorithmes développés aux chapitres 2 et 3 en utilisant Mathématica.

Dans le chapitre 5, nous présentons d'autres approches de résolution comme les automates hybrides à invariants polyèdre et les inclusions différentielles polygonales et nous comparons notre algorithme de résolution avec l'algorithme de la méthode de Heun qui présente une approche implicite de résolution des inclusions différentielles.

Dans le chapitre 6, nous reprenons l'ensemble des résultats et outils pour la simulation et le calcul des domaines atteignables obtenus dans les chapitres précédents. Nous présentons le tolérancement dans l'espace d'état. Nous exprimons les propriétés des systèmes par des domaines dans cet espace liés à des zones de fonctionnement désiré. Nous introduisons un outil de métrique topologique pour la mesure de ces ensembles et de leurs distances. Nous proposons des organigrammes conformément à la mise en œuvre du schéma général bouclé de la figure 4 vu précédemment.

Chapitre 1 : Etude du tolérancement mécatronique et méthodes d'approche

Introduction :

Dans ce premier chapitre on présente un état de l'art de notre problème. On définit quelques méthodes d'approche parmi les plus répondues pour l'étude et l'analyse des variations paramétriques. On commence par une méthode statistique : la méthode des plans d'expérience.

On introduit les méthodes stochastiques, puis, les méthodes d'approche dans le domaine fréquentiel et enfin, des méthodes temporelles comme la méthode d'analyse par intervalle, l'inclusion différentielle et l'inclusion différentielle impulsionnelle. On définit chacune de ces méthodes et on montre ses avantages et ses limites dans l'étude des systèmes mécatroniques à paramètres variables.

1- La méthode des plans d'expérience

La méthode des plans d'expériences cherche à déterminer une relation entre la réponse (la grandeur physique étudiée) et les facteurs ou les grandeurs physiques modifiables par l'expérimentateur et sensées influer sur les variations de la réponse [Schimmerling et al 1998].

La construction d'un plan d'expériences consiste à extraire du domaine expérimental, un nombre suffisant de combinaisons particulières afin d'estimer, avec une incertitude à la fois minimale mais aussi homogène, les inconnues du modèle (additif ou polynomial) tout en respectant au mieux les contraintes techniques et économiques de l'étude.

La méthode des plans d'expériences peut être utilisée dans deux types d'investigations [Rabier, 2007]:

• Les études de criblage ou screening : déterminer les facteurs influents (qui ont une influence considérable statiquement sur la variation de la réponse) et identifier les interactions de facteurs qui auront une influence significative sur la réponse. On recherche *pourquoi* la réponse varie.

• Les études de surface de réponse : calcul des variations en fonction des facteurs et des interactions influents (déterminé par criblage ou screening). Déterminer *comment* la réponse varie.

Il s'agit ici de facteur ou paramètre qui évoluent indépendamment du temps et non selon une loi de comportement continue ou hybride. Cette méthode requiert un nombre important d'essais réels ce qui n'est pas applicable dans notre cas d'étude. En effet, les facteurs variables sont des composants du système mécatronique et on ne dispose pas des plans d'expérience mais d'une évolution du fonctionnement du système sous ses variations.

La méthode des plans d'expérience est une méthode statique, elle ne couvre pas les modèles dynamiques. Dans le cadre de ce travail, cette méthode ne sera pas prise en compte pour l'étude de l'influence des variations paramétriques sur les systèmes mécatroniques .

2- Méthode industrielle : Modélisation d'incertitudes avec Openturns et Modelica

Dans les applications industrielles on trouve des données incertaines (délivrées par les capteurs par exemple). Ces données sont des mesures et non pas des paramètres. Ils peuvent influencer de manière importante le fonctionnement général du système et affecter ainsi sa performance, sa précision. Pour cela l'étude de ces incertitudes s'avère importante. EDF fait partie de ces industriels qui traitent des problèmes d'incertitudes pour améliorer ses centrales existantes (augmenter la durée de vie à 60 ans par exemple) et construire des nouvelles centrales.

EDF et IFPEN (Institut Française du Pétrole Energies Nouvelles) travaillent sur le développement d'une librairie pour la modélisation des centrales et des systèmes sous Modelica [Site Web Modelica] dans le but de créer un environnement pour la modélisation et l'étude d'incertitudes. Dans [Bouskella D, and all 2011] on présente différentes techniques pour le traitement d'incertitudes :

- *La conciliation des données* : ce qui consiste à réduire au minimum l'intervalle d'incertitude de variable (trouver une variable de conciliation de l'état physique qui se rapproche le plus de la valeur réelle de l'état physique).
- *La propagation des incertitudes* : propager la source des incertitudes à travers le modèle du système. EDF et R&D ont présenté une méthode composée de quatre étapes détaillées dans l'article [Global methodology of incertainty study (Site web openturns)]
- Etape A : ***Identification*** des sources d'incertitudes et ***spécification*** du cas d'étude y=h(x, d) avec :
    - y : vecteur des variables
    - h : le modèle
    - x : vecteur des variables d'entrée dont on va étudier l'incertitude
    - d : vecteur des variables d'entrée certain

En posant un critère déterministe (déterminer le domaine possible de y en déterminant le minimum et le maximum des variables de y) et un critère probabiliste (y est vecteur au hasard).

- Etape B : ***la quantification*** des sources des incertitudes. Quand on choisi le critère déterministe, alors le domaine possible de chaque élément de x doit être déterminé. Selon D.

Bouskella, quand on choisit le critère probabiliste, les sources de l'incertitude sont traitées comme étant des composants d'un vecteur aléatoire x (dans le contexte des auteurs x est un vecteur d'entrée). Pour chaque élément x_i du vecteur, la distribution de la probabilité doit être évaluée. Ainsi que la dépendance entre deux éléments de x doit être évaluée sous forme de coefficients de corrélation [Bouskella D, and all 2011].

- Etape C : ***la propagation de l'incertitude***. Le critère déterministe est de déterminer le domaine de variation de y en précisant son minimum et son maximum ; si h est monotone en x alors cela est facile sinon on doit avoir recours aux méthodes d'optimisation. Le critère probabiliste est de proposer des méthodes d'approximation et des méthodes de discrétisation pour l'estimation (exemple l'échantillonnage robuste).

- Etape D : de ***classer les sources d'incertitude*** selon un critère déterministe et un probabiliste. L'outil utilisé pour la simulation est OPEN TURNS.

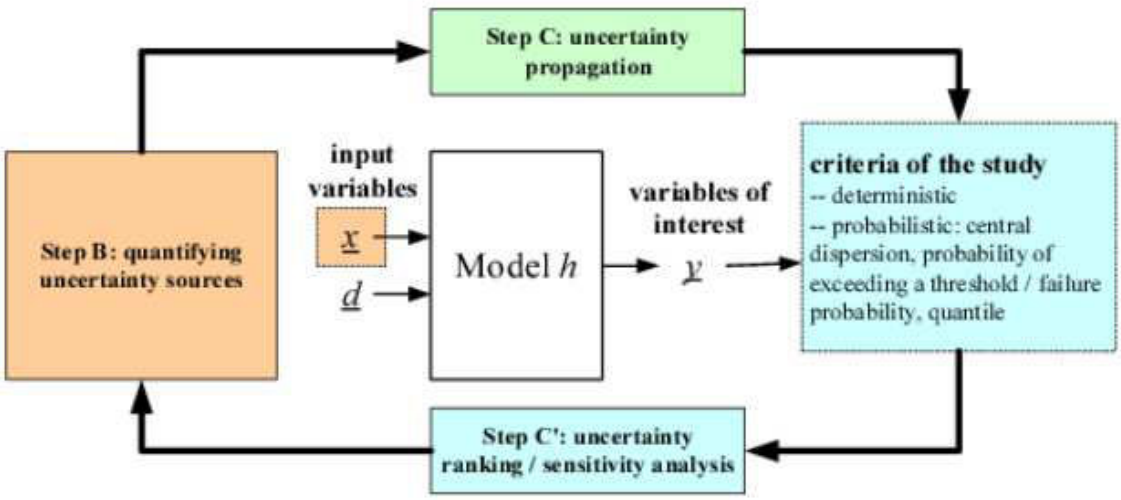


Figure 5 Les différentes étapes de la méthode de propagation des incertitudes développée par EDF et R&D [Bouskella D, and all 2011]

Pour l'étude d'incertitude des systèmes complexes IFPEN (Institut Française du Pétrole Energies Nouvelles) utilise des méthodes comme le krigeage. Cette méthode d'interpolation spatiale permettant une estimation linéaire basée sur l'espérance mathématique et la variance de la donnée. En effet, le krigeage se base sur le calcul, l'interprétation et la modélisation du variogramme, qui est une appréciation de la variance en fonction de la distance entre données. Cette méthode permet d'éliminer les valeurs improbables (comme par exemple dans le cas de météo on élimine les températures « absurdes ») [Gratton, 2002] .

Dans [Bouskella D, and all 2011] on trouve aussi une proposition d'extension du langage Modelica pour l'étude des incertitudes, en introduisant différentes méthodes de distribution probabilistes.

Cette méthode présentée par EDF est une méthode statistique qui consiste en l'utilisation des lois de la probabilité. Elle présente l'avantage de créer un bloc en extension du langage Modelica qui permet de modéliser un système complexe tout en tenant compte des variations. L'inconvénient de cette méthode est le manque de développement sur le modèle h car elle ne permet pas d'y entrer les variations paramétriques. Un autre inconvénient est que cette méthode n'est pas déterministe et peut négliger des valeurs ayant une influence importante sur le comportement du système. C'est pour ces raisons qu'on ne développera pas d'avantage cet outil dans le cadre de cette thèse.

3- Méthode d'approche stochastique

Dans un système mécatronique réel les composants ne sont pas fixes ; ils sont confrontés à des problèmes d'usure (dans le temps), défaut de fabrication, jeux… tout cela nous mène à chercher une méthode pour essayer de prévoir le fonctionnement du système et de l'optimiser et à trouver une modélisation adaptée au problème. Parmi les méthodes utilisées pour l'étude d'incertitude, on trouve les approches probabilistes aléatoires.

Les méthodes stochastiques permettent alors d'étudier les comportements en moyenne, en modélisant de façon probabiliste les parties d'un système qu'on ne souhaite pas ou qu'on ne peut pas décrire en détails. Les variations paramétriques sont réparties au hasard à l'intérieur de l'intervalle de variation. On ne dispose pas de loi de distribution.

Les méthodes stochastiques fournissent des outils pour déduire les distributions de probabilités des grandeurs de sortie importantes, en fonction de celles des entrées et du modèle du système. Elles permettent ensuite d'utiliser ces informations pour prendre des décisions appropriées.

Le traitement du signal (filtre de Kalman, traitement de la parole et de signaux physiologiques, traitement d'images) repose en grande partie sur des méthodes stochastiques. En imagerie spatiale, par exemple, ces techniques sont utilisées pour éliminer le bruit des informations brutes captées par les télescopes et ainsi identifier automatiquement les corps stellaires.

Les résultats fournis par ces méthodes sont souvent peu précis, définis par des probabilités, et obtenus en boucle fermée (Kalman). Ces méthodes ne seront pas détaillées dans le cadre de cette thèse où on considèrera essentiellement des méthodes d'étude déterministes.

4- Méthodes fréquentielles

Dans cette partie on se propose d'évaluer l'intérêt pour notre objectif de formalismes et de méthodes issus de l'étude générale de la robustesse. En effet, nous nous sommes appuyés avant d'explorer cette voie sur l'approche de G. Duc et S. Font [Duc, 1999, P53]. Selon eux, comme nous le verrons plus loin, après avoir regroupé les incertitudes paramétriques dans une matrice nommée Δ(s), ils affirment : « L'étude de la robustesse consiste à chercher à garantir une propriété particulière (par exemple la stabilité) pour un ensemble d'incertitudes Δ(s) ». Dans notre problème il ne s'agit pas de garantir explicitement la stabilité, mais suivant la forme de l'ensemble S représentant la propriété cela pourrait être étudié, et il s'agit plutôt d'explorer si les formalismes proposés peuvent nous aider en respectant nos hypothèses de travail.

4.1- La norme H∞

- Définitions générales pour un système linéaire invariant en boucle ouverte

Définition des valeurs singulières :

Considérant un système linéaire invariant avec un vecteur d'entrée e(t) et un vecteur de sortie s(t) de dimensions respectives m et p. Soit G(s) sa matrice de transfert. En réponse à une excitation harmonique $e(t) = Ee^{j\omega t}$, $E \in C^m$, la sortie du système s'écrit :

$$s(t) = G(j\omega)Ee^{j\omega t} \qquad (1.1)$$

Pour un système monovariable, on définit à partir de cette relation le gain du système à la pulsation ω par le module $|G(j\omega)|$. Dans le cas multivariable, on utilise la notion de valeurs singulières, définies comme les racines carrés des valeurs propres de G(jω) multipliées par sa transconjuguée :

$$\sigma_i(G(j\omega)) := \sqrt{\lambda_i(G(j\omega)G(-j\omega)^T)} = \sqrt{\lambda_i(G(-j\omega)^T G(j\omega))} \qquad (1.2)$$

- pour un système monovariable il existe une seule valeur singulière

$$\bar{\sigma}(G(j\omega) = \underline{\sigma}(G(j\omega)) = |G(j\omega)| \qquad (1.3)$$

Avec $\bar{\sigma}$ (G) la plus grande valeur singulière et $\underline{\sigma}$ la plus petite et $|G(jw)|$ le gain du système à la pulsation ω.

- Pour un système multivariable.

$$\underline{\sigma}(G(j\omega)) \le \frac{\|G(j\omega)Ee^{j\omega t}\|_2}{\|Ee^{j\omega t}\|_2} \le \bar{\sigma}(G(j\omega) \qquad (1.4)$$

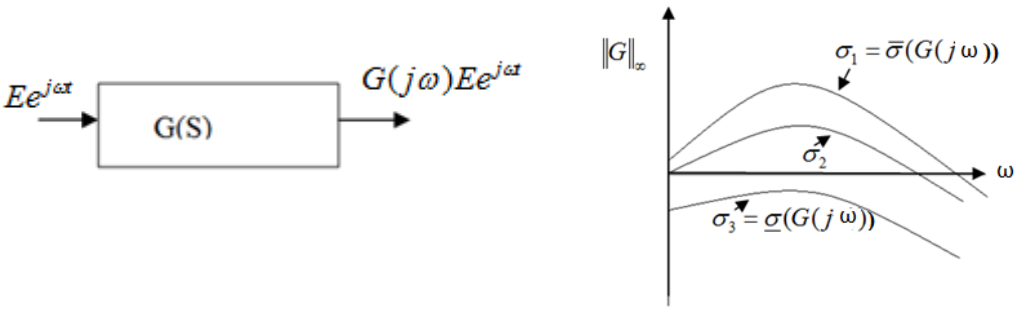


Figure 6 Valeurs singulières et norme H∞ d'une matrice de transfert [Duc, 1999]

Les valeurs singulières constituent une généralisation aux systèmes multivariables de la notion de gain. Dans la figure 6 ci-dessus on donne une représentation dans le plan de Bode. Pour un système multivariable, le gain à une fréquence donnée dépend du vecteur complexe E, et sera compris entre les valeurs singulières inférieure et supérieure.

4.2- Définition de la norme H∞ : $\| \quad \|_\infty$

Un système linéaire invariant est défini par la représentation d'état suivante [Duc, 1999] :

$$\begin{cases} \dot{x}(t) = Ax(t) + Be(t) \\ s(t) = Cx(t) + De(t) \end{cases} \qquad (1.5)$$

La matrice du transfert est : $G(s) = C(SI - A)^{-1}B + D$ (1.6)

Définition :

$$\|G(s)\|_\infty := \underset{w \in R}{Sup}\, \bar{\sigma}(G(j\omega)) \qquad (1.7)$$

$\|G(s)\|_\infty$ est la valeur la plus élevée du gain du système sur l'ensemble des pulsations (pour un système monovariable , c'est la valeur la plus élevée de $|G(j\omega)|$).

Pour calculer les valeurs qui encadrent le gain, on énonce la propriété suivante :

Propriété [Duc, 1999]

Soit un réel positif $\gamma \succ \bar{\sigma}(G)$, alors $\|G(s)\|_\infty \prec \gamma$ si et seulement si la matrice hamiltonienne H_γ n'a pas de valeurs propres sur l'axe imaginaire.

$$H_\gamma = \begin{bmatrix} A - BR^{-1}D^T C & -\gamma BR^{-1}B^T \\ \gamma C^T S^{-1} C & -A^T + C^T DR^{-1}B^T \end{bmatrix} \quad ; \quad \begin{matrix} R = D^T D - \gamma^2 I \\ S = DD^T - \gamma^2 I \end{matrix}$$

Cette propriété permet de chercher de façon explicite le plus petit majorant possible des gains en calculant les valeurs propres de Hγ, ou d'affiner le majorant en utilisant cette propriété de façon itérative. Cette méthode devient impraticable pour les systèmes d'ordre supérieur à 1 [Duc, 1999].

4.3- Problème standard de H∞ en boucle fermée

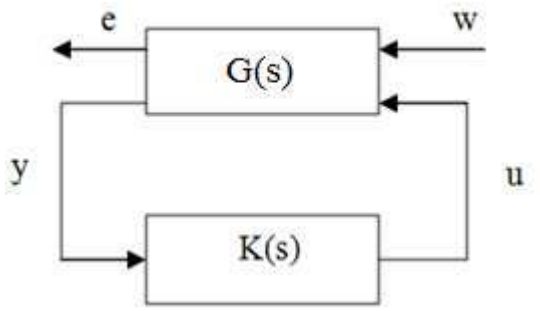


Figure 7 Problème H ∞ standard

La matrice du transfert G(s) modélise les interactions dynamiques entre 2 ensembles d'entrée et 2 ensembles de sortie : le vecteur w représente les entrées extérieures (signaux de référence, perturbation, bruit) ; le vecteur u représente les commandes ; les signaux e sont choisis pour caractériser le bon fonctionnement de l'asservissement et y représente les mesures disponibles pour élaborer la commande.

K(s) est le correcteur en boucle fermée.

On effectue une partition de la matrice G(s) de façon cohérente avec les dimensions de w,u,e,y.

$$G(s) = \begin{bmatrix} G_{ew}(s) & G_{eu}(s) \\ G_{yw}(s) & G_{yu}(s) \end{bmatrix}$$

La matrice du transfert entre w et e du système bouclé est appelée : Transformation Fractionnaire Linéaire (LFT) inférieure**.**

$$E(s) = F_l(G(s), K(s)))W(s)$$
$$F_l(G(s), K(s)) := G_{ew}(s) + G_{eu}(s)K(s)(I - G_{yu}(s)K(s))^{-1}G_{yw}(s). \qquad (1.8)$$

La synthèse H∞ du correcteur est définie par le problème suivant :

Problème H∞ standard : G(s) et $\gamma \succ 0$ étant donnés, déterminer K(s) qui stabilise le système bouclé et assure $\left\|F_l(G(s), K(s))\right\|_\infty \prec \gamma$.

On a commencé par l'étude de H∞ dont la définition est $\left\|G(s)\right\|_\infty = \underset{w \in R}{Sup}\, \overline{\sigma}(G(j\omega))$ qui est la valeur la plus élevée du gain du système sur l'ensemble des pulsations. Pour un système monovariable, c'est la valeur la plus élevée de $\left|G(jw)\right|$. Cette norme est utilisée pour introduire des fonctions de pondération utilisées comme des filtres en boucles fermée. Elle sert à calculer un correcteur en manipulant des concepts fréquentiels. Elle permet de prendre en compte des objectifs de stabilité et de robustesse. On peut utiliser cette norme pour répondre aux exigences du cahier des charges sur le gain ou le temps de réponse par exemple en modifiant les valeurs des filtres ou du majorant γ.

Dans le cas de variations paramétriques, on veut que le système ait une certaine réponse par rapport aux entrées et à la perturbation, la méthode H∞ peut présenter un outil de calcul dans le domaine fréquentiel.

Cette norme nous servira pour la méthode présentée dans le paragraphe suivant : la méthode µ-analyse.

4.4- Méthode de µ-analyse

Etude des systèmes en boucle fermée

La mise en équation d'un système physique nécessite de prendre en compte les variations paramétriques. Il convient alors de chercher la stabilité et la performance en dépit de différentes incertitudes.

Pour appliquer la méthode de µ analyse, on doit transformer les boucles fermées de façon à se ramener au schéma d'interconnexion standard : M(s) -Δ.

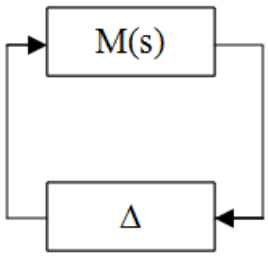


Figure 8 Schéma d'interconnections standard pour la **µ-**analyse

Cela consiste à ramener toutes les variations paramétriques (incertitudes) dans une matrice Δ et à modéliser le système dans la matrice M(s).

Si on ignore la structure de la matrice d'incertitude Δ(s), on cherche la plus grande valeur admissible de sa norme en utilisant la norme H∞, on obtient alors une condition de robustesse de la stabilité simple mais très pessimiste [Duc, 1999]. Par contre dans le cas où on prend en compte cette structure, on aura un résultat précis et cela nous ramène à calculer les valeurs singulières structurées.

Valeur singulière structurée

La définition de la valeur singulière structurée (VSS) : µ(M($j\omega$) à la pulsation ω est comme suit :

$$\frac{1}{\mu(M\omega)} = \inf \left\{ \bar{\sigma}(\Delta) \Big/ \det(I - \Delta M(j\omega) = 0 \right\} \qquad (1.9)$$

$$\mu(M\omega) = 0 \quad si \quad \det(I - \Delta M(j\omega) \neq 0 \forall\ \Delta \in E_{\Delta} \qquad (1.10)$$

La valeur singulière structurée est une extension de la valeur singulière (définie dans la partie H∞) permettant de tenir compte de la structure de la matrice d'incertitude Δ. M (jω) est le gain complexe du système à la pulsation ω. La valeur singulière structurée VSS est notée également μ.

Définition

La μ-analyse consiste à évaluer un réel positif μ, le plus faible possible qui soit un majorant de $\mu_{\Delta}(M(s))$ sur l'axe imaginaire. La procédure usuelle consiste à choisir un ensemble suffisamment dense de valeur ω et à calculer une borne supérieure de $\mu_{\Delta}(M(j\omega))$pour chaque valeur choisie, on prend alors pour μ la valeur la plus élevée obtenue.

On note que $\mu(M(j\omega))$ est choisie égale à 0 s'il n'existe pas de variations paramétriques de modèle.

$1/\mu(M(jw))$ est interprétée comme la taille de la plus petite variation paramétrique δ qui mène un pôle de la boucle fermée sur l'axe imaginaire en ±jω. La marge de robustesse k_{max} est obtenue par :

$$k_{\max} = \min_{\omega \in [0,\infty]} 1/\mu(M(j\omega)) \qquad (1.11)$$

On manipule le VSS $\mu(M(jw))$ plutôt que la marge de stabilité multivariable $1/\mu(M(jw))$, car la VSS ne prend pas de valeur infinie de fait de l'hypothèse d'une boucle fermée nominale asymptotiquement stable alors que la marge de stabilité multivariable peut devenir infinie s'il n'existe pas de perturbation du modèle capable de déstabiliser la boucle fermée.

$1/\mu$ est la plus petite norme de la matrice d'incertitudes Δ capable de déstabiliser le système. Elle est donc la marge de robustesse. La robustesse est garantie si μ<1.

La matrice d'incertitude Δ peut intégrer trois formes d'incertitudes :

- Des scalaires réels ce qui correspond à des incertitudes sur les paramètres.
- Des scalaires complexes qui proviennent des incertitudes sur les dynamiques.
- Des blocs complexes.

E_Δ l'ensemble des matrices Δ considérées, est défini par :

$$E_\Delta = \{\Delta = diag\,(\delta_1 I_{r1}, \ldots, \delta_s I_{rs}, \varepsilon_1 I_{c1}, \ldots, \varepsilon_t I_{ct}, \Delta_1, \ldots, \Delta_F)\}$$
$$ou \quad \delta_k \in \Re\ ,\ \varepsilon_k \in C\ et\ \Delta_k \in C^{m_k \times m_k}$$

Le calcul de μ peut être difficile car sa complexité croît de manière non polynomiale en fonction de la taille du problème. C'est pour cela que dans certains cas il y a recours à un encadrement de valeurs supérieures et valeurs inférieures.

4.5- Robustesse de la stabilité

La robustesse consiste à assurer que le système conserve certaines de ses qualités et à assurer son fonctionnement avec le changement des conditions de travail. Il peut être soumis à des incertitudes sur les paramètres du système. On peut avoir deux cas de figures (voir classification des variations dans le chapitre Introduction et problématique) :

- Les paramètres sont incertains mais constants et leurs vitesses de variation sont négligeables (exemple lors de la production).
- Les paramètres varient en fonction du temps et leurs vitesses de variation ne sont pas négligeables (exemple lors de l'exploitation).

Dans le cadre de paramètres incertains mais constants, on fait face alors à un modèle linéaire incertain et on peut appliquer les méthodes linéaires telles que la méthode de μ-analyse. Par contre, si les paramètres présentent une variation en fonction du temps, les méthodes linéaires ne sont plus exactes et le système entre dans le cadre des systèmes non linéaires, et les méthodes de type Lyapunov répondent mieux au problème [Laroche, 2007] .

Lorsque la matrice Δ(s) incorpore des incertitudes réelles, on en déduit notamment que le système reste stable pour chaque valeur d'incertitude inférieur à μ^{-1} ; on affecte ainsi un intervalle admissible à chaque paramètre. L'extraction des incertitudes n'est pas toujours immédiate [Duc, 1999].

Lorsque Δ(s) incorpore des incertitudes réelles, $\mu_\Delta(M(j\omega))$ peut être une fonction discontinue de la fréquence, par exemple lorsque les variations paramétriques sont telles qu'un pôle traverse l'axe imaginaire à une fréquence fixe.

Dans le cas d'incertitudes réelles, on peut appliquer une procédure de régularisation qui consiste à remplacer chaque paramètre réel δi par un paramètre complexe $\delta i + \alpha \xi i \quad , \quad \delta i \in R \quad , \xi i \in C$

4.6- Robustesse de la position des pôles

On suppose que la perturbation structurée Δ de modèle ne contient que des scalaires réels. La μ-analyse consiste à calculer la VSS $\mu_{\Delta}(M(j\omega))$ le long de l'axe imaginaire lorsqu'il est traversé par un des pôles de la boucle fermée, pour en déduire ensuite la marge de robustesse k_{max} .

Au lieu de considérer le demi-plan gauche, l'idée est de considérer plus généralement d'autres types de régions Ω du plan complexe. Si on suppose que les pôles de la boucle fermée nominale appartiennent à la région Ω, on calcule alors la VSS sur la frontière de Ω et on déduit la taille de la plus petite variation paramétrique du modèle.

Pour la robustesse de la position des pôles, on doit garantir que les pôles du système bouclé restent dans un domaine Ω.

Pour le domaine de stabilité Ω correspondant à –a<0 de la partie réelle, on teste les valeurs singulières sur ∂Ω.

De nombreuses régions Ω sont à priori envisageables, comme le disque unité dans le cas d'une boucle fermée discrète où on peut choisir un secteur tronqué dans le plan complexe dans le cas des systèmes continus.

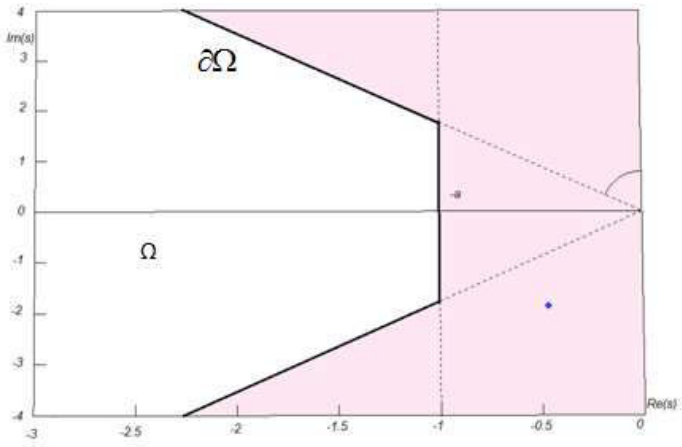


Figure 7 Etude de la robustesse d'un placement de pôles [Duc, 1999]

Cet outil est utilisé pour assurer la stabilité en boucle fermée. En effet, la position des pôles est fonction de la variation des paramètres.

Synthèse

L'objectif de l'étude de système à variation paramétrique est d'avoir la possibilité de prévoir son évolution pour mieux satisfaire les contraintes. La méthode de la μ-analyse fournit un cadre général pour l'étude de la robustesse d'une boucle fermée soumise à deux types d'incertitudes de modèle. Cette méthode se base sur le calcul de la VSS en utilisant les outils de la norme H∞. Ce calcul ne peut pas être exact c'est pour cela qu'on cherche à le cerner par une borne supérieure et une borne inférieure. Ceci peut être une réponse à des exigences dans le cahier des charges par exemple pour la robustesse de la marge de module. La robustesse de la position des pôles, établie en utilisant la μ-analyse, permet de définir une certaine zone de stabilité du système ; cette notion sera reprise est définie dans le chapitre « inclusion différentielle impulsionnelle » dans le cadre de la théorie de viabilité et de construction du noyau de viabilité et développée au dernier chapitre par un exemple illustratif.

La méthode de la μ-analyse utilise la méthode de H∞ et prend une forme matricielle qui peut être utilisée dans le domaine temporel. Il s'agit d'une méthode mixte.

L'objectif de cette méthode est l'analyse de la robustesse en boucle fermée, alors que notre objectif est d'analyser le comportement des systèmes mécatroniques en boucle ouverte dans l'espace d'état. Bien que cette méthode présente un outil d'études des variations paramétriques, elle ne répond pas à tous nos exigences sur la nature du système (elle n'est pas efficace pour les systèmes non linéaires et en boucle ouverte) et sur le degré de précision des solutions (une solution bornée dans l'espace des incertitudes alors que nous cherchons la solution dans l'espace d'état).

5- Méthodes d'approche temporelle

5.1- Approche algébrique : la méthode d'analyse par intervalles

Le calcul par intervalle est un cas spécial de calcul des ensembles, et la théorie des ensembles nous fournit les bases de l'analyse par intervalle [Le Bars 2011].

Définition d'un intervalle :

Un intervalle réel [x] est un sous ensemble connecté de R. Dans le cas des intervalles ouverts on doit conserver également la notation [x].

La limite inférieure d'un intervalle [x] notée par $\underline{x}$ est définie par :

$$\underline{x} \overset{\Delta}{=} \sup\left\{a \in R \cup \{-\infty, \infty\} \middle| \forall x \in [x], a \leq x\right\} \qquad (1.12)$$

La limite supérieure d'un intervalle noté $\overline{x}$ est définie par :

$$\overline{x} \overset{\Delta}{=} \inf\left\{b \in R \cup \{-\infty, \infty\} \middle| \forall x \in [x], x \leq b\right\} \qquad (1.13)$$

Le centre ou le milieu d'un intervalle non vide [x] est défini par :

$$mid(x) = \frac{\underline{x} + \overline{x}}{2} \qquad (1.14)$$

Définition d'un tube

Un tube [x] (t), avec un pas de temps $\delta > 0$ est une fonction d'intervalles vectoriels constante dans $[k\delta; k\delta + \delta]$; $k \in Z$. Le pavé $[k\delta; k\delta + \delta]$ [x] est appelé la $k^{ième}$ tranche du tube [x] (t) et sera noté [x] (k). On peut donc voir un tube comme une union de tranches.

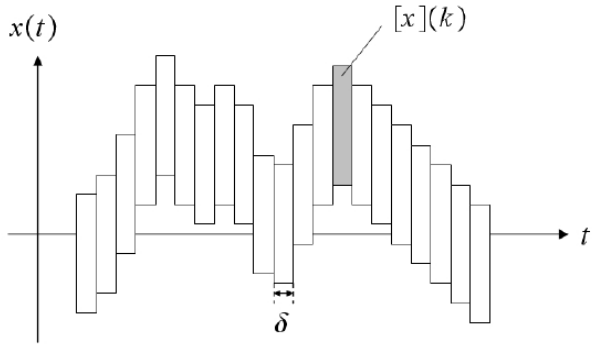


Figure 8 Un tube d'intervalle [x](t) avec un pas de temps **δ(t)** [Le Bars 2011]

Contraction et propagation

La propagation des intervalles des contraintes (ICP : Interval Constraint Propagation) est une technique qui utilise la combinaison de deux notions plus générales : la propagation des contraintes et le calcul par intervalles, pour résoudre des équations contenants des intervalles.

L'objectif de la méthode de propagation est de réduire au maximum les domaines contenant les variables recherchées tout en gardant la solution. Ceci peut être fait en utilisant la notion de contracteur définie dans [Jaulin and Chabaret , 2008] et [Chabaret and Jaulin ,2009].

La contraction est l'action du contracteur à réduire la taille d'un domaine contenant une variable. Le contracteur est un algorithme construit à partir d'une contrainte pour contracter les domaines des variables impliqués dans cette contrainte.

Algorithme de bissection

L'algorithme de bissection consiste à effectuer un arbre de recherche où chaque nœud est un intervalle représentant le domaine courant des variables. On effectue un test sur cet intervalle via l'arithmétique des intervalles. Si le test est faux, alors il le sera pour tous les points de l'intervalle, et peut être donc supprimé. S'il est vrai alors il sera aussi vrai sur tous les points de l'intervalle et on peut le ranger dans la liste des solutions [Chabert,2007] et [Le Bars , 2011]. Dans le cas où le test pour certains points de l'intervalle est faux et il est vrai pour d'autres points de l'intervalle alors il est coupé en deux sous-intervalles. On parle de bissection (branch) pour le découpage en sous intervalles, et d'évaluation (bound) pour le test. Le résultat de cet algorithme est sous forme de dessins en 2D de sous-pavages de solution, de sous-pavages non solution et des sous-pavages indéterminés.

Il existe différents algorithmes de bissection comme par exemple SIVIA (pour l'algorithme de base) et d'autres optimisés et utilisés en combinaison avec des contracteurs existants comme 3BCID ou STRANGLE. Parmi les logiciels utilisés on trouve PROJ2D ou QSOLVE .

La méthode d'analyse par intervalle présente plusieurs avantages. On peut avoir des algorithmes polynomiaux pour calculer des intervalles précis qui contiennent toutes les valeurs possibles des variables aléatoires grâce au processus de *propagation*. Les méthodes de propagation ne nécessitent pas que les équations soient linéaires. Lorsque le modèle n'est plus

valide (par exemple, si l'erreur sur les données est gaussienne non bornée), les méthodes de propagation par intervalles retournent un ensemble vide.

Les méthodes ensemblistes par intervalles sont utilisées dans différents domaines dans l'automatique et la robotique, comme l'estimation d'état et de paramètre [Raissi et al. 2004] , le contrôle [Lydoire and Poignet, 2003], [Vinas, et al. 2006], la localisation des robots [Meizel et al. 1996], le SLAM (Simultaneous Localization And Mapping) dans le cas des robots roulant terrestres [Drocourt et al., 2005], la détection de boucle dans une trajectoire de robot sous-marin [Aubry et al. 2011], analyse topologique d'espace de configuration [Delaoune et al. 2006].

Cependant, les méthodes par intervalles présentent aussi des inconvénients : la surestimation (ou conservatisme) des erreurs qui présente un inconvénient majeur qui correspond au fait d'avoir des intervalles très larges qui donnent une information non précise pour être utilisable [Jaulin et al, 2001]. Un autre inconvénient est que le temps de calcul peut être parfois exponentiel par rapport à la dimension des pavés (vecteurs d'intervalle) si des méthodes de bissection sont utilisées.

Dans le cadre de nos recherches on s'intéresse aux variations paramétriques et leur influence sur le comportement du système. La méthode d'analyse par intervalle effectue un pavage de l'espace d'état. Elle permet de caractériser l'ensemble de solutions sous forme d'un sous-pavage. Les parties de l'espace de recherche incompatibles avec les mesures sont exclues. On obtient ainsi un sous-pavage solution qui garantit d'inclure l'ensemble des solutions du problème.

Cette méthode encadre la solution dans R^n de manière large, alors que, nous verrons plus loin que les méthodes d'inclusion différentielle et d'inclusion différentielle impulsionnelle donnent exactement le domaine atteignable qui englobe toutes les solutions possibles du système. Nous proposons dans le chapitre 6 une perspective d'encadrement de solution plus fin mesurable en utilisant la distance de Haudorff.

5.2- Inclusions différentielles

Le fonctionnement d'un système mécatronique dépend du comportement de ses composants. Une manière générale de modéliser ces systèmes est d'utiliser des équations différentielles.

Mais, il faut noter qu'une variation infinitésimale de l'un des composants peut affecter considérablement le fonctionnement global du système. Car si un paramètre de la partie physique du système présente une incertitude alors le coefficient de l'équation différentielle correspondant varie aussi dans un intervalle. Un des outils mathématiques bien approprié pour modéliser ce type de systèmes est l'inclusion différentielle [Aubin, 1984] .

Les inclusions différentielles représentent une généralisation des équations différentielles avec des paramètres variables. La solution donnée par les inclusions différentielles est le « domaine atteignable » au lieu d'une seule trajectoire. Ce concept est très connu dans le domaine de la commande et plus précisément dans la commande optimale mais n'est pas encore exploité dans la modélisation et la simulation [Raczynski, 2006].
La forme générale de L'ID est comme suit :

$$\frac{dx}{dt} \in F(t, x(t)), \qquad x(0) \in X_0 \qquad (1.15)$$

Avec $x \in R^n$ et F une application de $R \times R^n$ dans des parties de R^n et X_0 est l'ensemble initial.
Pour montrer le lien entre les inclusions différentielles et le domaine de commande il suffit de considérer le système dynamique suivant :

$$\frac{dx}{dt} = f(t, x(t), u(t)), \quad x(0) = x_0, \quad u(t) \in C(t, x(t)) \qquad (1.16)$$

Avec u(t) variable de commande, et l'application F se définit comme suit :

$$F(t, x(t)) = \left\{ z = f(t, x, v) \,\middle|\, v \in C(t, x) \right\} \qquad (1.17)$$

v est une variable appartenant à un domaine de variation C. Cette variable peut être soit une commande soit un paramètre du système d'état.
Raczynski a utilisé ce concept théorique pour la simulation avec v correspond à une commande variable du système. Dans le cas général, le domaine de variation C correspond au domaine de variation de la commande. Cette notion est connue dans le domaine de la commande et plus précisément dans le domaine de la commande optimale.

Dans les méthodes dites « primitives » on discrétise le temps et on sélectionne des points dans le domaine F à chaque pas du temps avec une densité de distribution uniforme, ce qui rend difficile d'estimer la forme du domaine atteignable. Alors que, la méthode utilisée par

Raczynski pour la simulation des systèmes dynamiques en utilisant les inclusions différentielles, consiste à générer le contour du domaine atteignable.

En effet, la solution des ID est un domaine dans l'espace temps-état. Il s'agit d'un ensemble qui inclut toutes les trajectoires possibles du système. Cependant, trouver les limites de ce domaine atteignable n'est pas évident.

Pour calculer le domaine l'auteur définit et cherche à optimiser un hamiltonien particulier à son approche ; comme nous le verrons précisément au chapitre 2, cet hamiltonien n'utilise pas spécifiquement de loi de commande comme entrée mais fait appel aux paramètres variables du système comme degrés de liberté pour l'optimisation ; l'optimum correspond alors à l'enveloppe de toutes les trajectoires possibles du système.

Dans [Raczynski 2006] Raczynski développe un pseudo algorithme pour le solveur des inclusions différentielles défini comme suit :

Début

0. Initialiser la fonction de distribution de la variable v égale à la fonction de densité uniforme ce qui correspond à démarrer avec un pas d'incrémentation uniforme pour v, et initialiser x=x0.
1. Générer le vecteur adjoint initial P avec des conditions initiales $p(t_0)$ choisies arbitrairement .
2. Intégrer l'équation dynamique du système et l'équation de l'Hamiltonien avec une commande qui le maximise à chaque pas d'intégration en utilisant un algorithme d'optimisation.
3. Sélectionner les régions de densité minimale de points. cela peut être obtenu en détectant les zones de concentration faible de trajectoires dans le domaine atteignable ou sur sa projection 2D.
4. Modifier la fonction de distribution en augmentant la probabilité de distribution au voisinage des zones (points) de densité maximale et en raffinant la recherche de trajectoires dans les zones de concentration faible.
5. S'il y a assez de trajectoires enregistrées « Arrêter », sinon aller à l'étape 1.

Fin

Le pseudo algorithme ci-dessus représente un outil déterministe de modélisation et de simulation d'effet d'incertitudes (les différentes étapes de cet algorithme seront détaillées

dans le chapitre 2 : les inclusions différentielles). Cet outil est basé sur des concepts de commande optimale qui illustre l'ensemble accessible comme l'ensemble d'extrémités de solution à un instant t en faisant varier la commande [Trélat E, 2005].

Dans notre cas d'étude on va se baser sur cet algorithme pour traiter un problème de variation paramétrique qui concerne des éléments du vecteur d'état. Cette variation résulte d'une incertitude des composants physiques représentée par une variation du coefficient de l'équation différentielle correspondant.

Dans la résolution du problème de commande optimale on a besoin des équations supplémentaires pour déterminer le vecteur adjoint p, appelées conditions de transversalité. Mais pour l'ID on n'a pas besoin de ces équations pour résoudre le problème aux limites. Il suffit de choisir les conditions initiales arbitrairement. En effet cet algorithme fournit après la première itération un point sur le domaine atteignable ce qui résout le problème de valeur finale. Donc quel que soit la condition initiale, on résout le problème puisque notre objectif est de trouver les trajectoires qui parcourent le domaine atteignable.

Cet algorithme nécessitera par la suite le recours à une méthode d'optimisation pour la maximisation de l'Hamiltonien comme la méthode de la plus grande pente « steepest descent ».

5.3- Inclusions différentielles impulsionnelles

Les inclusions différentielles impulsionnelles représentent un modèle hybride qui contient une évolution continue et une évolution discrète [Aubin, 2002].

Définition 1 : Le temps de trajectoire hybride $\tau = \{I_i\}_{i=0}^{N}$ est une séquence finie ou infinie d'intervalle de lignes réelles telle que

- Pour i<N $I_i = [\tau_i, \tau_i^{'}]$
- Si N<∞ alors $I_N = [\tau_N, \tau_N^{'}]$
- Pour tout i , $\tau_i \leq \tau_i^{'} = \tau_{i+1}$

τ_i est le temps de la réalisation de la transition discrète.

Définition 2 : L'inclusion différentielle impulsionnelle est une collection H(X,F, R, J) avec :

- X : vecteur d'espace de dimension fini
- F : $X \rightarrow 2^X$ l'inclusion différentielle $\dot{x} \in F(x)$
- R : $X \rightarrow 2^X$
- $J \subseteq X$ l'ensemble des transitions forcées.

Définition 3 : Le fonctionnement d'une inclusion différentielle : H(X, F, R, J) est un couple (τ,x) qui consiste en temps de trajectoire hybride et une application x : τ → X qui satisfait :

- Une évolution discrète pour tout $x(\tau_{i+1}) \in R(x(\tau'_i))$
- Une évolution continue. Si $\tau_i < \tau'_i$, x(.) est solution de l'inclusion différentielle impulsionnelle $\dot{x} \in F(x)$ sur l'intervalle $[\tau_i, \tau_i']$ commençant par $x(\tau_i)$ avec $x(t) \notin J \ \forall \ t \in [\tau_i, \tau'_i[$

  Hypothèse :

Si $J \subseteq R^{-1}(x) \quad et \ si \quad J \ est \ ouvert \quad (I = X \ / \ J \ \ fermé)$
$F(x) \cap T_1(x) \neq \Phi \quad \forall \ x \in I \ / \ R^{-1}(x)$ (1.18)

Cette hypothèse montre que pour $x \in X$, l'évolution continue est possible car x est soit dans J, soit il est forcé à entrer dans J tout au long des solutions des inclusions différentielles. Donc, la transition est possible.

Les inclusions différentielles impulsionnelles sont une extension des inclusions différentielles et des systèmes à temps discrets dans un Vecteur d'Espace de Dimension Finie (V E D F).

L'inclusion différentielle $\dot{x} \in F(x)$ dans un(V E D F) X peut être considérée comme une inclusion différentielle impulsionnelle H= (X, F, R, J) avec $R(x) = \Phi \ \forall \ x \in X \quad et \quad J = \Phi$.

Un système à temps discret $x_{k+1} \in R(x_k)$ peut être considéré comme une inclusion différentielle impulsionnelle H=(X,F,R ,J) avec

$$F(x) = \{0\} \ \forall \ x \in X \quad et \quad J = X \quad (\frac{dx}{dt} = 0, \quad x = cst) \qquad (1.19)$$

(aucun point de X n'a une évolution continue) .

Nous proposons dans le tableau qui suit une petite synthèse comparative des différentes méthodes.

Méthode	Avantages	Inconvénients
Méthode des plans d'expérience	Déterminer une relation entre la réponse et les facteurs variables qui peuvent influent la réponse.	Nécessite un nombre important d'essais. Méthode statique qui ne couvre pas les modèles dynamiques.
Méthode industrielle Openturns	Modéliser un système complexe sous Modelica tout en tenant compte des variations.	Méthode non déterministe. Elle risque de négliger des paramètres qui ont une influence sur le comportement du système .
Méthode stochastique	Déterminer l'influence des variations sur le comportement du système.	Les paramètres sont répartis au hasard, on risque de négliger des variations influentes sur le comportement du système
Méthode fréquentiel : Commande H∞ Méthode de µ-analyse	H∞ est une norme pour les systèmes linéaires et un outil de calcul. La µ-analyse permet l'étude de la robustesse du système en boucle fermée.	Méthodes non efficaces pour les systèmes linéaires en boucle ouverte. La µ-analyse se base sur le placement des pôles et non sur le vecteur d'état.
Méthode d'analyse par intervalle	Donne un pavage de l'espace d'état	Elle donne des informations peu précises malgré un temps de calcul important.
✓ **Inclusion différentielle**	Etude des systèmes continus à paramètres variables Le résultat est un domaine atteignable qui englobe toute les solutions dans l'espace d'état.	Nécessite l'utilisation des algorithmes d'optimisation pour le calcul de pas de variation .

✓ **Inclusion différentielle impulsionnelle**	Etude des systèmes hybrides à paramètres variables Solution déterministe dans l'espace d'état en boucle ouverte pour des paramètres explicites et liés au tolérancement ayant des petites ou grandes variations.	Permet l'étude des systèmes hybrides avec une variation des paramètres et non des variations sur les performances. Peu de développement pour la génération de la commande en boucle fermée.

Tableau 1.1 : Les différentes méthodes d'étude des variations paramétriques

Conclusion

On a présenté un état de l'art des différentes méthodes d'étude d'incertitude. Ces méthodes sont complémentaires car chacune présente des avantages et des limites.

On a commencé par présenter une méthode statistique d'étude de variation à savoir la méthode des plans d'expérience. Il s'agit d'une méthode statique qui ne couvre pas les modèles dynamiques. Cette méthode nécessite un nombre important d'essais, il ne sera pas possible de l'appliquer à l'étude des systèmes mécatroniques et donc elle ne sera pas retenue dans le cadre des travaux de cette thèse. Ensuite, On a présenté une autre méthode statistique utilisée par EDF et IFPEN qui consiste en l'utilisation des lois de la probabilité. On crée un bloc en extension du langage Modelica qui permet de modéliser un système complexe en tenant compte des variations. Cette méthode n'est pas déterministe et peut négliger des valeurs ayant une influence importante sur le comportement du système.

Après, on s'est intéressé aux méthodes dynamiques d'approche. On a commencé par la définition d'une méthode stochastique. Cette méthode se base sur un échantillonnage au hasard des points d'intervalle de variation. Le résultat obtenu par cette méthode n'est pas très précis et donc ne satisfait pas les critères des systèmes mécatroniques.

Ensuite on a présenté les méthodes d'étude de variation paramétrique dans le domaine fréquentiel. Les méthodes fréquentielles présentent un autre aspect d'étude du tolérancement. En effet, la commande H∞ permet de résoudre de nombreux problèmes de commande sur différents types de systèmes. Elle permet de régler le compromis entre les objectifs de stabilité et de performance grâce au réglage des filtres de pondération pour un système linéaire en boucle fermée.

La commande H∞ est bien connue pour ses propriétés de robustesse, mais il faut savoir que ces propriétés dépendent de la complexité des systèmes et des incertitudes introduites. Dans certaines applications, la commande H∞ permet d'améliorer les performances nominales du système mais ne permet pas d'élargir le domaine de stabilité.

La méthode de μ-analyse se base sur le calcul des valeurs singulières structurées à laquelle on peut attribuer une borne supérieure et une borne inférieure. Cette méthode permet également l'analyse de la robustesse du système, la robustesse de la stabilité…Il s'agit d'une méthode mixte (fréquentielle et temporelle) qui analyse la robustesse des systèmes en boucle fermée.

On a présenté après une approche algébrique à savoir la méthode d'analyse par intervalle. Cette méthode permet le calcul et les manipulations des intervalles pour effectuer un pavage de l'espace d'état. Elle donne les solutions du système sous forme d'un sous pavage en excluant les solutions non compatibles.

Les différentes méthodes citées ci-dessus ont leurs avantages mais présentent des limites par rapport à leur utilisation dans le domaine de la mécatronique. En effet, le résultat de l'étude de variation paramétrique obtenu par ces méthodes est imprécis, risque d'exclure des solutions non compatibles et donc de rejeter des zones de fonctionnement du système. Cela viole les propriétés du système liées au tolérancement mécatronique (définies au chapitres introduction et problématique et détaillées au chapitre 6). Cela nous a emmené à étudier des méthodes déterministes comme les inclusions différentielles et les inclusions différentielles impulsionnelles.

Les inclusions différentielles traitent la variation des paramètres dans le domaine du comportement purement continu. Elle donne un domaine atteignable qui représente l'enveloppe de toutes les solutions du système suivant les variations.

Pour les systèmes dynamiques hybrides on a choisi le formalisme des automates hybrides pour la modélisation. Ceci présente un outil pour la modélisation des systèmes continus/ discrets.

On a introduit les inclusions différentielles impulsionnelles définies par [Aubin, 2002]. Ce formalisme utilise à la fois des automates hybrides qui incluent des inclusions différentielles dans ses états. Avec cette méthode on peut étudier des systèmes hybrides à paramètres variables (présentant des incertitudes).

Dans le domaine temporel ou fréquentiel l'objectif est commun, on veut étudier l'impact des incertitudes et assurer le meilleur fonctionnement du système.

Alors que dans le domaine temporel on cherche à voir l'impact du tolérancement sur le fonctionnement général du système, en fréquentiel on cherche à déterminer la stabilité du système face à des incertitudes. Dans le temporel on a eu recours à différents formalismes de modélisation comme les automates hybrides ou les inclusions différentielles impulsionnelles, en fréquentiel on a eu recours à des méthodes comme la commande $H\infty$ ou la μ-analyse où on fait adapter le système : l'extraction de la matrice des incertitudes, le calcul des valeurs singulières structurées… Dans le temporel le résultat est un domaine atteignable qui englobe

toutes les solutions possibles, en fréquentiel on détermine le domaine de stabilité du système suivant des critères appliqués comme le positionnement des pôles ou la robustesse de la marge de stabilité. Dans l'approche du tolérancement on part des variations paramétriques pour étudier le comportement, alors que dans l'approche de la μ-analyse on a le comportement et on en déduit les variations paramétriques.

Dans le cadre du travail de la thèse on s'est orienté vers l'utilisation des méthodes déterministes dans le domaine temporel pour l'étude des incertitudes paramétriques, comme l'inclusion différentielle pour les systèmes purement continus et les inclusions différentielles impulsionnelles pour les systèmes dynamiques hybrides.

L'inclusion différentielle impulsionnelle satisfait tous les critères cités dans le chapitre « Introduction et problématique » page 17 paragraphe 3.2. Il s'agit d'une méthode déterministe pour l'étude des systèmes hybrides et non linéaires dans un domaine d'étude temporel R^n en boucle ouverte. Elle permet l'étude du tolérancement avec petite ou grande variation. Le résultat est un domaine atteignable qui englobe toutes les trajectoires possibles.

Chapitre 2 : Les inclusions différentielles

Introduction

Dans ce chapitre, on considère un système mécatronique purement continu. Comme on l'a constaté après la présentation des différentes méthodes d'étude de variations paramétrique, la méthode la plus adaptée à la résolution de notre problème est la méthode des inclusions différentielles. On commence par un cas particulier des inclusions différentielles à savoir la commande optimale utilisée pour l'étude un système à paramètres fixes. Puis, on détaille l'approche des inclusions différentielles utilisée pour l'étude des systèmes à paramètres variables. Raczynski a développé cette méthode mathématique pour l'utiliser dans la simulation des systèmes dynamiques. Nous développons un algorithme de résolution d'une inclusion différentielle et nous l'appliquons à un système mécatronique de pilotage.

1- Cas particulier d'inclusion différentielle : La commande optimale

La commande optimale représente un cas particulier de l'inclusion différentielle. Il s'agit de calculer la commande optimale qui permet d'aller d'un point initial à un point final en optimisant un certain critère. En effet, l'objectif est de déterminer une seule commande optimale alors que pour l'inclusion différentielle on cherche l'ensemble des solutions optimales pour une certaine variation paramétrique.

Nous nous intéressons dans ce paragraphe aux méthodes de résolution des problèmes de contrôle optimal. Nous considérons un système sous la forme :

$$\dot{X} = f(t, x(t), u(t)) \qquad (2.1)$$

Soit $x_0 \in R^n$ un point donné de l'espace d'état, supposé contrôlable par le système ci-dessus. Parmi l'ensemble des trajectoires admissibles du système considéré reliant x_0 à x_f, on veut sélectionner celle(s) qui amène(nt) le système (2.1) de la position initiale x_0 à la cible x_f en minimisant un coût $J(x_0, u)$ donné.

La théorie du contrôle optimal a connu un véritable essor depuis les années cinquante avec la découverte d'outils puissants tels que le principe de programmation dynamique de R. Bellman [Bellman,1957] ou le principe du maximum de Pontryagin [Pontryagin,1962]. Ces résultats jouent un rôle essentiel en théorie du contrôle optimal des systèmes non linéaires et ont donné naissance à deux approches différentes des problèmes de contrôle optimal.

La première approche est basée sur l'utilisation du principe du maximum de Pontryagin [Pontryagin,1962] qui fournit une formulation hamiltonienne des problèmes de contrôle ainsi que des conditions nécessaires d'optimalité. Une famille importante de méthodes numériques issues de cette approche est constituée des méthodes de tir indirectes. Ces méthodes très efficaces en toute dimension peuvent être difficiles à mettre en œuvre d'un point de vue numérique et demandent une connaissance à priori de la structure de la trajectoire optimale.

La seconde approche repose sur le principe de programmation dynamique de R. Bellman [Bellman,1957] et la caractérisation de la fonction coût $J(x_0, u)$ en termes de solution de viscosité d'une équation d'Hamilton-Jacobi-Bellman (HJB) non linéaire [Bardi, and all, 1997]. Les méthodes numériques de résolution de l'équation (HJB) appliquent des schémas de discrétisation en temps et/ou en espace [Bardi, and all, 1997], [Bertsekas,1984] et sont

donc généralement simples à mettre en œuvre. Malheureusement, ces algorithmes efficaces en petite dimension demandent un coût trop élevé en mémoire pour être applicables en grande dimension.

Etant donnés un point initial x_0 et une cible x_f dans l'espace d'´état, le problème de contrôle associé au système (2.1) et au coût présenté par f_0 dans l'équation de l'hamiltonien H (défini ci-dessous équation (2.3)), consiste à déterminer des trajectoires du système (2.1) qui relient x_0 à x_f avec un coût minimum. Le problème est dit en horizon infini si le temps final t_f mis pour atteindre la cible n'est pas fixé.

Avant de s'intéresser aux méthodes de résolution de ce type de problème, la première question que nous nous posons est celle de l'existence de trajectoires optimales du système (2.1) par rapport au coût considéré.

Pour la résolution du problème de commande optimale on fait appel à la première approche en utilisant le Principe de Maximum du Pontryagin.
Commençons par rappeler le principe du maximum de Pontryagin, ou plus exactement du minimum pour le problème qui nous intéresse.

Le théorème 2.1 énoncé ci-après est une version forte du principe du maximum [Pontryagin,1962] prenant en compte les contraintes sur le contrôle. Les références concernant ce principe sont nombreuses ; citons en particulier [Pontryagin,1962] pour une démonstration de ce théorème ou encore[Trélat, 2005] pour une étude détaillée de son cadre d'application.

Théorème 2.1 (Principe du maximum de Pontryagin). On considère le système de contrôle :

$$\dot{X} = f(t, x(t), u(t)) \qquad (2.2)$$

où le champ f est supposé de classe C1 sur $R \times R^n \times R^m$. Soit u un contrôle admissible associé à une trajectoire x qui transfère le système de la position x_0 au temps t = 0 à la cible $x_f \in R^n$ au temps final t_f non fixé.
On introduit l'Hamiltonien H du système :

$$H(x,u,p,t) = \sum_{j=1}^{n} p_j f_j + f_0 \qquad (2.3)$$

Si le contrôle u est optimal sur l'intervalle de temps $[0,t_f]$, alors il existe une application non triviale $P : [0,tf] \rightarrow R^n$ absolument continue, appelée vecteur adjoint, et un réel $P_0 \geq 0$ tels que, pour presque tout $t \in [0,tf]$:

- $\dot{x}(t) = \frac{\partial H(x,u,p,t)}{\partial p} \quad [0,t_f]$ (2.4)
- $\dot{p}(t) = \frac{\partial H(x,u,p,t)}{\partial x}$ (2.5) (équation d'Euler Lagrange)
- $\frac{\partial H(x,u,p,t)}{\partial u} = 0$ (2.6) le long de la trajectoire optimale

La commande optimale permet d'étudier le système dans un état continu. On cherche la commande optimale qui a pour résultat, en minimisant un certain critère, la trajectoire optimale.

Dans le cas où on peut faire varier un des paramètres pour chercher la ou les trajectoire(s) optimale(s) correspondante(s), cela nous mène à étudier les inclusions différentielles.

La commande optimale est un cas particulier d'inclusion différentielle, car dans la commande optimale on cherche une trajectoire optimale en optimisant un critère. En inclusion différentielle on cherche les trajectoires optimales pour optimiser l'enveloppe ou le domaine atteignable avec en point commun l'utilisation de la maximisation de l'Hamiltonien.

2- Inclusions différentielles

Comme on l'avait déjà vu, une manière générale de modéliser les systèmes continus est d'utiliser les équations différentielles. Par contre, une variation infinitésimale de l'un des composants peut affecter considérablement le fonctionnement global du système. Car si un paramètre de la partie physique du système est variable alors le coefficient de l'équation différentielle correspondant varie aussi dans un intervalle. Un des outils mathématiques les plus appropriés pour modéliser ce type de systèmes est l'inclusion différentielle.

Les inclusions différentielles représentent une généralisation des équations différentielles avec des paramètres variables. La solution donnée par les inclusions différentielles est le « domaine atteignable » au lieu d'une seule trajectoire [Raczynski, 2006]. Dans les travaux de Raczynski

[Raczynski, 2007] on trouve une comparaison entre la méthode de Monte Carlo et la méthode des inclusions différentielles (ID) avec un exemple qui montre que le domaine atteignable solution trouvée par la méthode des ID est plus précis que celle trouvée par l'approche de Monte Carlo qui se montre sous un ensemble de points qui ne représente que 2-5% du diamètre de la solution réelle ; on pourra en trouver une meilleure application dans les variations des systèmes dynamiques [Raczynski, 2004]. L'auteur montre que le domaine atteignable solution de l'ID ne pourrait pas être obtenu par simple extension de l'algorithme de résolution des équations différentielles.

La solution des ID est un état « domaine » dans l'espace temps-état. Il s'agit d'une enveloppe qui inclut toutes les trajectoires possibles de l'ID. Cependant, trouver les limites du domaine atteignable n'est pas facile et systématique.

La forme générale de l'ID est comme suit :

$$\frac{dx}{dt} \in F(t, x(t)), \qquad x(0) \in X_0 \qquad (2.7)$$

avec F est une application de $R \times R^n$ dans R^n,

X_0 est le l'ensemble de l'état initial et $C \in R$

$$F(t, x(t)) = \left\{ z = f(t, x, v) \,\middle|\, v \in C(t, x) \right\} \qquad (2.8)$$

Comme déjà indiqué au chapitre I, on trouve dans [Raczynski, 2006] une méthode pour résoudre les inclusions différentielles appelée «Differential inclusion solver DIS ».

Début

1- Initialiser D la probabilité de fonction de distribution égale à la fonction de densité uniforme, et initialiser x=x0.
2- Générer le vecteur conjugué initial p
3- Intégrer l'équation dynamique du système et l'équation de l'Hamiltonien en utilisant une commande qui le maximise à chaque pas d'intégration.
4- Enregistrer le vecteur initial p ainsi que toutes les trajectoires dans un fichier.
5- Sélectionner le point final x_k qui correspond à la région de densité minimale de point x. cela peut être obtenu en détectant les zones de déconcentration de trajectoires dans le domaine atteignable ou sur sa projection 2D pour les systèmes de dimension n>2.

6- Modifier la fonction de distribution en augmentant la probabilité de distribution au voisinage du point x_k et en raffinant la recherche de trajectoires dans les zones de déconcentration de trajectoires.
7- S'il y en a assez de trajectoires enregistrées « Arrêt », sinon aller à l'étape 1.

Fin

Ce pseudo algorithme contient plusieurs sous problèmes comme la maximisation de l'Hamiltonien à chaque pas d'intégration et la modélisation de la partie variable du système.
Dans le point (5) de cet algorithme on pourra détecter les zones de déconcentration de trajectoires en calculant les distances entre chaque deux points voisins du domaine atteignable ou de sa projection 2D.
Dans le point (2) de l'algorithme ci-dessus Raczynski, [Raczynski, 2007] ne calcule pas les valeurs initiales vecteur conjugué p en utilisant les conditions de transversalité comme dans la commande optimale, mais il commence par des valeurs initiales choisies au hasard. Il fait la maximisation de l'Hamiltonien à chaque pas d'intégration et intègre les trajectoires une seule fois.

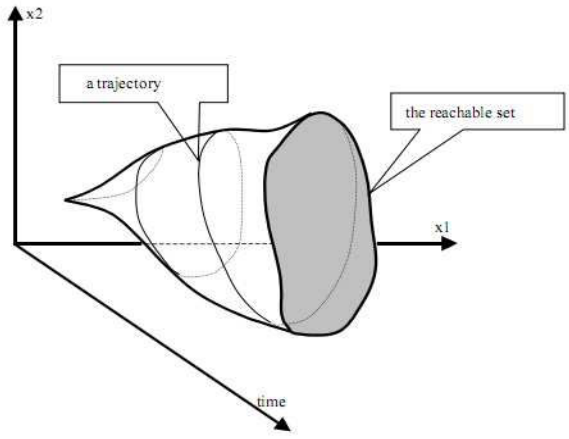


Figure 9 Un modèle de trajectoire et du domaine atteignable dans l'espace temps-état. [Raczynski 2006]

Dans le cas de commande optimale on calcule la loi de commande qui minimise un certain critère. On calcule la trajectoire optimale du système correspondante à la commande optimale

en utilisant le principe de maximum de Pontryagin pour la maximisation de l'Hamiltonien et les conditions de transversalité pour résoudre le problème aux deux limites de trajectoire.
D'autre part, en inclusions différentielles on cherche le domaine atteignable qui correspond au domaine qui inclut toutes les trajectoires possibles générées par la variation des paramètres. En d'autres termes, les trajectoires optimales solutions d'un système dynamique représentent les limites du domaine atteignable dans le cas des inclusions différentielles.
Dans le cas des ID on est dans une meilleure situation qu'en commande optimale car on n'a pas besoin d'utiliser les conditions de transversalité pour trouver le vecteur conjugué. Pour le générer on choisit les conditions initiales arbitrairement (prises ici de manière aléatoire). Il sera modifié en fonction des zones de déconcentration de trajectoires détectées dans le domaine atteignable. Car l'objectif n'est pas l'optimisation mais de trouver le domaine atteignable qui englobe toutes les solutions.
La résolution d'un problème de commande optimale nécessite l'utilisation du principe de maximum de Pontryagin pour la maximisation de l'Hamiltonien et pour trouver la trajectoire optimale. Alors qu'en ID la commande varie dans un intervalle dont les valeurs aux extrémités sont connues mais on n'a aucune idée sur leur distribution à l'intérieur de l'intervalle de variation. Dans ce cas la résolution du problème et plus précisément la maximisation de l'Hamiltonien nécessite un algorithme plus complexe que le principe de maximum de Pontryagin, comme la méthode du « steepest descent » .
Dans la partie qui suit, on donne un aperçu sur quelques algorithmes d'optimisation et on présente notre algorithme de « steepest descent pour un optimum global » utilisé dans l'algorithme de résolution d'inclusion différentielle.

3- Méthode d'optimisation possible pour le calcul de l'inclusion différentielle : la méthode du « Steepest descent »

Le problème d'optimisation est un problème relativement fréquemment rencontré. Dans la littérature on trouve différents algorithmes d'optimisation selon l'objectif et les données du problème [Bergounioux, 2001].

La méthode du « steepest descent » est la méthode la plus simple des méthodes du gradient. Le choix de la direction est celui correspondant à la diminution rapide de la fonction f qui est la direction opposée au $\nabla f(x_k)$. La recherche commence par un point arbitraire x_0 et ensuite

on suit la direction du gradient jusqu'au voisinage de la solution [Meza, 2010]. Les itérations suivent l'équation suivante :

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \qquad (2.9)$$

La méthode de steepest descent est simple et facile à appliquer et chaque itération est rapide. Cette méthode est très stable et garantie la localisation de l'optimum (s'il existe) après un nombre fini d'itérations. Mais malgré tous ces points positifs la méthode présente deux inconvénients : elle converge très lentement et elle ne donne que l'optimum local. Pour sa simplicité et tous ses avantages, plusieurs recherches ont été menées pour améliorer la méthode et trouver de meilleurs résultats.

4- Présentation d'algorithme du « Steepest descent » appliqué aux inclusions différentielles

Plusieurs recherches ont été menées sur la méthode du steepest descent pour profiter de ses avantages et essayer de l'améliorer et de limiter ses inconvénients. Parmi ces recherches on cite les travaux de Jaben et Dhunia [Jabeen, and all, 2010]. Ils ont développé des algorithmes pour la recherche du pas optimal pour l'optimisation et surtout ils ont modifié l'algorithme classique du « steepest descent » en utilisant différent points de départ pour la recherche de l'optimum global. Leur algorithme est nommé « Population based steepest descent method » (PSDM).

Dans notre cas on s'est inspiré de cette méthode pour obtenir notre propre algorithme d'optimisation. L'idée se base sur l'algorithme classique du steepest descent. On choisit un premier point de départ, on effectue les itérations jusqu'à trouver le premier optimum local. On prend la valeur de variable correspondante (par exemple le paramètre variant v), et on l'utilise comme un nouveau point de départ pour une nouvelle itération de l'algorithme. A la fin, on a parcouru tout l'intervalle de variation. On compare les valeurs d'optima locales pour trouver l'optimum global. On a appellé cet algorithme « Steepest Descent pour un Optimum Global et Déterministe » (SDOGD)

5- Proposition de l'algorithme du steepest descent pour un optimum global et déterministe (SDOGD) :

On applique un algorithme du steepest descent pour la maximisation de l'hamiltonien par rapport à un paramètre variable v.

Dans ce qui suit

Le paramètre v varie dans l'intervalle $v \in [v_{\min}, v_{\max}]$

α un pas fixé à l'avance tel que $\alpha \leq \frac{|v_{\max} - v_{\min}|}{10}$

λ_1 le premier pas d'incrémentation fixé à l'avance.

β coefficient de multiplication fixé pour permettre de rester dans l'intervalle de variation

Début

Initialiser $v_{f,1} = v_a$

m=1

boucle1 — **Tant que** $(v \geq v_a \quad et \quad v \leq v_b)$

k=1

On calcule $x_{1,m} = x(v_{1,m})$

On calcule $v_{2,m} = v_{1,m} + \lambda_1 \nabla H(v_{1,m})$

On calcule $x_{2,m} = x(v_{2,m})$

boucle 2 — **Tant que** $\left\| H(v_{k+1,m}) - H(v_{k,m)} \right\| > \xi$

On calcule $\lambda_{k,m} = \beta \frac{1}{\left\| x_{k+1,m} - x_{k,m} \right\|}$

On calcule $v_{k+1,m} = v_{k,m} + \lambda_{k,m} \nabla H(v_{k,m})$

k=k+1

Fin Tant que

Si l'optimum est un minimum alors « steepest Ascent »

Sinon « steepest Descent ».

$v_{f,m} = v_{k+1,m}$

m=m+1

Fin Tant Que

Fin

$\left\| x_{k+1,m} - x_{k,m} \right\|$ est la distance entre deux trajectoires voisines au même instant t.

La boucle 2 calcule la valeur optimale de l'hamiltonien en utilisant la méthode de « steepest descent ». A la sortie de la boucle on a $v_{f,m}$ qui nous donne la valeur optimale.

On enregistre cette valeur et on l'utilise comme entrée dans la boucle 1.

$v_{0,m} = v_{f,m} + \alpha$ est la nouvelle valeur de départ pour le calcul de l'optimum.

Par cette méthode on commence par une valeur de départ qui peut être l'un des extrema de l'intervalle de variation du paramètre variant v. On utilise la méthode de steepest descent jusqu'à obtenir le premier optimal local. On enregistre les valeurs de H et de v correspondantes. On prend une nouvelle valeur de point de départ qui vient juste après l'ancienne valeur donnant l'optimal local (avec un pas arbitraire α), puis on relance la recherche de nouvelle valeur d'optimal local en utilisant la méthode de « steepest descent » si le dernier point enregistré est un maximum, sinon on utilise la méthode de « steepest ascent ». cette démarche est illustrée dans la figure 10.

On enregistre tous les optimum dans un intervalle pour en faire la comparaison et trouver l'optimal global de la fonction.

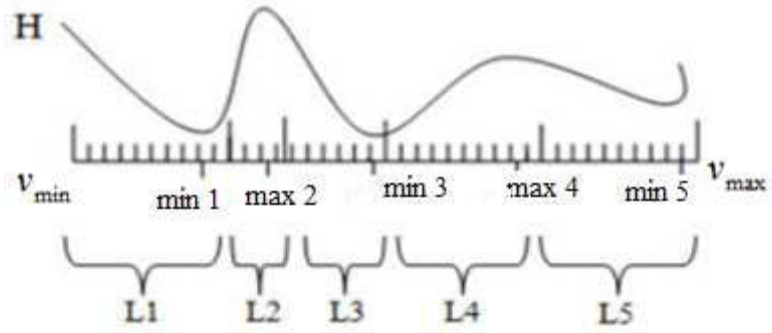


Figure 10 Exemple de l'Hamiltonien H et ses optima par la méthode de SDOGD

6- Exemple d'application de l'algorithme :

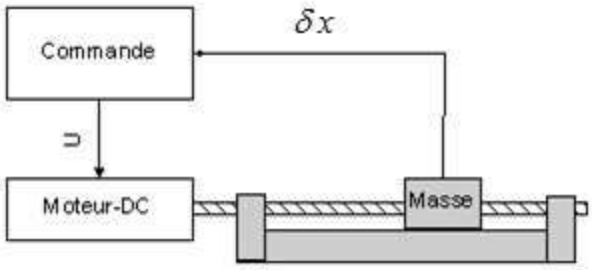


Figure 11 Système mécatronique de pilotage [Hadj-Amor 2008]

On a choisi comme application un système mécatronique simple qui est un système de pilotage à vis à billes. Le système de pilotage linéaire est contrôlé par une commande. Ce dernier alimente un moteur DC par une tension (U_0, $-U_0$) selon la position δx de la masse en déplacement.

Le modèle physique du système de pilotage linéaire, représenté dans la figure ci-dessus, est composé d'un moteur DC (Direct Current) à courant continu, d'une tige filetée et d'une masse pilotée en translation.

Les équations dynamiques du système sont : [Aublin, 1992]

$$\ddot{\theta} = \frac{d\dot{\theta}}{dt} = \frac{-b}{J+ma^2}\dot{\theta} + \frac{k}{J+ma^2}i \quad and \quad \delta x = p\theta$$
$$\frac{di}{dt} = \frac{-K}{L}\dot{\theta} - \frac{r}{L}i + \frac{V}{L} \qquad (2.10)$$
$$r \in [r_{\min}, r_{\max}] = [0.9\ ,\ 1]$$

Où :

<u>Les paramètres fixes :</u>

J : moment d'inertie ($kg.m^2$)

m : masse (kg)

L : inductance (H)

b : coefficient de frottement

V : tension (v)

k : constante électromécanique

<u>Les paramètres variables :</u>

r : résistance (Ω)

a : pas de vis (m)

Les variables

δx : déplacement (m)

θ : angle de rotation (rad)

i : intensité (A)

L'Hamiltonien est défini par l'équation suivante :

$$H = p_1(\frac{-b}{J+ma^2}\dot{\theta} + \frac{k}{J+ma^2}i) + p_2(\frac{-k}{L}\dot{\theta} - \frac{r}{L}i + \frac{V}{L}) \qquad (2.11)$$

Le vecteur conjugué est défini par intégration de l'équation de l'hamiltonien H en utilisant le principe de maximum de Pontryagin :

$$\dot{p}_i = -\frac{\partial}{\partial x_i}H(x,u,p,t) \qquad (2.12)$$

Soit $x = \begin{bmatrix} i \\ \dot{\theta} \end{bmatrix}$, $\qquad \dot{x} = \begin{bmatrix} \frac{di}{dt} \\ \ddot{\theta} \end{bmatrix} = A[x] + \begin{bmatrix} 0 \\ \frac{V}{L} \end{bmatrix}$ (2.13)

Avec $A = \begin{bmatrix} \frac{k}{J+ma^2} & \frac{-r}{L} \\ \frac{-b}{J+ma^2} & \frac{-k}{L} \end{bmatrix}$ (2.14)

$$\dot{p}_1 = -\frac{\partial H}{\partial i} = -p_1\frac{k}{j+ma^2} + p_2\frac{r}{L}$$
$$\dot{p}_2 = -\frac{\partial H}{\partial \dot{\theta}} = p_1(\frac{b}{J+ma^2}) + p_2\frac{k}{L} \qquad (2.15)$$

$$\begin{cases} \dot{p}_1 = -p_1\frac{k}{j+ma^2} + p_2\frac{r}{L} \\ \dot{p}_2 = p_1(\frac{b}{J+ma^2}) + p_2\frac{k}{L} \end{cases} \qquad (2.16)$$

Remarque : pour trouver le vecteur adjoint il suffit de résoudre le système (2.16) ci-dessus. Les conditions initiales $p_1(0)$ et $p_2(0)$ sont choisies arbitrairement.

Pour la maximisation de l'hamiltonien il faut définir la direction de la descente ce qui correspond au gradient de H.

$$\nabla H = \left[\frac{\partial H}{\partial r}\right] = \left[-\frac{1}{L} i\right] \qquad (2.17)$$

On suppose dans l'application numérique de l'exemple que la valeur initiale de la résistance r est 0.90.

Rappel :

α un pas fixé à l'avance tel que $\alpha \leq \frac{|v_{\max} - v_{\min}|}{10}$, on a choisi ici 0.02

λ_0 le premier pas d'incrémentation fixé à l'avance.

L'algorithme ci dessous constitue la résolution de l'inclusion différentielle pour un système continu avec des variations paramétriques. Les résultats de l'application de cet algorithme avec les données numériques ainsi que les résultats de simulation seront détaillés au chapitre 4.

Notre algorithme de résolution d'inclusion différentielle se limite à l'évolution positive dans le temps, par contre Raczynski est allé plus loin dans ses recherches en étudiant le problème inverse en considérant le domaine atteignable comme un état de départ.

Début

Initialiser $r_{f,1} = r_{\min} - \alpha$

m=1

Tant que $(r_{\min} \le r \le r_{\max})$

k=1

$r_{1,m} = r_{f,1} + 0.02 = 0.92$

$$x_{1,m} = \begin{bmatrix} \ddot{\theta}_{1,m} = \dfrac{-b}{J+ma^2}\dot{\theta}_{1,m} + \dfrac{k}{J+ma^2} i_{1,m} \\ \dfrac{di_{1,m}}{dt} = \dfrac{-K}{L}\dot{\theta}_{1,m} - \dfrac{r_{1,m}}{L} i_{1,m} + \dfrac{V}{L} \end{bmatrix}$$

$$r_{2,m} = r_{1,m} + \lambda_0 \nabla H(R_{1,m})$$
$$= 0.92 + \lambda_0 (-\frac{P_2}{L} i(0.92)$$

$$x_{2,m} = \begin{bmatrix} \ddot{\theta}_{2,m} = \dfrac{-b}{J+ma^2}\dot{\theta}_{2,m} + \dfrac{k}{J+ma^2} i_{2,m} \\ \dfrac{di_{2,m}}{dt} = \dfrac{-K}{L}\dot{\theta}_{2,m} - \dfrac{r_{2,m}}{L} i_{2,m} + \dfrac{V}{L} \end{bmatrix}$$

Tant que $\left\| H(u_{k+1,m}) - H(u_{k,m}) \right\| > \xi$

On calcule $\lambda_{k,m} = \beta \dfrac{1}{\left\| x_{k+1,m} - x_{k,m} \right\|}$

On calcule $u_{k+1,m} = u_{k,m} + \lambda_{k,m} \nabla H(u_{k,m})$

k=k+1

Fin Tant que

$u_{f,m} = u_{k+1,m}$

m=m+1

Fin Tant que

Fin

On pourra définir la distance euclidienne pour le calcul de pas d'incrémentation comme suit :

$$d(x1, x2) = \sqrt{\left|x_{2,1} - x_{1,1}\right|^2 + \left|x_{2,2} - x_{1,2}\right|^2} \qquad (2.18)$$

7- Etude du problème inverse

Dans [Raczynski, 2011] l'auteur développe la théorie du problème inverse c.à.d de prendre l'état final du domaine atteignable comme un état de départ. Dans ce cas le fait de reculer dans le temps de t_1 à t_0 ne suffit pas. Pour ceci on considère l'inclusion différentielle suivante :

$$\frac{dx}{dt} \in [-1,1] \quad , \quad x \in R \qquad (2.19)$$

Si on prend la condition initiale est x=0 à t=0 et on calcule le domaine atteignable à t=1 on aura comme résultat [-1, 1]. Par contre, si on commence de t=1 et du domaine atteignable $[-1,1]$ et on calcule l'inclusion différentielle (en inversant le signe de la dérivée), on aura comme résultat l'intervalle [-2,2] au lieu d'atteindre le domaine du départ (le point (0,0) dans ce cas).

Ceci montre que le domaine atteignable est en évolution continue, car la variation restera la même pour le problème inversé. La figure 12 illustre cette théorie pour le cas d'un problème de dimension 2. Dans la figure 12 le domaine atteignable solution de l'inclusion différentielle par inversion de signe de la dérivée est noté BRS.

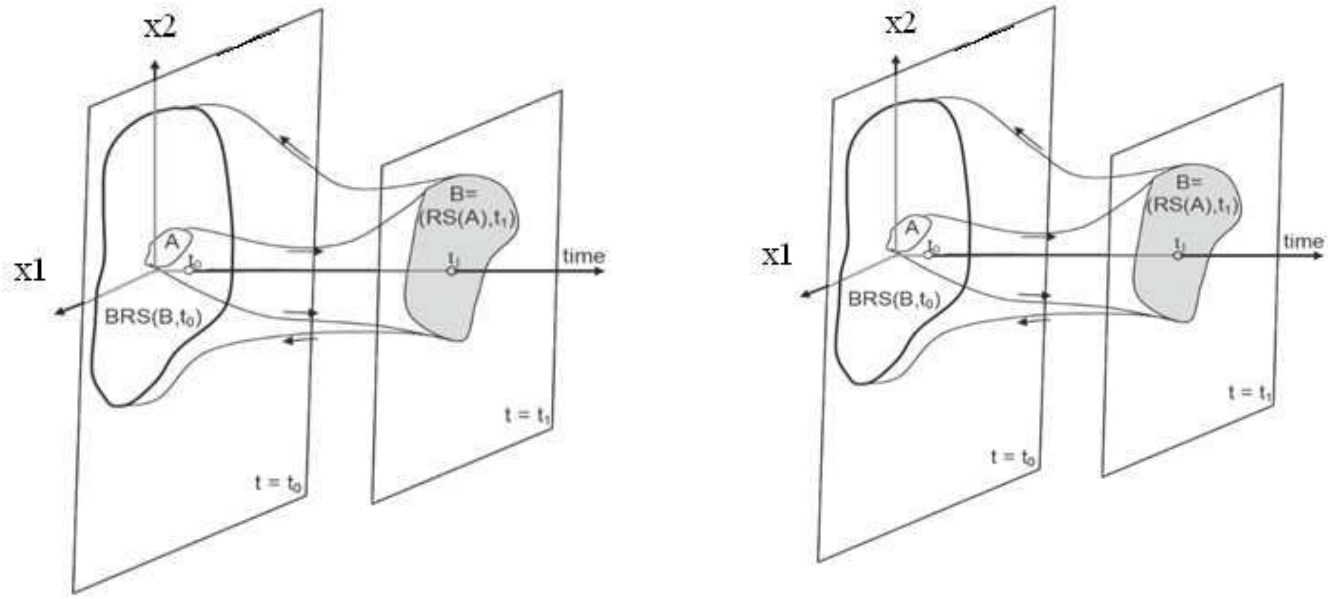


Figure 12 Etat initial A, domaine atteignable RS, et domaine atteignable inverse BRS [Raczynski, 2011]

Dans le cas général d'une inclusion différentielle à un intervalle du temps fixe, la solution du problème inverse devra atterrir sur l'ensemble initial A (figure 12). La différence entre l'ensemble A et l'ensemble BRS c'est que pour le premier on voit juste l'image inverse alors que pour le deuxième c'est l'inclusion différentielle avec inversion de signe de la dérivée (en d'autres termes c'est la continuité de l'inclusion différentielle mais avec un intervalle du temps négatif).

Cette étude montre que le domaine atteignable est en évolution continue. En effet, par définition, le domaine atteignable résultat d'une inclusion différentielle est le plus grand ensemble qui englobe toutes les solutions possibles. Il est en évolution continue dans le temps. L'inclusion différentielle n'est pas réversible, on ne peut en aucun cas revenir au domaine de départ mais il suffit de faire abstraction du temps positif pour voir l'évolution continue du domaine atteignable.

Pour le tolérancement, dans le cas où le domaine atteignable n'est pas inclus dans la zone du fonctionnement désiré. L'étude du problème inverse ne permet pas de déterminer la variation qui a entrainé ce résultat. Il faut trouver un autre outil qui répond à cela et permet de déterminer la variation limite pour rester dans la zone du fonctionnement désiré.

Cette étude peut ne pas être efficace en ingénierie mais plutôt en exploitation/maintenance car lorsqu'on cherche le problème inverse on cherche la cause du problème, c'est à dire que l'on a déjà le domaine atteignable d'un système avec une variation paramétrique et que l'objectif sera de chercher les variations qui ont mené à ce résultat. Le problème inverse ne résout pas ce problème car l'ID est irréversible dans le temps. Par contre elle montre la symétrie de l'évolution dans le temps. En effet, pour la même variation du départ, une évolution par l'ID dans le temps négatif donne le même résultat du domaine atteignable pour le même temps d'évolution positif.

Conclusion

Dans ce chapitre nous avons étudié une méthode déterministe à savoir la méthode des inclusions différentielles. Il s'agit d'un outil mathématique extension des équations différentielles. L'outil a été repris de son contexte mathématique et utilisé dans l'automatique et plus précisément dans la simulation par Raczynski. Cette méthode consiste à résoudre une équation différentielle avec un ou plusieurs de ses paramètres qui varient dans un intervalle. Cette méthode requière des connaissances en optimisation et commande optimale pour arriver à générer un résultat sous forme d'un domaine atteignable qui englobe tous les résultats possibles.

Dans cette partie on considère un système mécatronique purement continu et des variations paramétriques dont on connait les valeurs initiales et finales mais pas la loi de distribution à l'intérieur de l'intervalle. Comme on l'a constaté cette méthode constitue l'approche la plus appropriée pour l'étude de notre problème. Nous avons repris cette méthode et développé un algorithme qui permet la résolution d'inclusion différentielle. Dans cet algorithme on a soulevé différents sous problèmes comme l'optimisation de l'hamiltonien et le choix du pas de variation de paramètre à l'intérieur de l'intervalle. Nous avons développé un algorithme d'optimisation de steepset descent qui donne un optimum global.

L'application numérique de cet algorithme sur l'exemple mécatronique déjà introduit de pilotage et la simulation des résultats seront développés ultérieurement.

Il existe une autre méthode de résolution d'inclusion différentielle, il s'agit de l'algorithme pratique qui se base sur la méthode de Heun. Cette approche de résolution est une approche implicite. Elle sera détaillée dans le chapitre 5 et une comparaison avec notre algorithme de résolution sera établie.

Chapitre 3 : Les inclusions différentielles impulsionnelles

Introduction

Conclusion

Introduction

L'inclusion différentielle était le sujet d'étude du chapitre précédent, elle constitue l'étude d'un système continu avec une variation paramétrique. Pour élargir notre champ de recherches nous nous sommes intéressés à l'étude des systèmes hybrides qui comporte à la fois une évolution continue et discrète. L'automate hybride est un outil qui permet la modélisation des systèmes dynamiques hybrides dont les paramètres sont fixes. Pour modéliser des systèmes hybrides avec variations paramétriques on a choisi l'inclusion différentielle impulsionnelle. Aubin a présenté cet outil dans plusieurs de ses publications ; nous avons pris cette approche, nous avons identifié ses différents ensembles sur un système mécatronique et nous avons développé des algorithmes de résolution d'inclusion différentielle impulsionnelle avec un ou plusieurs paramètre (s) variable(s).

1- Les automates hybrides

1.1- Définition des systèmes dynamiques hybrides

Les systèmes dynamiques hybrides sont des systèmes à dynamiques différentes : continue et discrète. Rappelons qu'un état de variables est dit discret s'il prend un nombre fini de valeurs. Par contre, s'il est continu, il prend des valeurs dans l'espace Euclidien R^n . Par leur nature, les états discrets changent de valeurs par des « sauts » alors que, les états continus changent de valeurs soit par des sauts, soit dans le temps suivant des équations différentielles. Dans les systèmes hybrides on trouve ces deux dynamiques impliquées : l'évolution continue et les sauts discrets. En effet, la conception et l'analyse des systèmes hybrides peuvent s'avérer délicates et plus compliquées que celles d'un système purement discret ou purement continu car la dynamique discrète peut affecter l'évolution continue et vice versa [Lygeros, 2004], [Lygeros, 2003].

La dynamique hybride nous fournit un outil de modélisation pour différente applications dans le domaine de l'ingénierie :

- Dans les systèmes mécaniques : le mouvement continu peut être interrompu par des collisions.
- Dans les circuits électriques : les phénomènes continus comme le chargement des condensateurs sont interrompus par les commutations d'ouverture et de fermeture.
- Dans les systèmes de calcul embarqués : le domaine numérique peut interagir avec l'environnement analogique.

Dans ces systèmes cités ci-dessus, on modélise les composants discrets (commutateurs, valves, …) comme étant des composants qui introduisent un changement instantané sur le fonctionnement des composants continus [Bemporad, 2000].

1.2- Modélisation des systèmes dynamiques hybrides à paramètres fixes

La modélisation des systèmes dynamiques hybrides nécessite un outil de modélisation :

- Descriptive : qui permet de décrire les dynamiques continues et discrètes du système et de modéliser l'évolution continue et son affectation par les sauts discrets.

- Décomposable : qui permet de construire un model global à partir de simples composants
- Abstrait : réduire le problème de conception d'un modèle complexe à la conception des composants individuels, et pouvoir étudier la performance globale du système à partir de ses composants.

Dans la littérature on trouve différents langages de modélisation comme :

- Différentes variantes d'automates hybrides (HA)[Alur, 2000], [Alur, 1999]
- Le modèle « Hybrid Input /Output Automata » (HIOA) par exemple pour les systèmes avec du retard [Lynch, 1996]

Pour le cadre des systèmes dynamiques hybrides dont les composants sont fixes, les automates hybrides présentent un outil de modélisation bien adapté.

1.3- **Définitions des automates hybrides**

Les automates hybrides sont un outil de modélisation des systèmes dynamiques. Cela représente les systèmes qui ont un comportement continu avec des événements discrets. Cet outil est utilisé dans le cas non variationnel.

Un automate hybride est défini par la donnée de (Q, X, Σ, A, Inv, F, q0, x0) où :
- Q : est un ensemble fini de sommets, appelés situations, et q0 la situation initiale ;
- X, est l'espace d'état continu de l'automate, $X \subset \mathrm{R}^n$; x_0 est la valeur initiale de l'état continu;
- Σ, est un ensemble fini d'évènements ;
- A, est un ensemble de transitions définies par un quintuple (q, Guard, σ , Jump, q') et représentées par un arc entre les situations, où :

  - $q \in Q, q' \in Q$,
  - Guard, est un sous ensemble de l'espace d'état dans lequel doit se trouver l'état continu pour que la transition puisse être franchie,
  - Jump, représente la transformation de l'état continu lors du changement de situation ; elle est généralement exprimée sous forme d'une fonction de la valeur de l'état, avant commutation, dont le résultat est affecté comme valeur initiale de l'état continu dans la nouvelle situation ;

- $\sigma \in \Sigma$, est l'événement associé à la transition, cette association n'implique pas de donner un sens en terme d'entrée ou de sortie de l'événement ;

- Inv, est une application qui associe à chaque situation un sous-ensemble de l'espace d'état, appelé *invariant* de la situation, dans lequel l'état continu doit rester, lorsque la situation est q, l'état continu doit vérifier.
- F, définit pour chaque situation, l'évolution de l'état continu lorsque la situation est active ; cette évolution de l'état continu est le plus souvent exprimée par une équation différentielle ; elle peut dans certains cas être définie sous une forme moins explicite ou impérative, telle que l'inclusion différentielle qui définit, pour chaque variable, un intervalle dans lequel sa dérivée peut évoluer. [Henzinger, 1995].

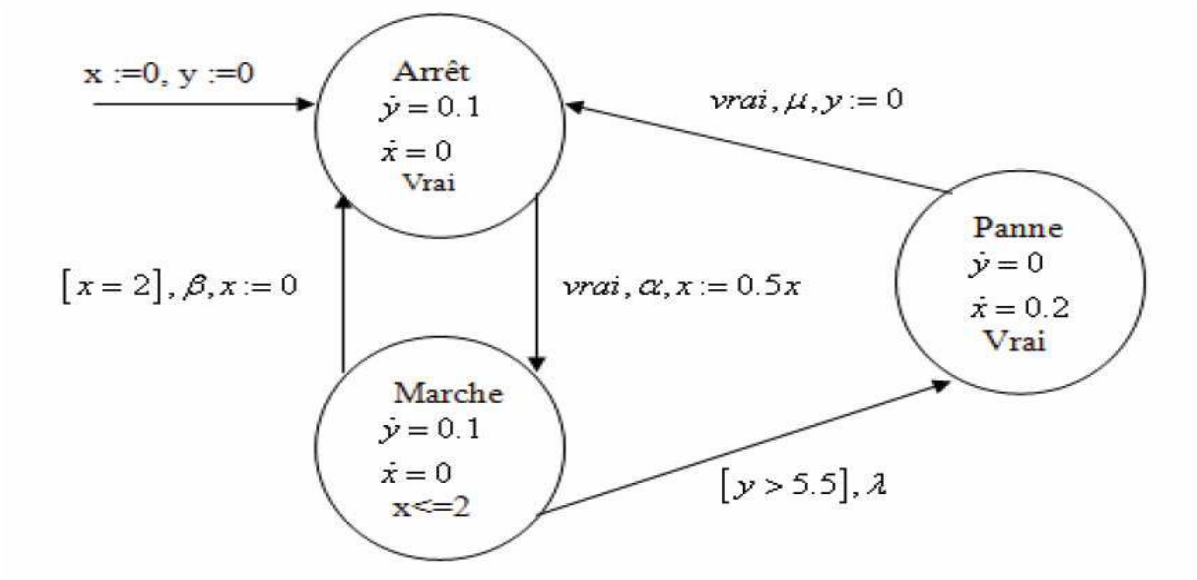


Figure 13 Automate hybride modélisant une machine simple [Hadj Amor, 2008]

La figure 13 correspond à un automate hybride possédant trois situations et un espace d'état continu de dimension deux, dans lequel les invariants des situations et les gardes des transitions sont exprimés sous la forme de prédicats, les activités de chacune des situations sont sous la forme d'équations différentielles, et les sauts sous forme de fonctions explicites [Zaytoon, 2001].

1.4- Evolution d'un automate hybride.

La sémantique des automates hybrides est définie en considérant qu'à chaque instant, l'état d'un automate hybride est donné par la paire (q, x) correspondant à l'association d'une situation et d'une valeur du vecteur d'état, et que cet état peut évoluer :

- Soit par une transition instantanée qui change la situation et la valeur de l'état continu (fonction de saut),
- Soit par la progression du temps dans la situation courante, ce qui entraîne un changement de l'état continu conformément à l'activité, F, de la situation.

Il est ainsi possible de franchir une transition si l'état continu est dans le sous-espace défini par son *Guard*, et est tel que sa transformation par le saut satisfait l'*Invariant* de la situation d'arrivée. Cependant, rien n'impose ce franchissement tant que l'*Invariant* de la situation de départ est satisfait.

On peut utiliser l'automate hybride pour modéliser le comportement d'un système dynamique hybride complexe. Mais, si l'on veut utiliser ce formalisme pour réaliser un système dynamique hybride de commande industrielle, alors il faut ajouter des commandes telles que les variables continues et les évènements typés Entrée/Sortie. [Hien, 2001]

1.5- Couplage des automates hybrides

Il est possible de coupler des automates hybrides en se basant sur le mécanisme de composition synchrone. Ce couplage a pour but de séparer la partie contrôle de la partie opérative. Une première définition est la suivante:

« Si l'on considère deux automates et un événement qui appartient à l'ensemble des évènements de chacun d'eux, une transition étiquetée par cet événement ne peut être franchie dans l'un des automates que s'il existe une transition étiquetée, par ce même événement, franchissable dans l'autre automate. Les deux transitions sont alors franchies simultanément. » [Hadj Amor, 2012],[Zaytoon, 2001]

Le formalisme des automates hybrides est issu d'un formalisme de description de systèmes à événements discrets. Les situations de l'automate hybride comportent les équations différentielles des états continus. Il est donc possible d'extraire les équations d'états du système. D'un autre côté, ce formalisme permet de concevoir des commandes grâce au

couplage entre les automates hybrides. Un autre avantage est la disponibilité d'outil de vérification formelle pour ce formalisme. On peut citer HyTech [Henzinger, 1997], [Henzinger, 1997a].

Définition : un ensemble de temps hybride est une séquence fini d'intervalles $\tau = \{I_0, I_1, ..., I_n\} = \{I_i\}_{i=0}^{N}$

- Pour $i < N$; $I_i = [\tau_i, \tau_i^{'}]$
- Si $N < \infty$ *alors* $I_N = [\tau_N, \tau_N^{'}]$ *ou* $I_N = [\tau_N, \tau_N^{'}[$
- Pour tout i, $\tau_i \leq \tau_i^{'} = \tau_{i+1}$

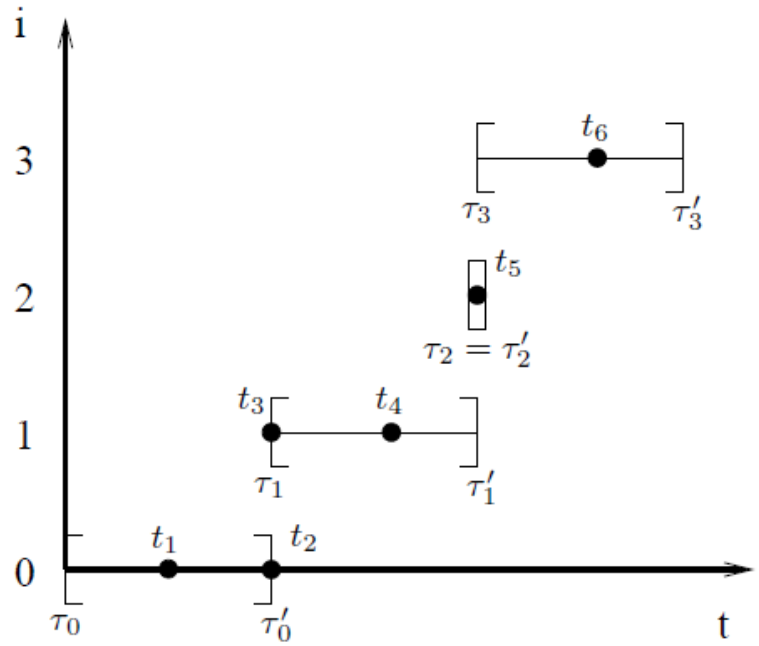


Figure 14 Un ensemble de temps hybride $\{[\tau_i, \tau_i^{'}]\}_{i=0}^{3}$, [Lygeros, 2004]

Un exemple d'ensemble de temps hybride est donné dans la figure 14. On remarque que l'extrémité droite d'un intervalle $\tau_i^{'}$ d'un intervalle I_i coïncide avec l'extrémité gauche τ_{i+1} de l'intervalle I_{i+1} (les instants du temps sont notés t_1, t_2, t_3). $\tau_i^{'}$ correspond à l'instant juste avant la transition et τ_{i+1} correspond à l'instant juste après la transition discrète. La transition est supposée être instantanée, donc $\tau_i^{'} = \tau_{i+1}$. L'avantage de cette convention est qu'elle nous permet de modéliser une situation où plusieurs transitions pourront être franchies l'une après

l'autre au même instant (le parallélisme est supporté), dans ce cas $\tau'_{i-1} = \tau_i = \tau_i' = \tau_{i+1}$ (ceci correspond à l'intervalle $I_2 = [\tau_2, \tau_2']$ dans la figure 14).

On peut classer les éléments d'un ensemble de temps hybride suivant un certain ordre. Pour $t_1 \in [\tau_i, \tau_i'] \in \tau$ et $t_2 \in [\tau_j, \tau_j'] \in \tau$ on peut dire que t_1 précède t_2 (noté par $t_1 \prec t_2$) si $t_1 < t_2$ (le réel t_1 est numériquement inférieur à t_2) ou si i < j (cad t_1 parcourt un intervalle antérieur à t_2). Dans la figure 14 on a $t_1 \prec t_2 \prec t_3 \prec t_4 \prec t_5 \prec t_6$. Dans le cas général, pour deux instants différents t_1 et t_2 qui parcourent un intervalle τ on a soit $t_1 \prec t_2$ soit $t_2 \prec t_1$. Chaque ensemble de temps hybride peut être linéairement classé par la relation $\prec$.

L'ensemble de temps hybride est utilisé pour déterminer le temps dans lequel les états du système hybride évoluent :

- Pour les systèmes continus, dont l'évolution de l'état $x \in R^n$ est une fonction $x(.): [0,T] \to R^n$, de l'intervalle $[0,T]$ dans l'ensemble R^n.
- Pour les systèmes discrets dont les états évoluent suivant un ensemble fini $q \in \{q_1, \ldots, q_n\}$ est une séquence d'état.
- Pour les systèmes hybrides dont les états ont un comportement continu $x \in R^n$ et un comportement discret $q \in \{q_1, \ldots, q_n\}$ nous devons fournir une combinaison de ces deux notions.

Définition de trajectoire hybride **:**

Une trajectoire hybride est un triplet (τ, q, x) qui consiste en un ensemble de temps hybride $\tau = \{I_i\}_{i=0}^N$ et deux séquences de fonctions $q = \{q_i(.)\}_{i=0}^N$ et $x = \{x_i(.)\}_{i=0}^N$ avec $q_i : I_i \to Q$ et $x_i : I_i \to R^n$

Définition de l'exécution :

L'exécution d'un automate hybride est une trajectoire hybride (τ, q, x) qui satisfait les conditions suivantes :

- Les conditions initiales $(q_0(0), x_0(0)) \in$ Init.
- L'évolution discrète : pour tout i,

$$\left(q_i(\tau_i^{'}), q_{i+1}(\tau_{i+1})\right) \in A, \quad x_i(\tau_i^{'}) \in G(q_i(\tau_i^{'}), q_{i+1}(\tau_{i+1}))$$

Avec Init l'état initial et G : l'ensemble de guard,

- L'évolution continue : pour tout i

1- $q_i(.): I_i \to Q$ est constant pour $t \in I_i, càd\, q_i(t) = q_i(\tau_i)$ pour tout $t \in I_i$

2- $x_i(.): I_i \to X$ est la solution de l'équation différentielle de $\frac{dx_i}{dt} = f(q_i(t), x_i(t))$

it

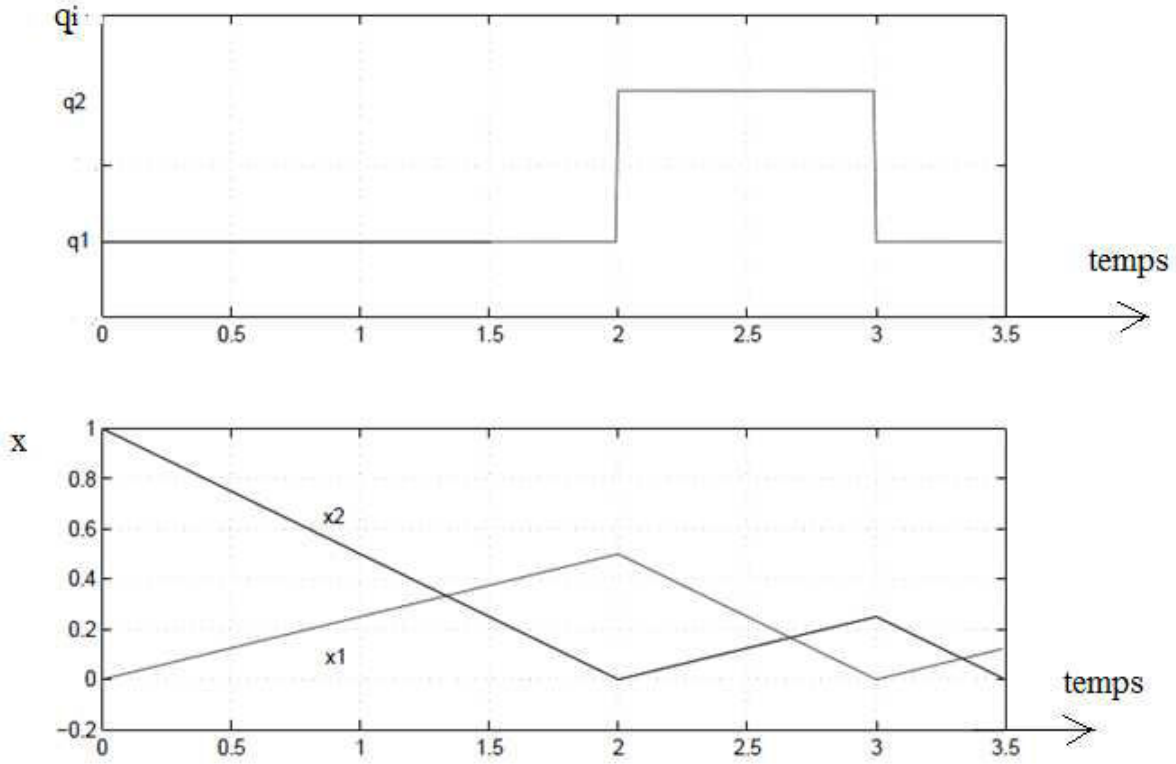


Figure 15 Exemple d'exécution d'un automate hybride [Duc, 1999]

C'est un exemple d'évolution d'un automate hybride. La partie supérieure montre l'évolution discrète et la partie inférieure montre l'évolution continue d'un automate hybride.

Il s'agit d'une commande tout ou rien. De 0 à 2 sur l'axe du temps, q1 est active. x1 et x2 évoluent dans le temps. De 2 à 3 sur l'axe du temps, c'est q2 qui est active. A cet instant correspond un changement du comportement de x1et de x2 dû au changement de l'état.

Ce formalisme permet l'étude des systèmes dynamiques hybrides à paramètres fixes. En effet, dans le cas de variation paramétrique, la dynamique du système ne peut pas être modélisée par des équations différentielles (comme on l'a montré au chapitre « les inclusions différentielles ») mais nécessite une modélisation plus générale du système, d'où le recours aux inclusions différentielles impulsionnelles.

2- Inclusion différentielle impulsionnelle

Les inclusions différentielles impulsionnelles permettent de modéliser des systèmes hybrides avec des paramètres variant dans l'évolution discrète, l'évolution continue ou les deux.

2.1- Définition d'une inclusion différentielle impulsionnelle[Aubin, 2002] :

Une Inclusion Différentielle Impulsionnelle IDI est une collection H=(X, F, R, J) avec

- X : vecteur d'espace de dimension fini
- $F : X \to 2^X$ est l'inclusion différentielle avec $\dot{x} \in F(x)$
- $R : X \to 2^X$ l'ensemble de reset
- $J \subseteq X$ l'ensemble des transitions forcées.

Notation :

- 2^K est l'ensemble de tous les sous-ensembles de K
- $R^{-1}(K)$ l'image inverse de K dans R ($R^{-1}(K) = \{x \in X \mid R(x) \cap K \neq \phi\}$)

Le fonctionnement d'une inclusion différentielle impulsionnelle peut être interprété comme un contrôle impulsionnel en introduisant l'ensemble de commutation (switching map) :

$$S : X \to 2^X$$
$$S(x) = \{x' \in X \mid \exists\, y \in R(x), \quad x' = y - x\} \qquad (3.1)$$

Dans le reste de ce document, on appelle situation le couple ([τi,τi'], et l'inclusion différentielle F associée.)

2.2- Définition du fonctionnement de l'IDI

Le fonctionnement d'une IDI H=(X, F, R, J) est une paire (τ,x) avec τ est l'ensemble de temps hybride et $x:\tau \to X$ une application qui satisfait :

- L'évolution discrète : $\forall i\,,\, x(\tau_{i+1}) \in R(x(\tau_i^{'}))$
- L'évolution continue : si $\tau_i < \tau_i^{'}$, $x(.)$ est solution de l'inclusion différentielle $\dot{x} \in F(x)$ dans l'intervalle $[\tau_i, \tau'_i]$ avec $x(t) \notin J \quad \forall \quad t \in [\tau_i, \tau_i^{'}[$

Le fonctionnement d'une IDI peut évoluer d'une manière continue suivant l'inclusion différentielle $\dot{x} \in F(x)$ jusqu'à atteindre J. Tant que $R(x) \neq \phi$ une transition de la situation x dans une situation de $R(x)$ peut être franchie (une transition peut être franchie durant $R(x)$ mais ce n'est pas une condition), alors que J force les transitions (quand $x \in J$ la transition devra être franchie). Il faut noter que si dans une situation $x \in X$ une transition doit être franchie ($x \in J$) mais qu'elle ne peut pas l'être ($R(x) = \phi$) alors le système bloque. Dans ce cas il n'existe aucun fonctionnement de l'IDI qui commence de x autre que le fonctionnement trivial ([0,0],x). Dans la figure 16 un exemple d'un fonctionnement d'une IDI est représenté, avec les lignes continues représentent l'évolution continue et les lignes interrompues représentent les transitions discrètes.

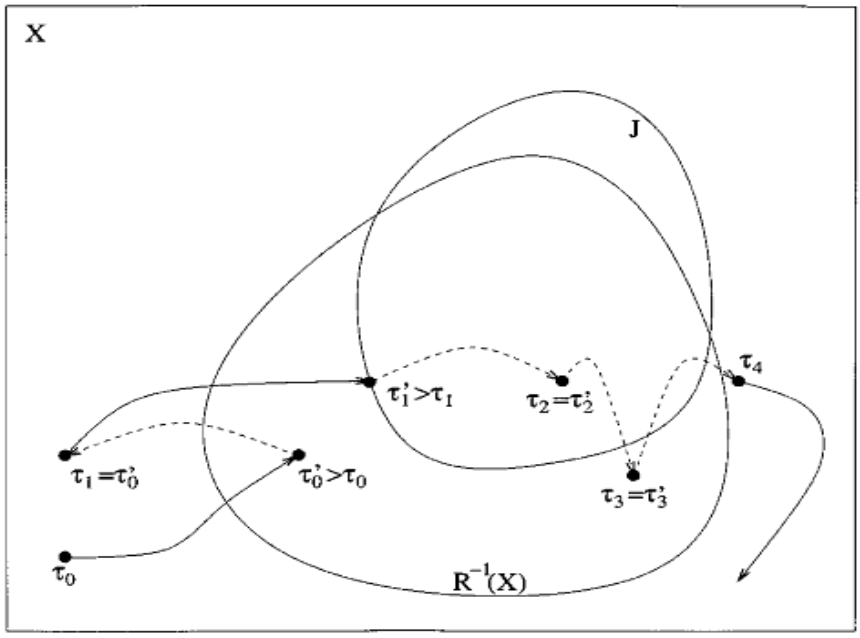


Figure 16 Un fonctionnement d'une inclusion différentielle impulsionnelle [Aubin, 2002]

2.3- Classification des fonctionnements de l'IDI :

Les fonctionnements des IDI sont classés suivant « l'horizon du temps » s'il est fini ou non, s'il a un nombre fini ou infini des transitions discrètes,…

Un fonctionnement (τ, x) d'une inclusion différentielle impulsionnelle H est dit :

- ***Fini :*** si τ est une séquence finie avec un intervalle compact
- ***Ouvert fini :*** si τ est une séquence finie avec un intervalle de la forme $[\tau_N, \tau'_N[$ avec $\tau'_N < \infty$
- ***Infini :***si τ est une séquence infinie ou si $\Sigma_i(\tau'_i - \tau_i) = \infty$
- ***Zeno :*** si τ est une séquence infinie et si $\Sigma_i(\tau'_i - \tau_i) < \infty$

Dans certains cas une inclusion différentielle impulsionnelle peut avoir un fonctionnement qui peut échapper à l'infini dans un intervalle du temps fini durant une évolution continue, ou un fonctionnement qui bloque ou un fonctionnement zeno.

Dans le cas d'un « temps fini » le fonctionnement est défini sur une séquence finie τ qui correspond à l'extrémité d'un intervalle ouvert $[\tau_N, \tau'_N[$ avec $\tau'_N < \infty$ et $\lim_{t \to \tau_N} \|x(t)\| = \infty$. Cette situation peut être évitée en imposant des hypothèses de régularité [Aubin, 2002].

Dans le cas où le système bloque, le fonctionnement est défini sur une séquence finie τ au long de $[\tau_i, \tau'_i]$ tel que à $x(\tau'_N)$ l'évolution continue et l'évolution discrète sont non autorisées. (c.à.d $x(\tau'_N) \in J$ et $R(x(\tau'_N)) = \phi$) pour éviter cette situation, l'hypothèse suivante est émise :

<u>*Hypothèse 1*</u> : On dit qu'une inclusion différentielle impulsionnelle H=(X, F, R, J) satisfait l'*hypothèse 1* si $J \subseteq R^{-1}(x)$ et si J est ouvert . $F(x) \cap T_I(x) \neq \phi$) pour tout $x \in I \setminus \mathrm{R}^{-1}(\mathrm{x})$.

L'hypothèse 1 implique que pour $x \in X$ l'évolution continue est impossible (car $x \in J$ ou forcé d'entrer dans J tout au long des solutions d'inclusion différentielle) alors, une transition discrète peut être franchie ($R(x) \neq \phi$). En tenant compte de l'hypothèse 1 et en y ajoutant des conditions techniques, on pourra dire que chaque fonctionnement fini d'une inclusion

différentielle impulsionnelle pourra être étendu à un fonctionnement infini [Aubin, 2002], [Liu, 2008].

Dans le cas de *fonctionnement Zeno* le système prend un nombre infini de transitions discrètes dans un intervalle fini du temps.

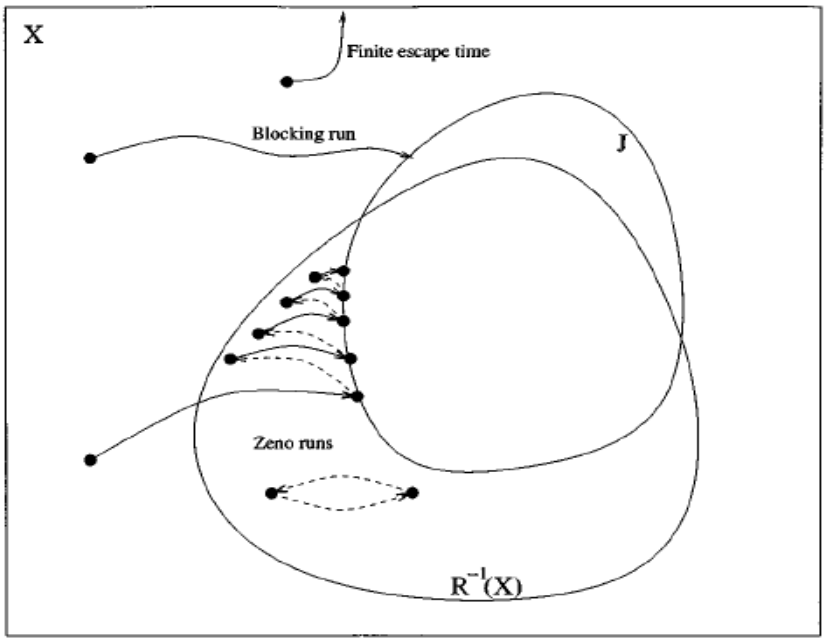


Figure 17 Exemple d'un fonctionnement zeno, bloqué et fini [Aubin, 2002]

2.4- Identification des ensembles d'une inclusion différentielle impulsionnelle

Une inclusion différentielle impulsionnelle est une extension d'une inclusion différentielle et des systèmes à temps discrets.

Une inclusion différentielle $\dot{x} \in F(x)$ peut être considérée comme une inclusion différentielle impulsionnelle H=(X, F, R, J) avec $R = \phi \quad \forall x \in X \quad et \quad J = \phi$.

Selon la nature de ses ensembles (vide, non vides ou nul …) une inclusion différentielle impulsionnelle peut être définie de différentes manières : inclusion différentielle, système à temps discret ou encore un système de contrôle hybride.

Dans la partie qui suit on a procédé à une identification de différents ensembles d'une inclusion différentielle impulsionnelle appliquée à un système mécatronique.

On a mis notre exemple d'application sous forme d'inclusions différentielle impulsionnelle en maintenant la modélisation du système dynamique par un automate hybride et en introduisant les incertitudes sous forme d'inclusion différentielle dans les situations.

Considérons la modélisation du système mécatronique de pilotage présenté au chapitre précédent (voir chapitre 2) :

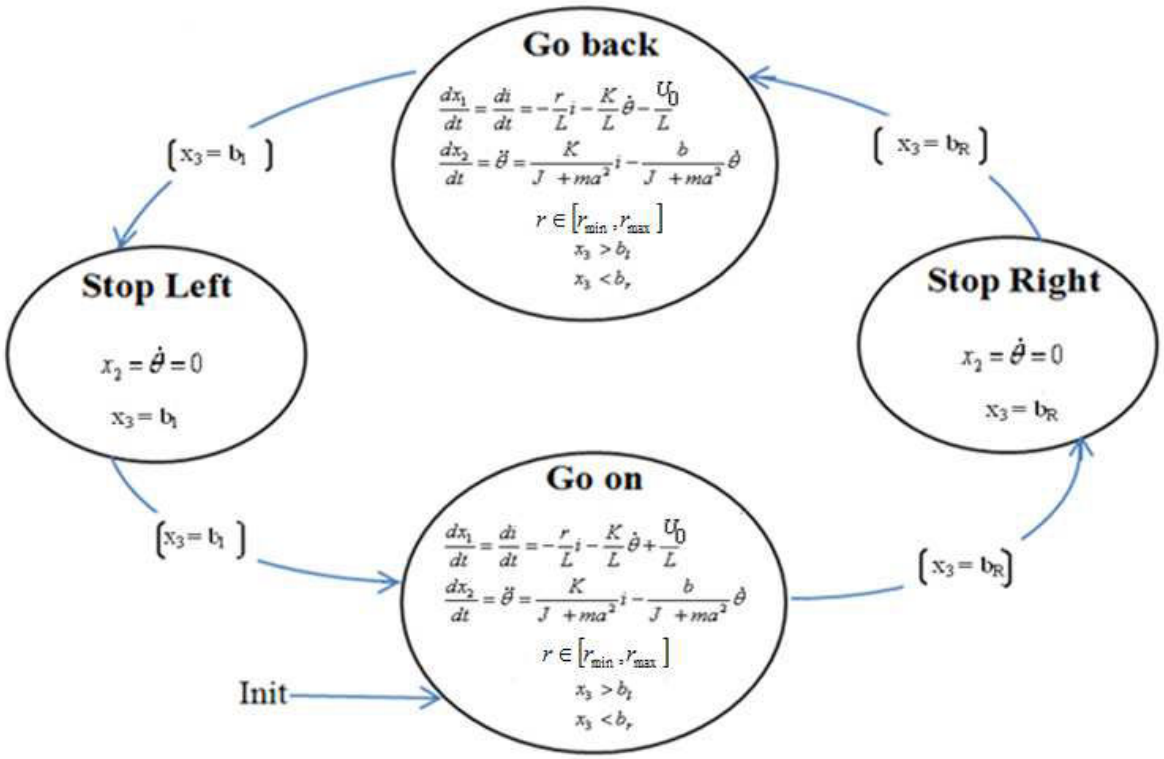


Figure 18 Un automate hybride d'un système mécatronique

Cet automate décrit le fonctionnement d'un système mécatronique (il sera détaillé dans le chapitre 4). Le système est composé d'une masse qui effectue des allers retours. La masse avance jusqu'à atteindre une butée et elle s'arrête avant de rebrousser chemin et revient sur sa position de départ.

Dans cet exemple on suppose que la résistance « r » varie dans un intervalle.

On considère le vecteur d'état suivant :

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} i \\ \dot{\theta} \\ a\theta \end{bmatrix}$$

Avec :

- i l'intensité
- $\dot{\theta}$ la vitesse angulaire
- $a\theta$ la position

L'inclusion différentielle F :

L'inclusion différentielle est définie par: $F(t,x(t)) = \{z = f(t,x,v) \mid v \in C(t,x)\}$

Dans notre exemple mécatronique:

$$F_m : X \to 2^X$$

$$F_m = \begin{cases} \dfrac{di}{dt} = -\dfrac{r}{L} i - \dfrac{K}{L}\dot{\theta} + \dfrac{U}{L} \\ \ddot{\theta} = \dfrac{K}{J + ma^2} i - \dfrac{b}{J + ma^2} \end{cases} \qquad (3.2)$$

$$r \in [r_{\min}, r_{\max}]$$

L'ensemble des transitions forcées J:

J force les transitions, donc une transition doit être franchie quand $x \in J$. Dans notre exemple une transition est franchie quand $x=b_l$ ou $x=b_r$

$$J_m : X \to 2^X$$

$$J_m = \begin{cases} x_3 = b_l \quad if \quad U := U_0 \\ or \\ x_3 = b_r \quad if \quad U := -U_0 \end{cases} \qquad (3.3)$$

L'ensemble du reset R:

R permet aux transitions d'être franchies quand $R \neq \phi$ sinon le système bloque. Pour notre exemple l'ensemble du reset est :

$$R_m : X \to 2^X$$

$$R_m = \begin{cases} i \in [i_{\min}, i_{\max}] \quad and \quad \dot{\theta} = 0 \quad if \quad x_3 = b_t \quad or \quad x_3 = b_r \\ \phi \end{cases} \qquad (3.4)$$

L'ensemble de commutation (swich) S :

$$S(x) = \{x' \in X \mid \exists\, y \in R(x), \quad x' = y - x\} \qquad (3.5)$$

L'ensemble S donne le vecteur d'état à la commutation entre les situations. Dans notre exemple S est :

$$S_m : X \rightarrow 2^X$$

$$S_m = \left\{ \begin{array}{l} si \;\; x_3 = b_l \quad v = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \\ \\ ou, \quad si \quad x_3 = b_r \quad v = \begin{bmatrix} 0 \\ -\dot{\theta} \\ 0 \end{bmatrix} \end{array} \right\} \qquad (3.6)$$

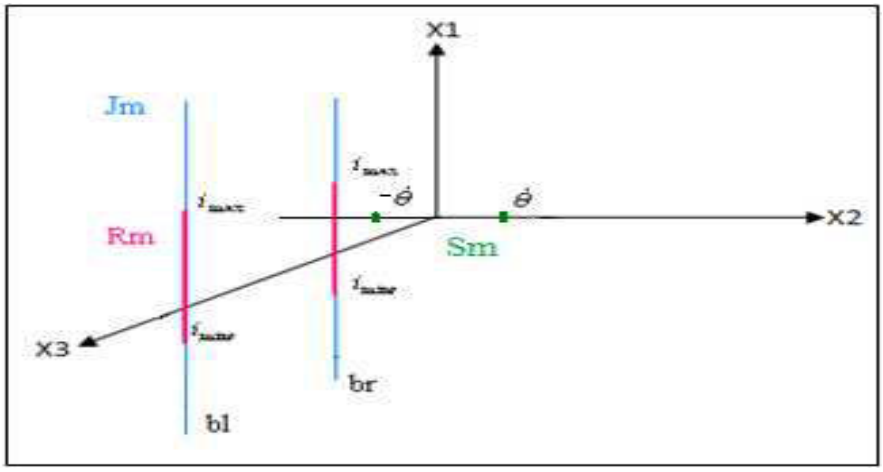


Figure 19 Les différents ensembles d'une inclusion différentielle impulsionnelle dans notre exemple d'application.

Dans cette application l'ensemble des transitions discrètes Jm est constitué de deux droites d'équations respectives $x_3 = b_l$ et $x_3 = b_r$. Donc à chaque fois que le système se place sur un point appartenant à une de ces deux droites, les transitions discrètes sont forcées et la tension U change de signe et donc le système rebrousse chemin.

L'ensemble de reset Rm ou « la remise à zero » permet les transitions discrètes donc il peut être contenu dans l'ensemble Jm, et de plus la vitesse angulaire au moment des transitions s'annule d'où $x_2 = \dot{\theta} = 0$. A cet instant i varie entre $i_{\min} \; et \; i_{\max}$.

L'ensemble de commutation (switch) est le vecteur d'état juste après le franchissement de la transition. Par exemple, lors de la commutation à l'instant où $x_3 = b_r$, le vecteur d'état x_τ juste avant le franchissement et juste après, est :

$$x_\tau = \begin{Bmatrix} x_1 = i_{\max} \\ x_2 = \dot{\theta} \\ x_3 = b_r \end{Bmatrix} \text{ et } \quad x_{\tau'} = \begin{Bmatrix} x_1 = i_{\min} \\ x_2 = 0 \\ x_3 = b_r \end{Bmatrix} \qquad (3.7)$$

D'où l'ensemble Sm.

3- Proposition d'un algorithme d'inclusion différentielle impulsionnelle à 1 paramètre variable

Dans cette section on présente une des contributions. Il s'agit d'un algorithme pour la construction d'une inclusion différentielle impulsionnelle. Cet algorithme utilise l'algorithme d'inclusion différentielle déjà obtenu au chapitre précédent et étend le domaine d'application aux automates hybrides. En effet, à chaque pas on doit résoudre le problème des inclusions différentielles ainsi que les commutations.

Dans cet algorithme on suppose que la variable v varie dans un intervalle $[v_{\min}, v_{\max}]$.

Dans cet algorithme on fait appel à des fonctions utilisées dans l'algorithme de résolution de l'inclusion différentielle (développé dans le chapitre 2 Inclusion Différentielle) comme la maximisation de l'Hamiltonien et le calcul de pas de variation z.

Cet algorithme permet la simulation de fonctionnement d'un automate hybride. On donne les étapes à suivre pour chaque situation. Comme l'inclusion différentielle impulsionnelle est une extension des inclusions différentielle du domaine continu au domaine hybride on va réutiliser tout ce qu'on a développé dans le chapitre précédent sur les inclusions différentielles.

On suppose que les situations de l'automate hybride sont définies par une inclusion différentielle (sinon une équation différentielle est une inclusion différentielle avec une variation nulle). On commence pour chaque situation par résoudre l'inclusion différentielle. Le domaine atteignable résultat de l'inclusion différentielle sera défini de l'instant de

démarrage de la situation à l'instant de son arrêt. En effet, l'évolution continue de la situation s'arrête en deux cas : la transition est franchie ou le système bloque. On récupère l'instant de franchissement de transition pour arrêter l'évolution continue, générer le domaine atteignable et passer à la situation suivante.

L'algorithme ci-dessous utilise l'algorithme de SDOGD (chapitre 2 paragraphe 5). Il présente une méthode générale de constructions des différents ensembles de l'inclusion différentielle impulsionnelle.

Début

Étape 1 : Déclaration d'INIT : la situation initial X_0

(déclaration de la situation initial du système : les conditions initiales au démarrage du système à t=0)

Étape 2: Déclaration des différents ensembles X, F, R et J

Étape 3*:*

Tant que $x(t) \in F \setminus J$,

* Evolution continue* :

-Déclaration des inclusions différentielles F de la situation

- Déclaration des équations différentielles

- Déclaration des conditions initiales

- Déclaration des paramètres variables (exemple v)

- Déclaration d'intervalle de variation (exemple $[v_{\min}, v_{\max}]$)

* Résolution de l'inclusion différentielle et utilisation de l'algorithme SDOGD présenté au chapitre 2 de l'inclusion différentielle *

Initialiser $v_{f,1} = v_a$

m=1

Tant que $(v \geq v_a \quad et \quad v \leq v_b)$

k=1

On calcule $x_{1,m} = x(v_{1,m})$

On calcule $v_{2,m} = v_{1,m} + \lambda_1 \nabla H(v_{1,m})$

On calcule $x_{2,m} = x(v_{2,m})$

Tant que $\left\| H(v_{k+1,m}) - H(v_{k,m}) \right\| > \xi$

On calcule $\lambda_{k,m} = \beta \dfrac{1}{\left\| x_{k+1,m} - x_{k,m} \right\|}$

On calcule $v_{k+1,m} = v_{k,m} + \lambda_{k,m} \nabla H(v_{k,m})$

k=k+1

Fin Tant que

Si l'optimum est un minimum alors « steepest Ascent »

Sinon « steepest Descent ».

$v_{f,m} = v_{k+1,m}$

m=m+1

Fin Tant que

Fin Tant que

Étape 4:

Tant que $R(x) \neq \phi$ *ou* $x(t) \in J$

Evolution discrète :

-Vérification des conditions aux limites
-Evolution discrète vers la situation suivante

Fin tant que

Étape 5 : Génération des résultats:

****Fin****

4- Proposition d'un algorithme d'inclusion différentielle impulsionnelle avec n paramètres variables

On présente notre l'algorithme de l'inclusion différentielle impulsionnelle et on l'étudie avec n paramètre variable avec n ≥ 2. L'automate hybride sera la même mais à l'intérieur de la situation on pourra avoir n paramètres variables.

La généralisation du premier algorithme pour n paramètres variables n'entraine pas une grande modification sur l'algorithme mais plutôt sur sa programmation.

Avec n paramètres variables, on a un nombre élevé de combinaisons possibles à balayer, ce qui demande un temps de calcul plus important.

L'implémentation de ces algorithmes sera développée dans le chapitre 4 « Outils de programmation et implémentation des algorithmes » ainsi que les résultats obtenus.

Début

Étape 1 : Déclaration d'INIT : la situation initial X_0

(déclaration de la situation initial du système : les conditions initiales au démarrage du système à t=0)

Étape 2: Déclaration des différents ensembles X, F, R et J

Étape 3:

Tant que $x(t) \in F \setminus J$,

* Evolution continue* :

- Déclaration des n inclusions différentielles de la situation
- Déclaration des équations différentielles
- Déclaration des conditions initiales
- Déclaration des n paramètres variables
- Déclaration des n intervalles de variation

* Résolution des n inclusions différentielles en utilisant l'algorithme SDOGD présenté au chapitre 2 de l'inclusion différentielle *

Étape 3: Création d'un tableau qui contient toutes les solutions de l'inclusion différentielle .

Étape 4:

Tant que $R(x) \neq \phi \quad ou \quad x(t) \in J$

Evolution discrète :

-Vérification des conditions aux limites
-Evolution discrète vers la situation suivante

Fin tant que

Étape 5: Génération des résultats:

Étape 6: Construction et assemblages des résultats : événements et domaines atteignables pour chaque événement.

Fin

Comme pour l'algorithme précédent, celui-ci donne une méthode générale de construction des différents ensembles F, R et J de l'inclusion différentielle impulsionnelle.

5- Cas d'une inclusion différentielle périodique : discussion sur le temps de commutation

5.1- Analyse de l'intervalle de transition

Dans le cadre de l'inclusion différentielle impulsionnelle la transition n'est pas à la même date pour toutes les valeurs d'un paramètre ; cette date appartient à un intervalle de temps.

En effet, dans le cas où l'évènement déclencheur de transition est un évènement interne dépendant de la dynamique du système, la transition n'est pas à la même date ; elle est franchie à une date différente pour chaque valeur de paramètre.

La transition s'étale sur un intervalle du temps $\left[\tau_i, \tau_{i+1}^{'}\right]$, ce même intervalle est une partition de petits intervalles.

$$\left[\tau_i, \tau_{i+1}^{'}\right] = P_i^{i+1}\left[\delta_j, \delta_j^{'}\right] \qquad (3.8)$$

Chaque partition de l'intervalle de transition $[\delta_j, \delta_j^{'}]$ correspond à la durée de franchissement de la transition d'une seule trajectoire (qui correspond à une variation) .

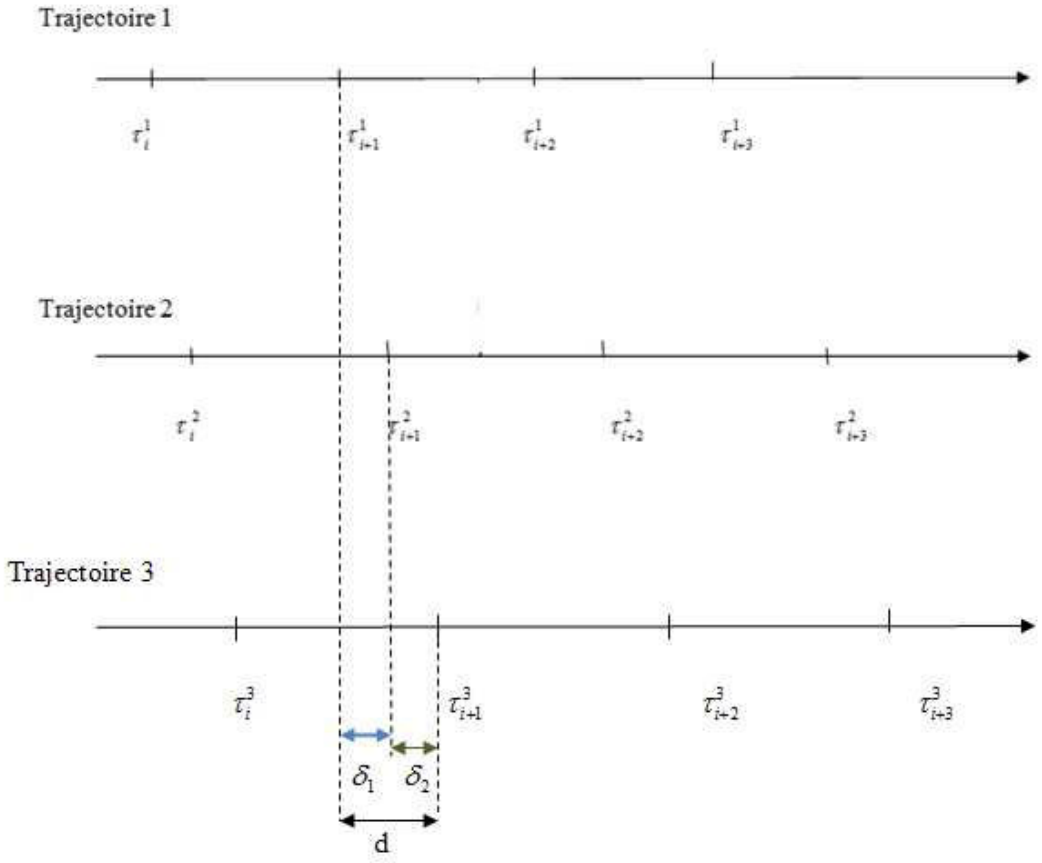


Figure 20 Exemple de trajectoires d'un système avec les instants de commutation

<u>Création d'une suite d'intervalle</u>

$$I_i = [\tau_i^{\min}, \tau_i^{\max}], \quad I_{i+1} = [\tau_{i+1}^{\min}, \tau_{i+1}^{\max}] \qquad (3.9)$$

Soit :

$$f : N \to R \times R$$
$$i \to [\tau_i^{\min}, \tau_i^{\max}]$$

f est une application de N dans R^2 qui à chaque cycle i associe un intervalle de transition $\left[\tau_i^{\min}, \tau_i^{\max}\right]$

La longueur de l'intervalle est $d_i = \tau_i^{\max} - \tau_i^{\min}$

Existence de l'enveloppe dans le temps

Dans le cas d'IDI cyclique, si elle existe, la solution dans l'espace d'état est bornée quel que soit t_i et x_i. En effet, les conditions initiales sont remises à zéro à chaque cycle. Par conséquent, l'enveloppe de la solution de l'IDI dans le temps existe.

Si $d_i \to \infty$, alors la longueur de l'intervalle $d_i = \tau_i^{\max} - \tau_i^{\min} \to \infty$. Donc le système est divergent et ne commute plus pour certaines valeurs de paramètres.

Si $\forall I_i, d_i \in R,$ alors si $\lim_{i \to \infty} d_i = \infty$ alors le système est divergent en temps de réponse .

Pour avoir une IDI cyclique, bornée et convergente Il faut que $\bigcup I_i$ soit une partition de R

C à d : $\bigcup I_i = R \quad et \quad \bigcap I_i = \Phi$

La réunion des intervalles I_i peut s'étendre jusqu'à tout R, alors que l'intersection entre les intervalles I_i doit être l'ensemble vide pour éviter le recouvrement.

Si $\bigcup I_i = R$ alors dans ce cas les intervalles sont disjoints et forment une partition de R

Si $\bigcup I_i \supset R$ alors d_i est très grand et on a un recouvrement

5.2- Analyse du temps de réponse

Définition du temps du cycle

Le temps du cycle du système est le temps nécessaire pour l'exécution d'un cycle d'aller retour.

Soit :

- T la durée d'un cycle d'aller-retour pour la même valeur du paramètre.
- T_{max} : la durée maximale d'un cycle

- T_{min} : la durée minimale d'un cycle
- $\Delta t = T_{max} - T_{min}$:
- n : le nombre de cycle

Exemple du temps de cycle de l'inclusion différentielle impulsionnelle périodique :

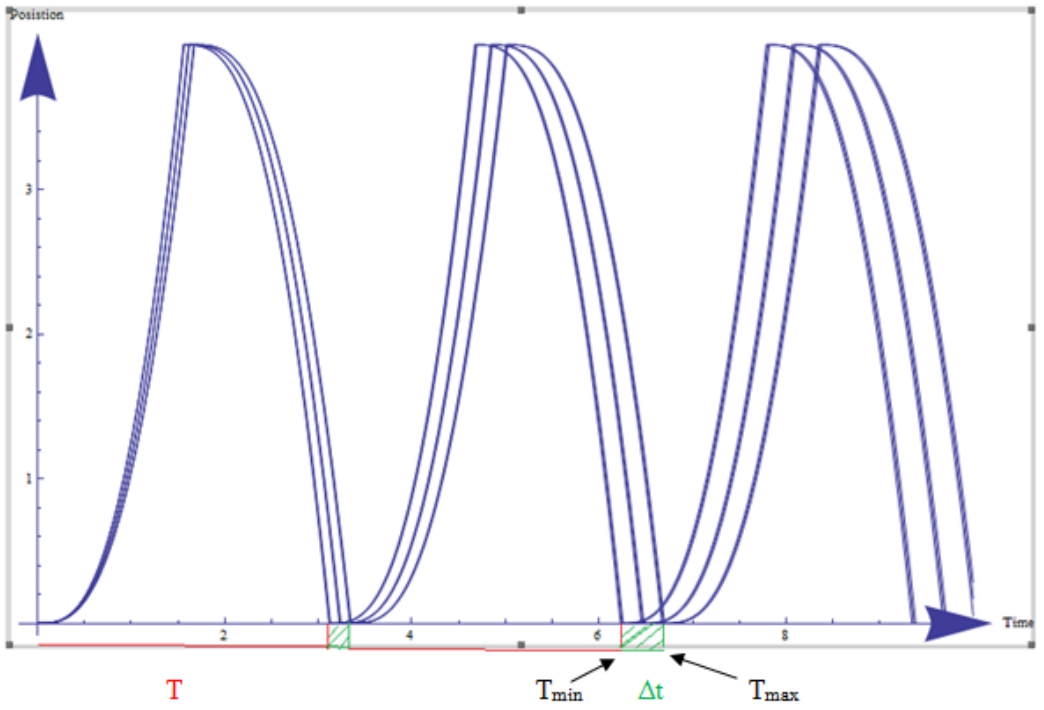


Figure 21 Temps de réponse de système

Dans notre exemple on voit clairement que la durée d'un cycle est toujours la même ; en revanche ce qui varie est le temps total pour n cycles T_N du système. En effet :

$$T_N = n \cdot (T + \Delta t) \qquad (3.10)$$

Notre système est cyclique. L'IDI est bornée dans l'espace d'état par les contraintes physiques du système qui sont traduites par des conditions initiales réinitialisées à chaque début du cycle. Dans ce cas, plus le nombre de cycles n augmente, plus le temps total de système augmente. Alors notre système est convergeant dans l'espace d'état mais divergeant dans le temps.

6- La théorie de viabilité associée à l'inclusion différentielle impulsionnelle

Dans cette partie nous allons compléter par des notions de viabilité et d'invariance qui sont liées à des domaines de R^n générés par des variations d'entrée et de paramètres. Ces domaines pourront être des propriétés pour le tolérancement attendu du système que nous traiterons au chapitre 6.

6.1- Un fonctionnement viable

Un fonctionnement (τ,x) d'une inclusion différentielle impulsionnelle H= (X, F, R, J) est dit viable dans un domaine $K \subseteq X$ si $x(t) \in K$ pour tout $I_i \in \tau$ et tout $t \in I_i$.

Dans la définition ci-dessus d'un fonctionnement viable, l'état reste dans un domaine K, tout au long l'évolution continue, durant la transition discrète et après la transition discrète [Aubin, 2002b], [Lygeros, 2004], [Lygeros, 2010].

En se basant sur cette définition on peut définir deux classes de domaines :

- Ensemble viable / ensemble invariant
- Noyau de viabilité / noyau d'invariance

6.2- Un ensemble Viable / Invariant

Un ensemble $K \subseteq X$ est dit viable sous une inclusion différentielle impulsionnelle H= (X, F, R, J) , si pour tout $x_0 \in K$ il existe un fonctionnement viable dans K et dont le point initial est x_0.

K est dit invariant sous l'inclusion différentielle impulsionnelle , si pour tout $x_0 \in K$ tous les fonctionnements dont le point initial est x_0 sont viables dans K.

Dans le cas où l'inclusion différentielle impulsionnelle ne satisfait pas les conditions de viabilité et d'invariance, on pourra établir un ensemble de conditions initiales pour satisfaire les conditions. Cette notion sera définie par le noyau viable/ invariant.

6.3- Noyau de viabilité /d' invariance :

Un noyau viable $Viab_H(K)$ d'un ensemble $K \subseteq X$ sous une inclusion différentielle impulsionnelle H(X, R, J, F) est l'ensemble d'états $x_0 \in X$ pour lequel il existe un fonctionnement viable dans K.

Un noyau invariant Inv_H (K) d'un ensemble $K \subseteq X$ sous une inclusion différentielle impulsionnelle H(X, R, J, F) est un ensemble des états $x_0 \in X$ pour lequel tous les fonctionnements sont viables dans K.

Par définition $Viab_H(K) \subseteq K$ *et* $Inv_H(k) \subseteq K$ mais ils ne sont pas liés simplement [Aubin, 2001].

On trouve une application de cette théorie de viabilité et d'invariance sur les différents ensembles de l'inclusion différentielle impulsionnelle dans [Lygeros, 2004], [Aubin, 1991], [Aubin, 1990]

7- Méthodes de calcul et d'approximation du noyau de viabilité pour les systèmes dynamiques

Dans [SAI, 94], on propose une méthode constructive qui permet l'approximation du noyau de viabilité. On se base sur les théorèmes proposés dans [Aubin, 1991] pour introduire une relation de récurrence permettant de construire le noyau de viabilité [Aubin, 2011], [Aubin, 2013].
La deuxième méthode possible est de calculer le noyau de viabilité non pas comme le domaine de viabilité le plus grand mais comme l'ensemble regroupant tous les points à partir desquels est issu une solution viable. Dans le cas continu on peut trouver les propriétés nécessaires dans [Mattioli, 96].
Dans le cas des systèmes dynamiques continus définis par des équations différentielles ou des inclusions différentielles le calcul exact du noyau de viabilité présente une tâche complexe. Dans [Mattioli, 2003] et [Bernard, 2011] on trouve un algorithme d'approximation du noyau de viabilité ainsi que des exemples d'applications.

La théorie de viabilité a été appliquée dans différents domaines. Initialement, elle était centrée sur les systèmes évolutionnaires en sciences économiques, puis, elle s'est considérablement

élargie dans les années 90 aux sciences sociales (économie dynamique [Aubin, 1997], démographie [Bonneuil,1997], finance [Doyen, 1996], environnement [Doyen,1996, Doyen,1997], . . .) et aux sciences de l'ingénieur (contrôle [Quincampoix 91, Saint-Pierre, 1998], jeux différentiels [Cardaliaguet, 1998], optimisation continue [Gorre, 1996, Aubin, 1998], optimisation de formes [Aubin,1999]).

L'objet de cette théorie mathématique est d'offrir une boîte à outils permettant l'étude de l'évolution de systèmes. Elle permet donc de rechercher les conditions dans lesquelles les contraintes opérationnelles seront toujours satisfaites et donc dans lesquelles le système pourra fonctionner de manière durable. Le tolérancement sur les performances ou sur les paramètres, présente des contraintes du système. La théorie de viabilité consiste à trouver l'ensemble K de toutes les trajectoires de point initial x_0 dont l'évolution reste toujours dans ce domaine K.

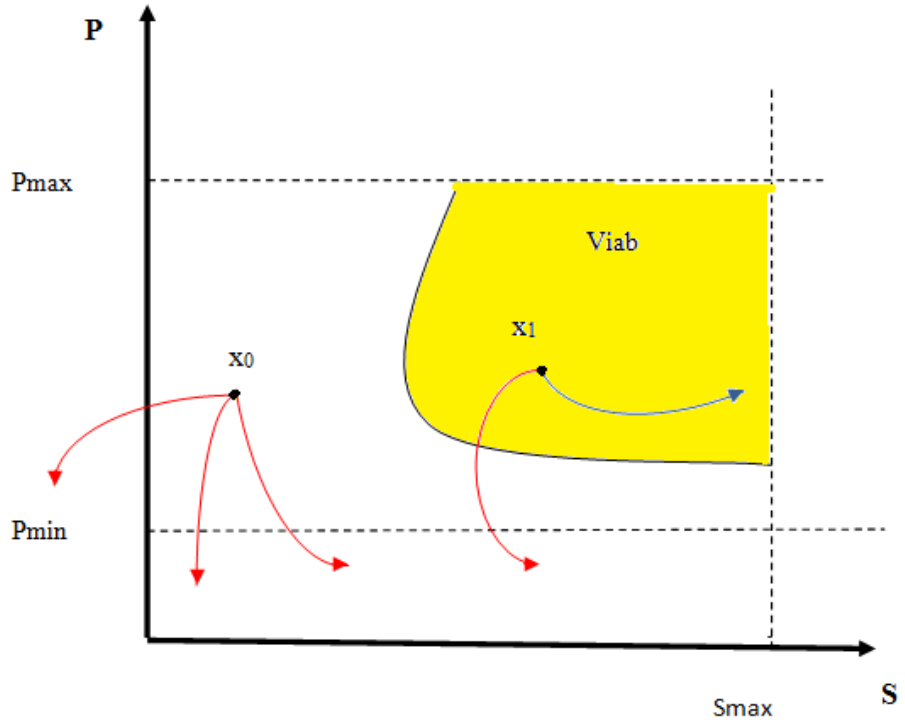


Figure 22 Domaine de viabilité sous contraintes

Dans la figure 22, on considère des contraintes modélisées par P_{max}, P_{min} et S_{max} , le système viable doit évoluer dans ce domaine délimité par ces contraintes. On considère aussi deux points initiaux x_0 et x_1, on remarque que toutes les trajectoires au départ de x_0 violent les contraintes alors que ce n'est pas le cas pour les trajectoires au départ de x_1. Ceci permet alors d'avoir le noyau de viabilité « Viab » qui contient toutes les trajectoires viables du système.

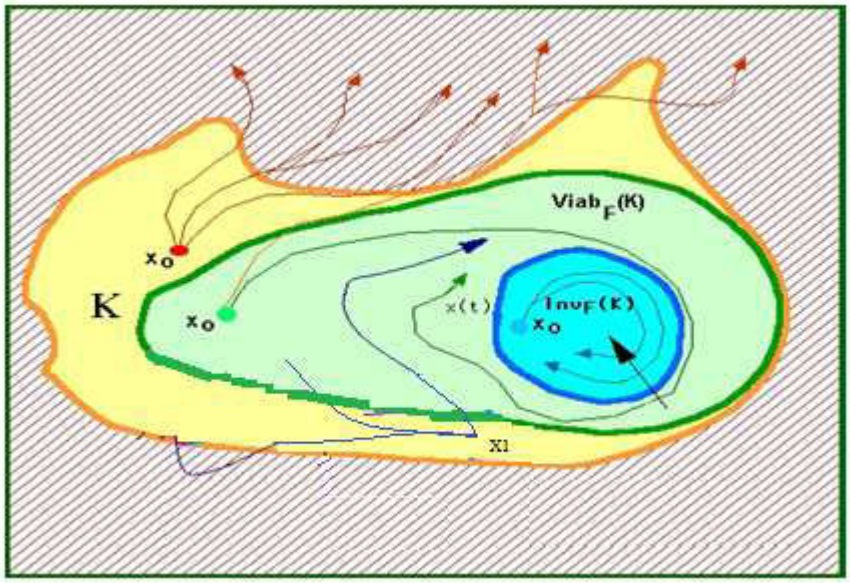


Figure 23 Exemple de noyau de viabilité et d'invariance [Aubin, 2011]

Dans le chapitre 6 sur les perspectives en tolérancement et le développement de nos travaux sur les propriétés du système nous reprendrons et utilisons les notions de noyaux de viabilité. Cette notion est utile au problème du tolérancement dans le cas où on ne peut pas agir sur l'intervalle de variation des paramètres ni sur les caractéristiques du système. Ce cas sera repris et expliqué dans le paragraphe 4.3 du chapitre 6.

Conclusion

Dans ce chapitre nous nous sommes intéressés à l'étude des systèmes hybrides à paramètres variables. Nous avons commencé par une présentation de l'outil d'automate hybride comme outil de modélisation des systèmes hybrides à paramètres fixes. Cet outil est constitué des états d'évolution continue, des transitions discrètes, des sauts, des gardes,… Pour modéliser des systèmes hybrides à paramètres variables nous avons choisi l'inclusion différentielle impulsionnelle. Cet outil utilisé dans le domaine des systèmes hybrides et présenté par Aubin permet la modélisation des automates hybrides dont les situations contiennent des inclusions différentielles. Comme les automates hybrides, les inclusions différentielles impulsionnelles contiennent des situations d'évolution continue et des transitions discrètes. Nous avons proposé notre algorithme pour la définition des ensembles de l'IDI : l'ensemble des variables d'état, l'inclusion différentielle, l'ensemble de remise à zéro et l'ensemble des transitions discrètes. Nous avons repris ces ensembles et nous avons proposé une identification sur un exemple mécatronique de pilotage.

Pour simuler les résultats, nous avons développé des algorithmes de résolution des inclusions différentielles impulsionnelles avec 1 ou n paramètre(s) variable(s).

Dans le cas particulier d'inclusion différentielle impulsionnelle cyclique, nous avons établi une analyse du temps de commutation ce qui permet d'étudier le temps de réponse du système et de discuter ses propriétés de nombre de cycles.

Nous avons repris la notion de la théorie de viabilité ainsi que l'ensemble variable et invariant et les noyaux de viabilité et d'invariance. Cette théorie ne se limite pas au domaine de mathématique, elle est utilisée dans différentes applications dans le domaine de l'automatique, et de l'économie par exemple. Cette notion peut être utilisée dans le cadre du tolérancement pour limiter une certaine zone du fonctionnement du système soumise à des contraintes.

Notre objectif à ce stade est de spécialiser et appliquer ces algorithmes à la mécatronique. Cela sera l'objectif du chapitre suivant. On va appliquer ces algorithmes sur un exemple mécatronique et on va simuler les résultats sous un logiciel adéquat.

Chapitre 4 : Outils de programmation et implémentation des algorithmes

Introduction

Après avoir développé la partie théorique de l'étude des variations paramétriques, il nous faut un outil adéquat pour simuler l'effet de ces variations sur le comportement d'un système mécatronique. Dans ce chapitre on définit différents outils de programmation, on commence par Modelica qui est un logiciel de simulation orienté objet. Ensuite, on introduit l'environnement très répandu Mathlab/Simulink puis Mathematica qui est un outil de calcul symbolique, numérique et un éditeur graphique qui requiert plusieurs options. On présente aussi Mathmodelica et le SystemModeler qui sont des langages regroupant à la fois les caractéristiques de Modelica et Mathmodelica. On justifie notre choix de Mathématica.

Dans un second temps, on choisit un exemple d'application, il s'agit d'un système de pilotage mécatronique. On y applique l'algorithme de résolution d'inclusion différentielle et l'algorithme de résolution d'inclusion différentielle impulsionnelle. On simule les résultats sous l'outil choisi Mathematica en détaillant quelques fonctions utilisées.

1- Choix du langage de simulation

1.1- Modelica

Modelica est un langage orienté objet dont le but est de modéliser et de simuler des systèmes physiques complexes comprenant des composants électriques, mécaniques, hydrauliques et thermiques. Le développement et la promotion de Modelica sont assurés par l'association à but non lucratif OpenModelica.

Les quatre aspects les plus importants de Modelica sont **[Fritzon, 2003]** :

- Il est principalement basé sur des équations et non sur des instructions d'affectation. Ceci permet une modélisation qui favorise une meilleure réutilisation des classes puisque les équations ne spécifient pas de direction de flux de données.
- Il a la capacité de fournir une modélisation multi domaines. Les modèles de composants correspondant aux objets physiques de plusieurs domaines différents, électrique, mécanique, thermodynamique, hydraulique, biologique et des applications de contrôle peuvent être décrits et connectés.
- C'est un langage orienté objet avec un concept de classe générale qui englobe les classes. Ceci facilite la réutilisation des composants, l'évolution des modèles et le lie à des modèles d'analyse (UML/SysML) **[Turki, 2008]**.

Sous Modelica, le paradigme orienté objet est vu comme un concept structurant utilisé pour traiter la complexité des systèmes. Un modèle Modelica est une description mathématique déclarative. Les propriétés des systèmes dynamiques sont exprimées par des équations d'une façon déclarative.

Le concept de programmation déclarative est inspiré des mathématiques. Ce paradigme de programmation traite le calcul comme l'évaluation des fonctions mathématiques, au contraire d'une programmation procédurale où le programme contient une série d'étapes ou

d'instructions à réaliser montrant comment atteindre le but final. L'approche déclarative orientée objet de la modélisation des systèmes et de leurs comportements est d'un niveau d'abstraction plus haut que celui de l'approche orientée objet ordinaire puisque quelques détails d'exécution peuvent être omis. Par exemple, on n'a plus besoin de code pour transporter explicitement les données entre les objets à travers des assignations ou des opérations de passage de variables. Un tel code est généré automatiquement par le compilateur Modelica en se basant sur les équations données.
Comme dans les langages orientés objet, les classes sont des modèles de base pour créer les objets. Les variables, les équations et les fonctions peuvent être héritées par d'autres classes.

Les programmes Modelica contiennent des classes, qui sont appelées aussi des modèles. A partir de la définition d'une classe, on peut créer un certain nombre d'objets qui sont des instances de cette classe. Une classe Modelica contient des éléments, des déclarations de variables et des équations, en particulier des équations différentielles. Les variables contiennent des données appartenant aux instances de la classe. Elles composent le stockage de données de l'instance. Les équations d'une classe spécifient le comportement des instances de cette classe [Hadj Amor, 2009].

1.2- OpenModelica

OpenModelica a deux buts à court terme et à long terme[Fritzson, 2006]:
- Le but à court terme est de développer un environnement informatique interactif efficace pour le langage Modelica ainsi que de fournir une implémentation complète du langage.
- Le but à long terme est d'avoir une implémentation de référence complète du langage Modelica et d'inclure également la simulation des modèles basés sur les équations et des facilités additionnelles dans l'environnement de programmation ainsi que le développement des spécifications complètement formelles pour le langage Modelica dont les sémantiques statiques et dynamiques.
La version courante de l'environnement OpenModelica permet l'exécution interactive de la plupart des expressions, algorithmes et des parties de fonctions de Modelica, ainsi que la génération du code C efficace à partir des modèles d'équations et des fonctions Modelica. Le code C généré est combiné avec une librairie de fonctions utilitaires, une librairie run-time et un solveur numérique DAE.

Vue d'ensemble du système

L'environnement OpenModelica consiste en plusieurs sous-systèmes connectés. Les sous-systèmes suivants sont actuellement intégrés [Fritzon, 2007]:

- Système de traitement interactif de session : analyse la syntaxe et interprète les commandes et les expressions Modelica pour l'évaluation, la simulation, l'affichage courbes.
- Un compilateur Modelica : traduit le code Modelica en code C avec une table symbolique contenant les définitions des classes, des fonctions et des variables. Ces définitions peuvent être prédéfinies, définies par l'utilisateur ou obtenues à partir des librairies. Le compilateur contient aussi un interpréteur Modelica pour l'usage interactif et l'évaluation des expressions constantes.
- Un module d'exécution : ce module exécute le code binaire compilé à partir des expressions et des fonctions traduites, aussi bien que le code de la simulation à partir des modèles basés sur les équations, lié avec les solveurs numériques.
- Un éditeur/explorateur d'un modèle textuel Emacs : N'importe quel éditeur de texte peut être employé. *Gnu-Emacs10* a été utilisé parce qu'il peut être programmé pour des futures extensions. Le mode Emac cache les annotations graphiques pendant la rédaction, ce qui rend la lecture plus facile.
- Un éditeur/explorateur plugin Eclipse : Le plugin Eclipse est appelé MDT (Modelica Development Tooling) fournit un explorateur hiérarchique des classes et des fichiers ainsi que des avantages lors de l'édition des modèles Modelica comme celles pour Emacs.
- Un éditeur de modèle OMNotebook DrModelica : Ce module fournit un éditeur calepin. Cette fonctionnalité permet de traiter le tutorial DrModelica.
- Un éditeur/explorateur de modèles graphiques : C'est un éditeur graphique de connexion pour la conception des modèles basés sur les composants en connectant les instances des classes Modelica et en explorant les librairies de modèles pour la lecture et la sélection des modèles de composants.
- Un programme de mise au point Modelica (debugger) : C'est le programme de mise au point conventionnel utilisant Emacs pour montrer le code source durant la progression, mettre des points de contrôle, etc. La trace arrière et les commandes d'inspection sont disponibles.

1.3- Matlab/Simulink

MATLAB est un logiciel de calcul matriciel. Avec ses fonctions spécialisées, MATLAB peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques. Il interprète les instructions et les exécute ligne par ligne ainsi qu' au fur et à mesure qu'elles sont données par l'usager.

Il peut fonctionner dans plusieurs environnements tels que X-Windows, Windows, Macintosh. MATLAB exécute les instructions mode exécutif: MATLAB exécute ligne par ligne.

Simulink est un environnement de diagramme fonctionnel destiné à la simulation multidomaine et à l'approche de conception par modélisation Model-Based Design. Il prend en charge la conception et la simulation au niveau système, la génération automatique de code, ainsi que le test et la vérification en continu des systèmes embarqués.

Simulink propose un éditeur graphique, un ensemble personnalisable de bibliothèques de blocs et des solveurs pour la modélisation et la simulation de systèmes dynamiques. Il est intégré à MATLAB, ce qui permet d'incorporer les algorithmes MATLAB dans les modèles et d'exporter le résultat des simulations vers MATLAB pour compléter les analyses.

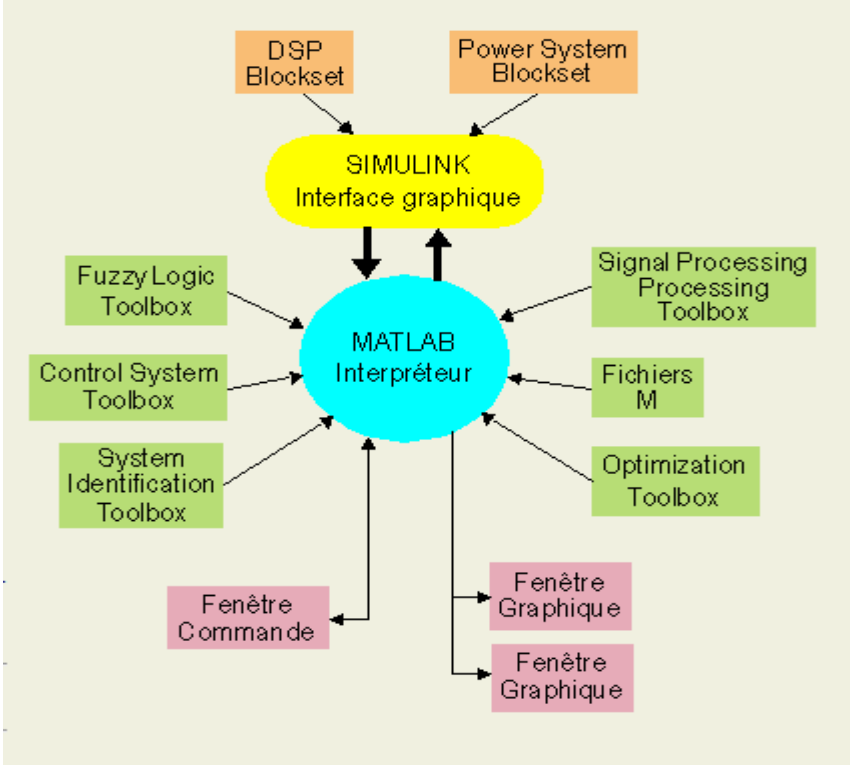


Figure 24 MAtlab/Simulink [Site web matlab]

1.4- Mathematica

Mathematica est un logiciel de calcul formel et numérique développé par Wolfram Research. Il permet essentiellement de faire du calcul formel (manipulation d'expressions mathématiques sous forme symbolique, par exemple : calcul de dérivées, de primitives, simplification d'expressions, etc.) et du calcul numérique (évaluation d'expressions mathématiques sous forme numérique; par exemple : calcul des premières décimales du nombre π, évaluation approchée d'intégrales, etc.). Mathematica incorpore un langage de programmation sophistiqué et permet aussi de faire des graphiques. C'est un logiciel très utilisé en enseignement, dans la recherche scientifique et dans l'industrie.

Figure 25 Exemple de résultats sous Mathematica [Site web Mathematica]

Mathématica ne se limite pas au calcul formel. Il pourra être utilisé comme [Site Web Mathematica] :

- Un calculateur numérique et symbolique.
- Un système de visualisation des fonctions et des données.
- Un langage de programmation de très haut niveau qui permet la création aussi bien de grands que de petits programmes.
- Un environnement de modélisation et d'analyse de données.
- Un système de représentation de données dans des domaines scientifiques et techniques.
- Un logiciel qui permet l'exécution des applications spécifiques.
- Un moyen de création de document interactif qui englobe texte, graphique animé, son et formules actives.

- Un langage de commande pour des programmes externes et des processus.

On peut dire que Mathematica est divisé en trois classes : calculateur numérique, calculateur symbolique et un éditeur graphique.

Mathematica utilise les expressions symboliques pour couvrir un grand nombre d'applications variées avec le minimum de méthodes mathématiques et algorithmiques.

Dans un cas de figure très simple, on pourra utiliser Mathematica comme un calculateur. On tape l'expression à calculer et Mathematica calcule le résultat et l'affiche en sortie « out ». Par contre Mathematica est beaucoup plus performant qu'un calculateur électronique traditionnel ou un langage de programmation tel que Fortran ou BASIC. Par exemple un système traditionnel contient 30 opérations mathématiques, Mathematica pourra contenir jusqu'à 750 opérations. En plus, les systèmes traditionnels calculent juste les équations numériques alors que Mathématica manipule aussi bien les systèmes numériques, symboliques et graphiques.

Ci-dessous quelques exemples sous Mathematica. Les lignes étiquetés par *In [n] :=* c'est ce que rentre l'utilisateur et les lignes étiquetés par *Out [n]* = représente le résultat fourni par Mathematica.

a- Le calcul numérique

Example: Find the numerical value of $\log(4\pi)$.

`Log[ 4 Pi ]` is the *Mathematica* version of $\log(4\pi)$. The `N` tells *Mathematica* that you want a numerical result.

```
In[1]:= N[ Log[ 4 Pi ] ]
Out[1]= 2.53102
```

Here is $\log(4\pi)$ to 40 decimal places.

```
In[2]:= N[ Log[ 4 Pi ], 40 ]
Out[2]= 2.531024246969290792977891594269411847798
```

Figure 26 Le calcul numérique sous Mathématica [Site web Mathematica]

Les calculateurs traditionnels manipulent des nombres avec un degré de précision fixe, alors qu'avec Mathematica on peut définir le degré de précision, de plus il inclut un nombre très élevé de fonctions mathématiques (intégrales elliptiques, fonctions de Bessel, fonctions hypergéométriques, et factorisation des entiers…)

Mathematica peut aussi appliquer les formules de calcul matriciel. Il supporte l'algèbre linéaire comme l'inversion des matrices, et le calcul des systèmes singuliers. Il pourra aussi manipuler les données numériques pour les analyses statistiques par exemple. Il est aussi performant pour les opérations comme les transformations de Fourrier, les interpolations et les opérations de moindre carré.

Mathematica peut faire des opérations numériques sur les fonctions comme les intégrations numériques, minimisation numérique ou la programmation linéaire. Il génère aussi des solutions numériques aussi bien pour les équations algébriques que pour les équations différentielles ordinaires.

b- Le calcul symbolique

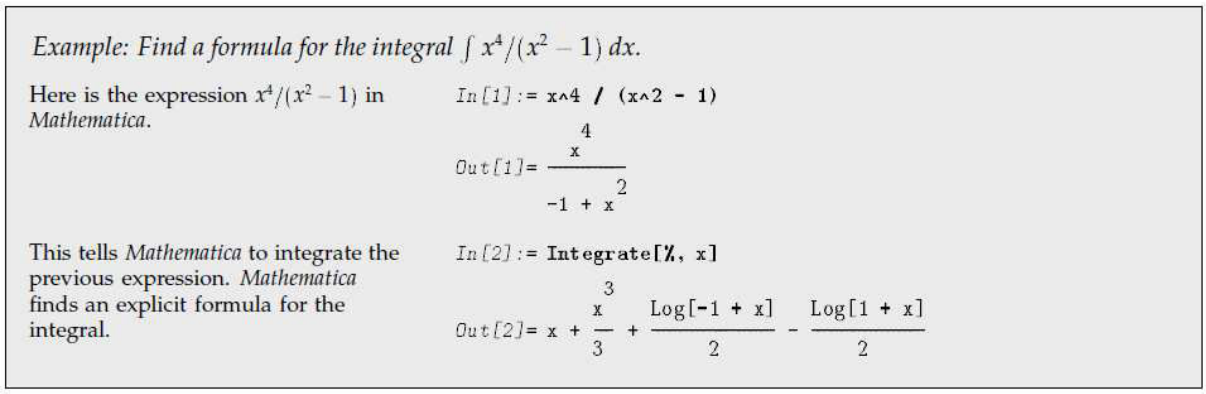


Figure 27 Le calcul symbolique sous Mathematica [Site web Mathematica]

L'une des principales performances de Mathematica dans le calcul symbolique est sa capacité à manipuler des formules algébriques. Il pourra exécuter différents types d'opérations algébriques. En effet, il est capable de développer, factoriser et simplifier des expressions rationnelles et polynomiales. Il pourra fournir des solutions algébriques pour des systèmes d'équations et pour les équations polynomiales.

Mathematica peut évaluer des dérivés et des intégrales symboliquement et fournir des solutions symboliques pour les équations différentielles ordinaires. Il peut dériver et manipuler les séries de puissance et d'approximation et trouver les limites. Mathematica s'étend à plusieurs domaines comme l'analyse vectorielle et la transformation de Laplace.

c- Le graphique

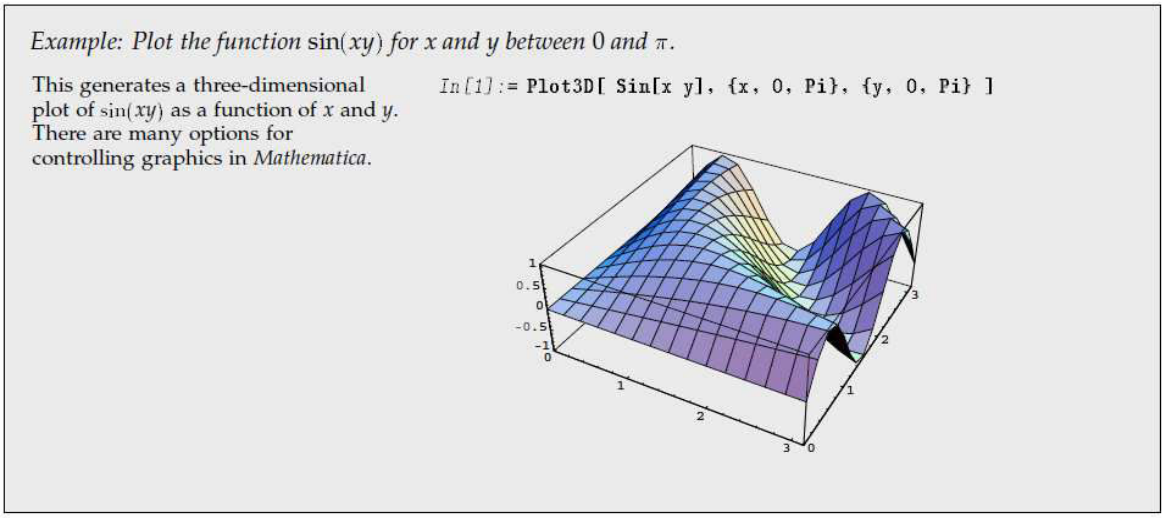


Figure 28 Graphique en 3D sous Mathematica v

Mathematica produit des graphiques de dimension 2 et 3, des contours et des graphiques de densité. On pourra avoir sur la même figure un graphique et une liste des données. Mathematica permet de contrôler les détails du graphique moyennant différents options. On pourra contrôler l'ombre, les couleurs, la luminosité, le maillage des surfaces …) les dernières versions de Mathematica supportent les graphiques animés.

Mathematica contient un langage graphique qui pourra donner une représentation symbolique des objets géométriques en utilisant les primitives comme les polygones. Tous les graphiques sont générés en PostScript standard et pourront être transférés en un grand nombre d'autres programmes.

d- Langage et interfaces de Mathematica

On pourra programmer différents types de programmes avec Mathematica :

- *La programmation des procédures* avec des structures en blocks, conditionnel, itérations et récursivité.
- *La programmation fonctionnelle* avec des fonctions pures, opérateurs fonctionnels, opérations du programme structuré.
- *La programmation qui se base sur des règles* avec une programmation orientée objet.

Mathematica est composé de deux parties : le noyau ("kernel") et l'interface graphique ("front end"). Le noyau constitue le cœur du logiciel; il interprète les instructions d'entrée (écrites en

langage Mathematica), puis calcule et retourne le résultat. L'interface graphique s'occupe de l'interaction avec l'utilisateur. Elle gère le fichier de travail (souvent appelé "feuille Mathematica" ou "notebook"), permet de taper les instructions et de visualiser les résultats. Le logiciel dispose aussi d'un traitement de texte, permettant ainsi d'inclure du texte parmi les calculs effectués. Plusieurs palettes d'outils sont disponibles pour aider à l'édition aussi bien de textes que d'expressions mathématiques (voir menu "Palettes").

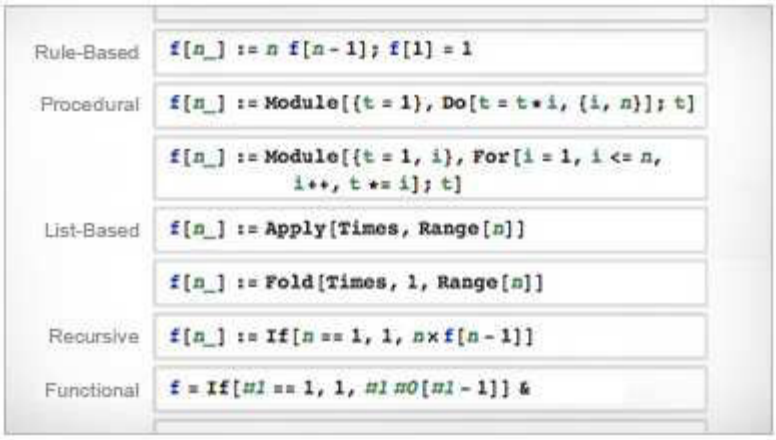


Figure 29 Exemples de fonctions sous Mathematica [Site web Mathematica]

1.5- Extension de Mathematica : Le SystemModeler

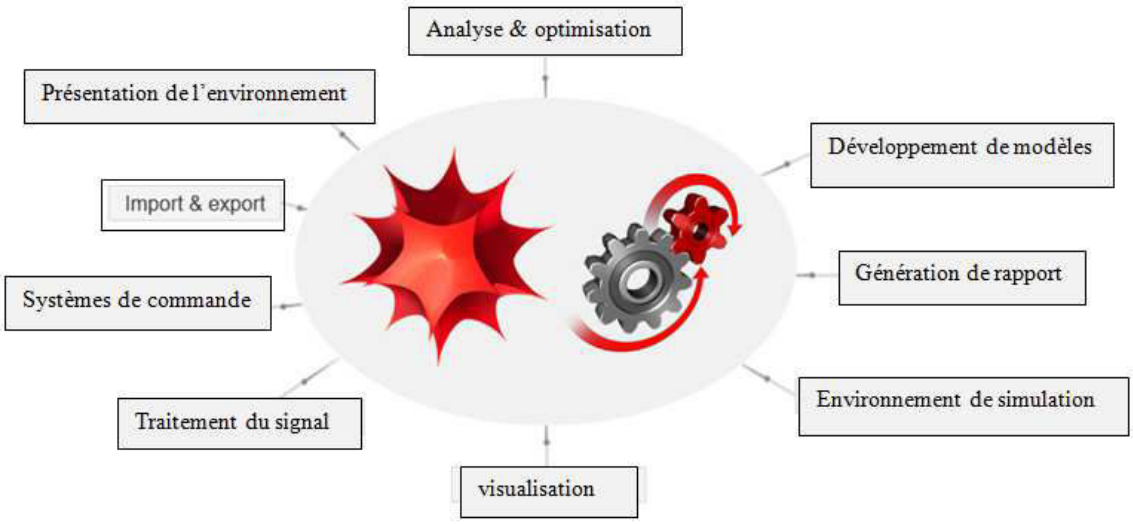


Figure 30 SystemModeler : combinaison de Mathematica et Modelica [Site web Mathematica]

Le SystemModeler implémente le langage Modelica. Il inclut les différentes librairies de Modelica. Le SystemModeler combine ainsi les avantages de Modelica et de Mathematica, il permet la modélisation en créant des systèmes à travers les composants dans les librairies

1.6- Extension de Mathematica : Mathmodelica

MathModelica est un environnement de développement interactif pour la modélisation et la simulation des systèmes avancés. Il est composé par trois sous systèmes qui sont utilisés pendant les différentes phases du processus de modélisation et de simulation **[Site Web Mathmodelica]**.

Les sous systèmes sont les suivants :

- Un éditeur graphique pour la conception des modèles à partir de la librairie des composants.
- Un bloc note interactif pour la programmation, la documentation et l'exécution des simulations.
- Le centre de la simulation pour l'initialisation des paramètres de la simulation, son exécution et l'affichage des courbes.

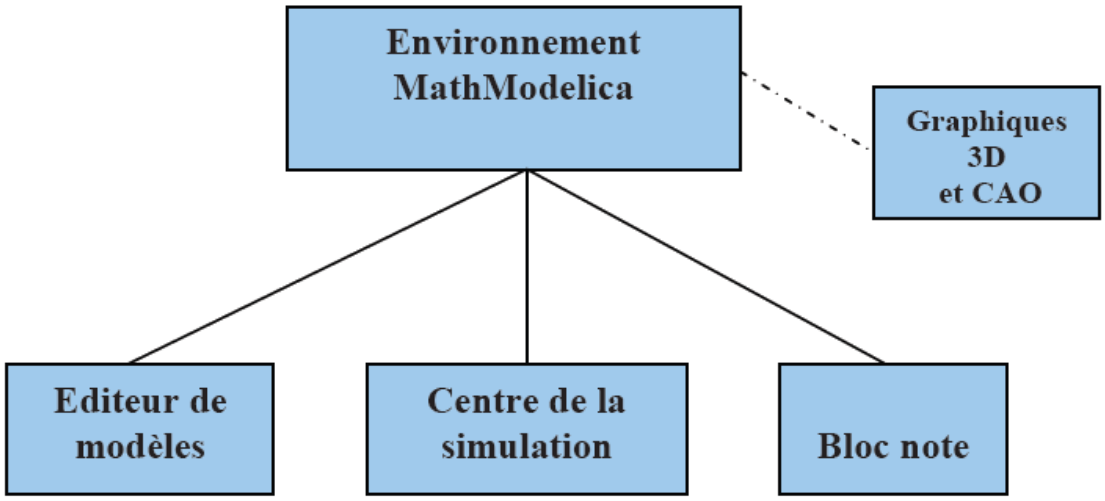


Figure 31 Architecture de l'environnement Mathmodelica

a- Editeur graphique des modèles

L'éditeur graphique des modèles MathModelica est une interface utilisateur où on peut construire des modèles en utilisant la méthode « glisser-déposer » à fin de glisser les classes des modèles de la bibliothèque standard de Modelica à partir des composants des librairies définis par l'utilisateur. Ces composants sont souvent représentés par des icônes graphiques dans l'éditeur. Cet éditeur peut être vu comme une interface utilisateur pour la programmation graphique sous Modelica.
Le centre de simulation est un sous-système pour exécuter les simulations, initialiser des valeurs et des paramètres du modèle, afficher des résultats.

b- Bloc note interactif

En plus de la programmation graphique utilisant l'éditeur des modèles, MathModelica fournit aussi un environnement de programmation pour créer des modèles Modelica textuels. Cet environnement textuel est le bloc note.

MathModelica est une extension de Modelica ciblée pour le travail dans l'environnement Mathematica. Ce langage est orienté objet (les programmes consistent en classes). Le langage est basé sur les équations au lieu des fonctions traditionnelles et des procédures. Nous utilisons les équations non-causales qui spécifient des relations algébriques et différentielles entre des variables numériques. La causalité d'entrée-sortie n'est pas spécifiée et donc ces équations peuvent être utilisées de façons multiples.

L'environnement intègre la plupart des activités nécessaires à la conception de simulation comme la documentation, le programme, le traitement symbolique et la transformation de formules, apport (saisie) et visualisation de données en sortie. Cet environnement de programmation peut être appliqué dans diverses applications de simulation [Fritzon, 1998].

2- Choix de l'outil

Après cette vue d'ensemble des différents langages, on avait le choix entre Modelica/Dymola, Matlab/Simulink ou Mathematica. Nos algorithmes nécessitent plusieurs manipulations mathématiques pour le calcul des inclusions différentielles et pour la programmation des modèles des systèmes hybrides. De plus on a besoin d'un outil ayant de bonnes qualités graphiques. Dans Mathematica, toutes les fonctions (de calcul, de graphique, …) sont

incluses (et vendues) sans avoir besoin à des toolbox supplémentaires. Avec Mathematica on peut manipuler des modèles de donnée de dimension n ≥ 2 sans avoir recours à un pointeur. Mathematica accepte aussi les équations différentielles d'ordre ≥2 sans les linéariser manuellement comme il peut gérer des intégrations d'ordre n ≥ 3. Il permet la programmation procédurale ainsi que la programmation fonctionnelle et orienté-objet

Mathematica est utilisé dans différents domaines et pour différentes applications comme par exemple dans le domaine de l'automatique où on pourra avoir des résultats assez précis sans avoir recours à des algorithmes très complexes.

Il est constitué d'une plateforme qui intègre des fonctionnalités pour différents domaines d'application. Il offre des méthodologies pour l'étude des différents problèmes hybrides avec les précisions requises.

Il contient une base de données énorme qui ne cesse d'être développée et mise à jour.

Le transfert des programmes de Mathematica vers Modelica est possible à travers Mathmodelica ou SystemModeler.

Cet outil répond à nos exigences, c'est pour cela qu'on va réaliser toutes les simulations des algorithmes sous Mathematica.

3- Exemple d'application (Rappel)

Pour mettre en œuvre notre approche et simuler les résultats obtenus, on a choisi un système mécatronique simple qui est un système de masse pilotée et guidée sur une vis à billes. Le système de pilotage linéaire est contrôlé par une commande. Ce dernier alimente un moteur DC par une tension (U_0, $-U_0$) selon la position δx de la masse en déplacement.

Le modèle physique du système de pilotage linéaire, représenté dans la figure suivante, est composé d'un moteur DC (Direct Current) à courant continu, une tige filetée et une masse pilotée en translation.

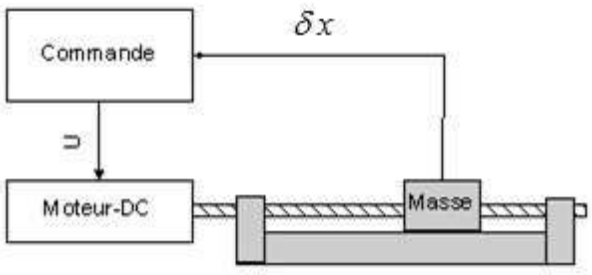


Figure 32 Système linéaire de pilotage

Les équations dynamiques du système sont : [Aublin, 1992]

$$\ddot{\theta} = \frac{d\dot{\theta}}{dt} = \frac{-b}{J+ma^2}\dot{\theta} + \frac{k}{J+ma^2}i \quad and \quad \delta x = a\theta$$
$$\frac{di}{dt} = \frac{-K}{L}\dot{\theta} - \frac{R}{L}i + \frac{V}{L} \qquad (4.1)$$

Cet exemple constitue un système mécatronique simple qui représente une application académique. Ce système simple constitue le système de base d'application réelle d'actionneur électromécanique par exemple aéronautique en translation vis-écrou qui peut être vu comme hybride (saturation de la commande) [Hospital, 2012].

4- Réalisation de l'algorithme de l'inclusion différentielle

On a appliqué l'algorithme développé dans le chapitre sur l'inclusion différentielle à notre exemple. Dans un premier cas de figure on fait varier le pas de la vis « a ». On a fait le test pour des grandes et des petites variations. On a appliqué à la résistance « R » des variations de même ordre de grandeur.

Lors de la programmation de l'algorithme sous Mathematica, on fait appel à la fonction « NDSolve » pour résoudre les équations différentielles. On définit l'Hamiltonien ainsi que les composantes du vecteur adjoint. On calcule le pas variable z en utilisant l'algorithme SDOGD (voir chapitre 2 des inclusions différentielles) et on génère les résultats.

On considère le vecteur d'état suivant :

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} i \\ \dot{\theta} \\ a\theta \end{bmatrix}$$

Avec :

- i l'intensité
- $\dot{\theta}$ la vitesse angulaire
- $a\theta$ la position

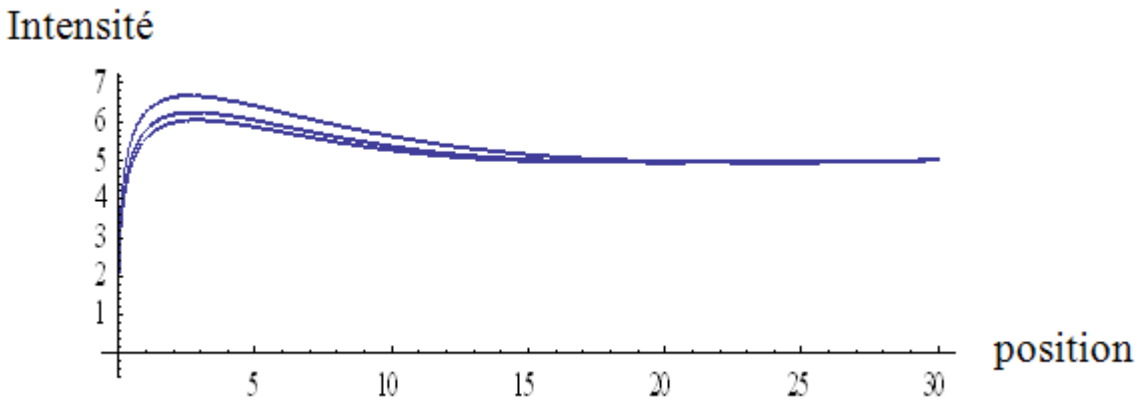


Figure 33 Projection du domaine atteignable sur le plan (x_1, x_3) avec variation de « a » de 10%

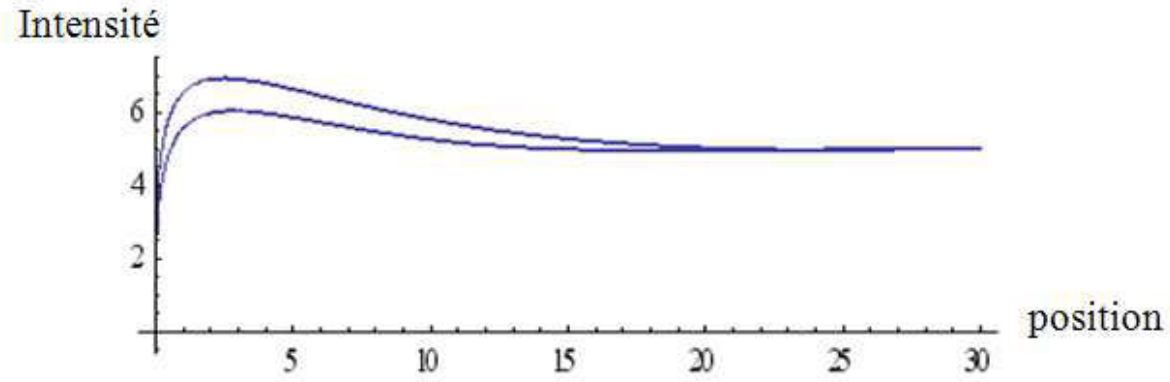


Figure 34 Projection du domaine atteignable sur le plan (x_1, x_3) avec variation de « a » de 50%

Dans un premier temps, on s'intéresse à la variation du pas de vis et à son influence sur le comportement du système. Tout d'abord on fixe la valeur nominale du pas de vis à 1. On considère une variation totale de 10% (c.à.d 1 ± 5%). Le domaine atteignable solution de l'inclusion différentielle a la forme d'un fuseau qui englobe tous les cas de figure possibles. Dans un second temps on fait varier le pas de la vis par 50% (c. à. d 1 ± 25%). Et on considère le nouveau domaine atteignable.

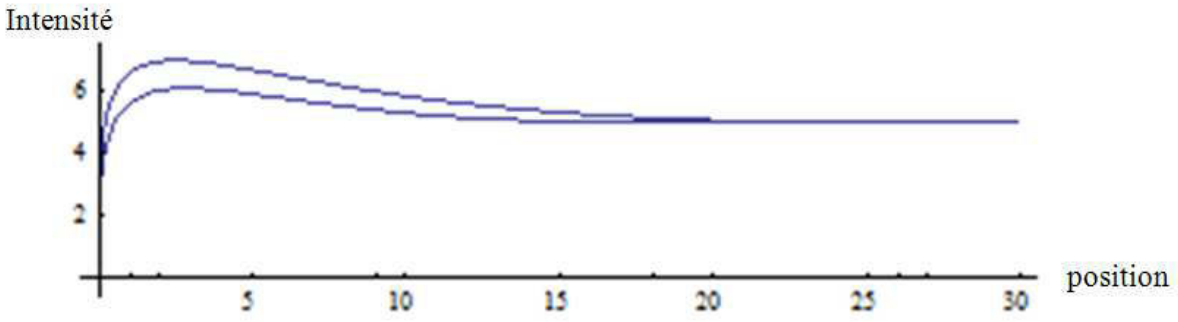


Figure 35 Projection du domaine atteignable sur le plan (x_1, x_3) avec variation de « R » de 10%

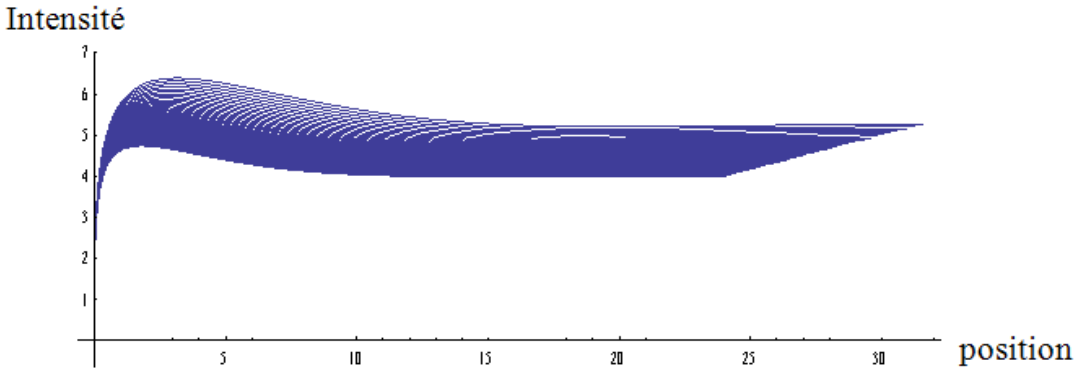


Figure 36 Projection du domaine atteignable sur le plan (x_1, x_3) avec variation de « R » de 50%

On a appliqué la même variation à la résistance « R ». On a pris une valeur nominale de « R » égale à 1. On a varié la résistance de 10%. On obtient un certain ensemble de courbes qui constitue le domaine atteignable. On a fait varier la résistance après de 50% par rapport à sa valeur nominale. On obtient en résultat un nouveau domaine.

En effet, l'augmentation de la résistance entraine une diminution de l'intensité et donc la masse se déplace plus lentement. Dans ce cas le système reste dans le domaine atteignable par petite variation mais il pourra ne pas la balayer en totalité.

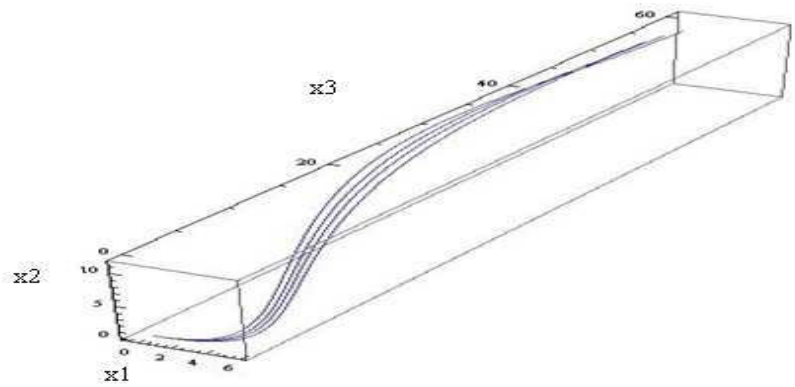


Figure 37 Domaine atteignable dans l'espace d'état avec une variation du pas « a » de 10%

Comme on l'a conclut pour la projection de l'inclusion différentielle sur un plan, le domaine atteignable dans l'espace d'état R^3 occupe une certaine zone de l'espace qui englobe toutes les trajectoires possibles dans l'espace d'état.

Le domaine atteignable constitue la solution de l'inclusion différentielle. Nous allons, dans la partie suivante, élargir le domaine de recherche et étudier l'inclusion différentielle impulsionnelle.

5- Implémentation de l'algorithme de l'inclusion différentielle impulsionnelle

Les résultats de l'implémentation de cet algorithme nécessitent certaines fonctions spécifiques de Mathematica. On va détailler ces fonctions, des exemples de leurs utilisations et comment les utiliser dans notre programme.

5.1- La méthode « Event Locator » pour « NDSolve »

La méthode « Event Locator « pour « NDSolve » permet de détecter et de localiser un changement dans le système différentiel

On définit une fonction événement (qui décrit l'évènement) ; dans notre cas cela sera la transition. L'évènement n'est détecté que si la valeur de la solution est nulle.

Il est aussi possible de considérer des fonctions d'événement booléennes. Dans le cas où l'événement est détecté quand la fonction change de Vrai à Faux ou vice versa.

La méthode de « Event Locator » doit être à l'intérieur de « NDSolve ». Elle peut être utilisée comme un contrôleur. Il détecte l'évènement et prend l'action appropriée.

Dans Mathematica l'utilisation de la méthode de « Event Locator » nécessite l'appel des « packages » :

```
Needs["DifferentialEquations`NDSolveProblems`"];
Needs["DifferentialEquations`NDSolveUtilities`"];
Needs["DifferentialEquations`InterpolatingFunctionAnatomy`"];
Needs["GUIKit`"];
```

La méthode « Event Locator » contient des spécifications pour localiser l'évènement et traite l'action. Quand l'évènement est détecté, l'action par défaut est d'arrêter l'intégration. L'action sera exprimée par une valeur numérique qui substitue les variables dans les équations différentielles.

Dans notre programme on commence par définir l'inclusion différentielle. En effet, on définit les équations différentielles à résoudre en utilisant la fonction « NDSolve » qui est une fonction qui donne une solution numérique. On définit les différents variables d'état sauf le paramètre variant (dans notre exemple c'est le pas de variation de la vis « a » qui sera défini ultérieurement dans un intervalle de variation).

```
sol[p_?NumericQ, r_, t0_, x0_, y0_, dx0_] :=
 With[{k = 1, j = 0, m = 1, b = 0, l = 1},
```

Cette partie du programme permet de définir les différentes variables du système. En effet, « a » est un paramètre (qui ne prend pas de valeur fixe) alors que t_0, x_0, y_0, et dx_0 sont les conditions initiales et seront définies dans le corps du « NDSolve ». On attribue aux autres paramètres des valeurs numériques en utilisant la commande « With ».

```
NDSolve[
 {x''[t] == (k / (j + m (a * a))) * y[t] - (b / (j + m (a * a))) x'[t],
  y'[t] == -(r / l) x'[t] - (k / l) y[t] + 10,
  x[t0] == x0,
  y[t0] == y0,
  x'[t0] == dx0},
 {x, y},
 {t, t0, 10},
 Method →
  {"EventLocator",
   "Event" → (x[t] - bl),
   "Direction" → 1,
   "EventAction" :↦ Throw[tf = t, "StopIntegration"]
  },
 MaxSteps → Infinity
]
]
```

Dans cette partie du programme on résout les équations différentielles en utilisant « NDSolve ». On définit les équations différentielles, les conditions initiales ($x[t_0]$, $y[t_0]$ et $x'[t_0]$). Le paramètre variable « a » est défini plus haut et son intervalle de variation sera déterminé ultérieurement. On utilise la méthode « Event Locator » au sein de la fonction « NDSolve » pour détecter le changement d'état de système, dans notre cas cela sera la transition. Dans l'exemple ci-dessus, l'évènement est défini par la transition « arrêt droit ». Quand le système arrive à l'arrêt droit « b_l », l'évènement $x[t]-bl$ devient nul. On définit l'évènement dans un seul sens en utilisant « Direction → 1 ». L'action associée à l'évènement est de donner le temps de commutation « tf » qui sera défini comme un temps de départ du prochain état car comme on l'a défini dans le chapitre des inclusions différentielles impulsionnelles le temps de franchissement de la transition $[\tau,\tau']$ est supposé nul.

Il faut noter que l'action « Event Action » est notée avec du retard (:↦) pour qu'elle ne soit exécutée que si l'évènement est localisé.

Quand l'action n'arrête pas l'intégration, différentes instances de l'évènement sont détectées. En effet, l'évènement est détecté quand le signe de l'expression de l'évènement change et c'est pour cela qu'on pourra utiliser l'option « Direction » pour limiter l'exécution de l'évènement que dans un seul sens.

La méthode « Event Locator » peut avoir une liste d'évènements. Chaque composant de la liste constitue un évènement à part entière. On pourra ainsi définir différentes actions, directions … pour chacun de ces évènements dans une liste de la même longueur que celle des évènements [Site web Mathematica]

5.2- Les fonctions « Reap », « Sow »

La fonction « Sow [val] » étend la valeur de « val » pour la plus proche « Reap ».

La fonction « Reap [expr] » évalue « expr » et retourne les valeurs de « Sow ».

Exemple :

Ici on a uniquement le résultat final :

```
In[1]:= a = 3; a += a^2 + 1; a = Sqrt[a + a^2]
Out[1]= √182
```

En utilisant « Reap » , « Sow » on a, en plus du résultat final, deux résultats intermédiaires :

```
In[2]:= Reap[Sow[a = 3]; a += Sow[a^2 + 1]; a = Sqrt[a + a^2]]
Out[2]= {√182, {{3, 10}}}
```

Dans notre application on utilise la fonction « Row » :

```
ss2 = Table[With[{p0 = val}, Block[{tf = 0, xf = 0, yf = 0, dxf = 0, u = 10},
      Reap[
        While[tf < 10, Sow[sol[p0, tf, xf, yf, dxf]]]][[2, 1]]
    ]], {val, 1, v, z}];
```

Dans notre programme, on crée un tableau « ss2 » qui va contenir toutes les différentes valeurs que peut prendre la variable v ; on utilise un pas de variation variable « z » . Ce pas est précédemment défini dans l'algorithme des inclusions différentielles.

5.3- La fonction « Piecewise »

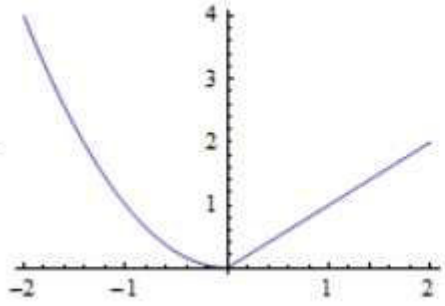
```
Piecewise[{{val1, cond1}, {val2, cond2}, …}]
```

La fonction « Piecewise » représente une fonction par morceaux de valeur ou d'expression « val_i » dans les régions définies par « $cond_i$ »

Exemple d'utilisation :

```
In[1]:= Plot[Piecewise[{{x^2, x < 0}, {x, x > 0}}], {x, -2, 2}]
```

Out[1]=



Dans notre programme on a utilisé la fonction « Piecewise » comme suit :

```
f = Table[Piecewise[{x[t] /. {First[#]}, #[[1, 1, 2, 1, 1, 1]] ≤ t < #[[1, 1, 2, 1, 1, 2]]} & /@ ss2[[x]], {x, 1, Length[ss2], 1}];
```

En effet, on classe les résultats des inclusions différentielles, précédemment calculées, suivant les intervalles du temps. Un exemple de résultat est le suivant :

```
{{InterpolatingFunction[{{0.`, 1.8425178525999748`}}, "<>"][t]}                0.` ≤ t < 1.8425178525999748`
 {InterpolatingFunction[{{3.685035705199995`, 5.527553557799925`}}, "<>"][t]}  3.685035705199995` ≤ t < 5.527553557799925`
 {InterpolatingFunction[{{7.370071410399899`, 9.212589262999876`}}, "<>"][t]}  7.370071410399899` ≤ t < 9.212589262999876`
 0                                                                             True

{{InterpolatingFunction[{{0.`, 1.8898648779721319`}}, "<>"][t]}                0.` ≤ t < 1.8898648779721319`
 {InterpolatingFunction[{{3.7797297559442633`, 5.669594633916395`}}, "<>"][t]} 3.7797297559442633` ≤ t < 5.669594633916395`
 {InterpolatingFunction[{{7.559459511884228`, 9.44932438985206`}}, "<>"][t]}   7.559459511884228` ≤ t < 9.44932438985206`
 0                                                                             True

{{InterpolatingFunction[{{0.`, 1.9374975529891456`}}, "<>"][t]}                0.` ≤ t < 1.9374975529891456`
 {InterpolatingFunction[{{3.874995105978291`, 5.812492658967437`}}, "<>"][t]}  3.874995105978291` ≤ t < 5.812492658967437`
 {InterpolatingFunction[{{7.749990211956582`, 9.687487764945729`}}, "<>"][t]}  7.749990211956582` ≤ t < 9.687487764945729`}
 0                                                                             True
```

6- Résultat de l'algorithme de l'inclusion différentielle impulsionnelle pour l'exemple de référence.

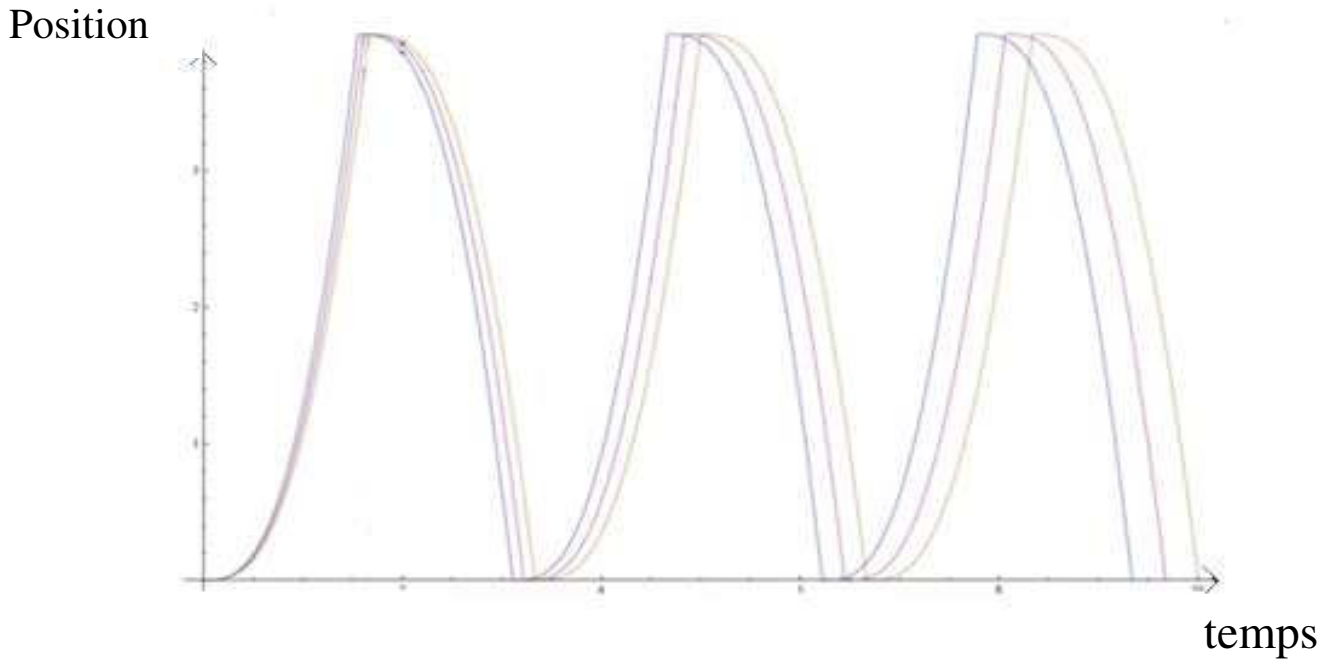


Figure 38 Résultat de l'inclusion différentielle impulsionnelle avec le pas de vis « a » variable

Comme on l'a défini au paragraphe 5.2 du chapitre 3 « Inclusion différentielle impulsionnelle » le temps du cycle du système est le temps nécessaire pour l'exécution d'un cycle d'aller retour.. Dans cette figure, on remarque que le mouvement de notre système mécatronique est cyclique. Ceci est dû au forçage des conditions initiales à chaque état. En effet, on réinitialise les conditions initiales à chaque cycle de l'automate hybride

Dans cette figure 38, on pourra distinguer le mouvement continu qui correspond à l'évolution continue des états et les commutations qui correspondent au franchissement des transitions discrètes. On remarque que la date de transition n'est pas unique ; l'ensemble des commutations prend un intervalle du temps pour toutes les valeurs du paramètre voir (figure 21 chapitre 3 paragraphe 5.2).

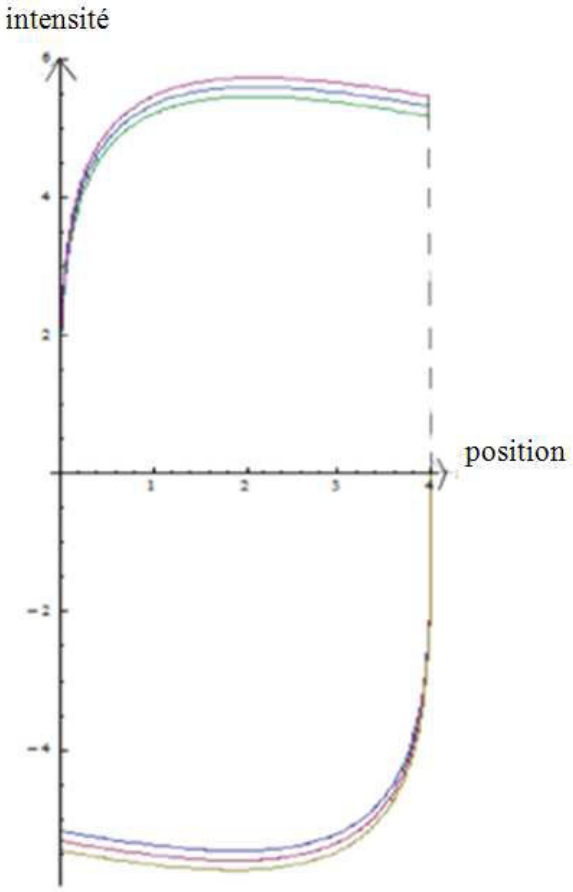


Figure 39 Projection du domaine atteignable de l'inclusion différentielle impulsionnelle sur le plan (x3, x1)

Dans les figures ci-dessus l'évolution de l'état est modélisée par des lignes continues alors que les transitions sont modélisées par des lignes interrompues. Le système démarre de sa position initiale br=0, évolue dans le sens positif du courant (x1) jusqu'à atteindre l'arrêt droit bl. A cet instant, le programme exécute l'action de l'événement et la transition est franchie. La masse change de sens et redémarre de $x_3=b_l$ vers $x_3= b_r$.

Le franchissement de transition n'est pas à temps nul. En effet, cela est dû à la nature de l'évènement. Dans notre exemple l'évènement est interne au système. Il dépend de ses variables d'état donc la transition ne peut être franchie que si toutes les instances du système pour l'inclusion différentielle sont vérifiées et du coup le système devrait attendre de valider toute l'inclusion différentielle pour franchir la transition. Dans le cas d'évènement externe au

système, la transition est franchie dès la détection de l'évènement et l'intervalle $[\tau,\tau']$ sera de durée nulle.

L'intervalle $[\tau,\tau']$ ne paraît pas dans la figure 39 car on est dans l'espace d'état et on ne voit pas la variation dans le temps qui peut apparaitre dans la figure 38 mais ici la variation de la position.

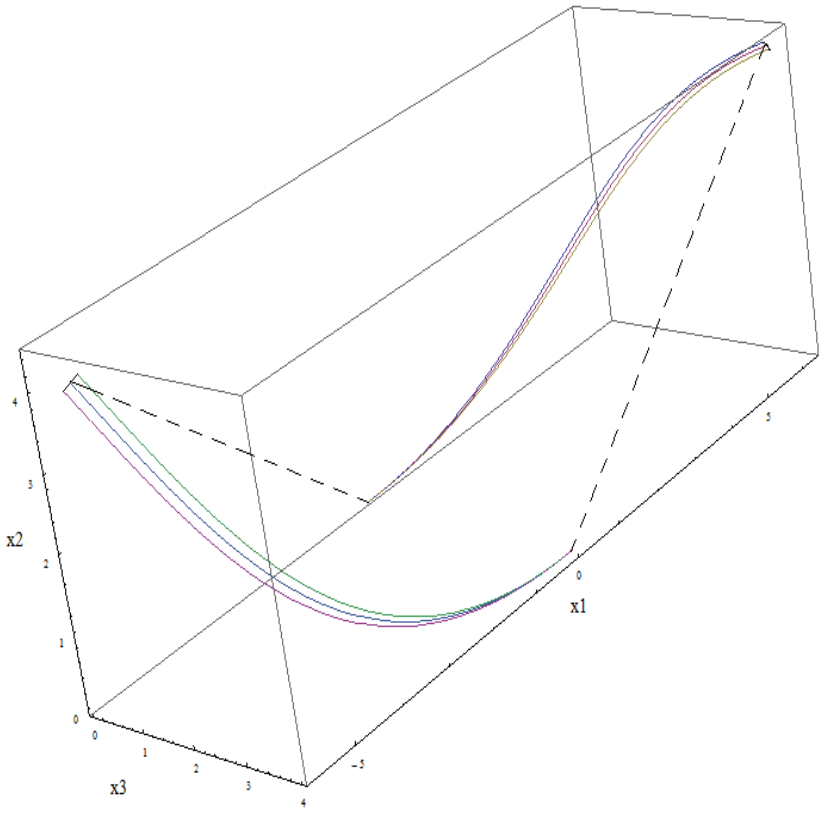


Figure 40 Domaine atteignable de l'inclusion différentielle impulsionnelle dans l'espace d'état R^3

On peut voir que l'inclusion différentielle impulsionnelle est un ensemble d'inclusions différentielles reliées par des transitions. Bien que le temps de franchissement de transition de notre application ne soit pas nul dans un premier temps, cela ne change rien à la forme générale du résultat de l'inclusion différentielle.

Cet algorithme a été utilisé dans le cas d'un système mécatronique avec un seul paramètre variable, nous avons étendu ensuite et généralisé l'algorithme pour étudier l'inclusion différentielle impulsionelle avec plusieurs paramètres variables.

7- Implémentation de l'algorithme de l'inclusion différentielle impulsionnelle avec plusieurs paramètres variables

Dans cette section on va généraliser le programme utilisé précédemment et ceci en supposant que le système présente n paramètres variables avec (n>1).

La première partie de ce programme est exactement la même que la précédente. On définit les équations différentielles, on définit les conditions initiales, on utilise les fonctions « NDSolve » pour résoudre les équations différentielles et la méthode « Event Locator » pour détecter les événements et franchir les transitions. La différence est que l'on enregistre les différentes combinaisons dans un tableau de dimension « n ». On a crée une matrice (n x n') avec « n » le nombre de paramètres variables et « n' » le nombre de valeurs que prennent les paramètres.

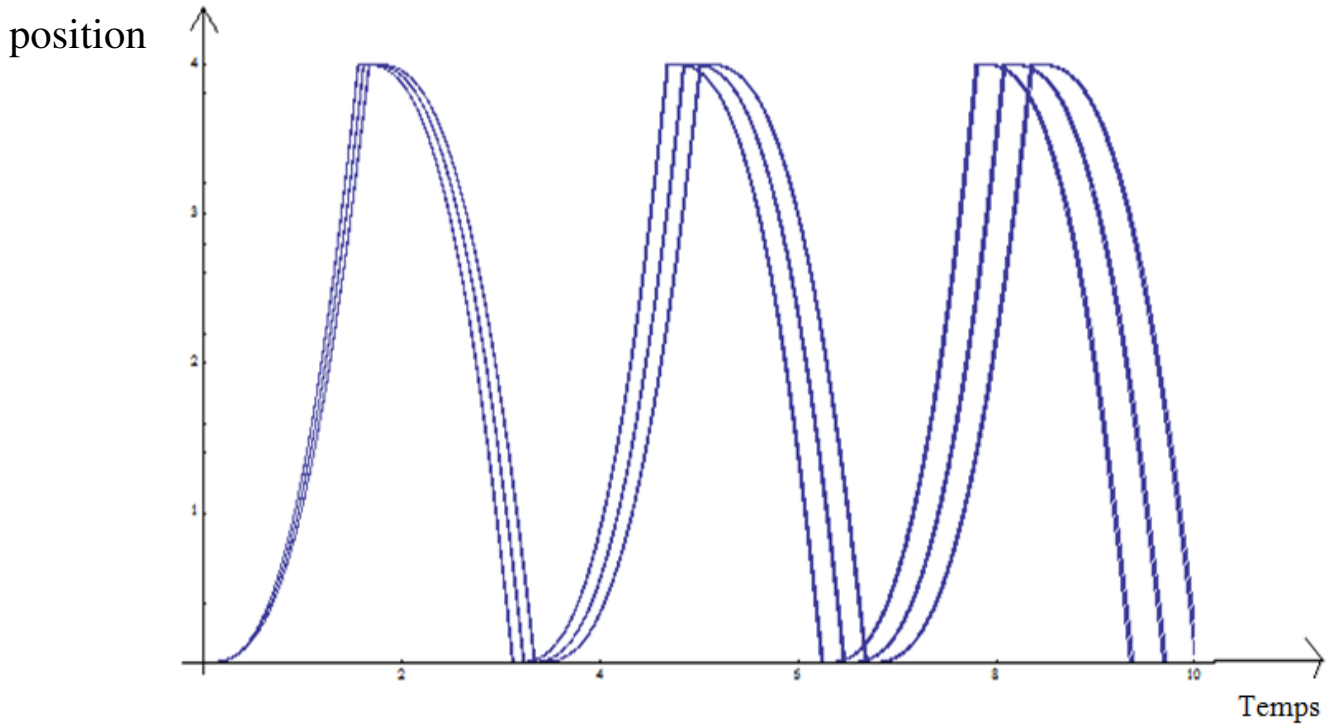


Figure 41 Projection inclusion différentielle impulsionnelle avec 2 paramètres variables sur le plan (x3, temps)

Dans notre exemple initial, le résultat est semblable à celui de l'inclusion différentielle impulsionnelle avec un seul paramètre variable. Notons que la figure 41 présente un résultat plus dense en courbes ce qui est normal puisque le nombre de combinaisons des cas possibles est plus grand.

Il faut noter que pour notre exemple le nombre de paramètres variables n'a pas trop influencé le fonctionnement global du système alors que la grandeur de variation agit directement sur le système et pourrait modifier la forme générale du domaine atteignable.

On remarque aussi que le temps de franchissement de transitions n'est pas nul et ceci est dû au fait que l'événement est interne au système et illustre l'analyse présentée au chapitre « les inclusions différentielles impulsionnelles ».

L'intervalle de temps de franchissement des transitions est de plus en plus grand au fil du temps ; ceci est dû au cumul des retards.

Conclusion

Dans ce chapitre, on a introduit quelques outils de programmation comme Modelica et Mathematica. Le premier est un langage de programmation orienté objet basé sur des classes. C'est un langage multi physique qui contient des librairies des différents domaines : mécanique, électronique, thermodynamique… Cet outil est très performant pour la programmation et la simulation. Mais dans notre algorithme on manipule plusieurs équations différentielles, inclusions différentielles et des systèmes hybrides c'est pour cela qu'on a besoin d'un système qui combine aussi bien les compétences de modélisation de simulation et les performances mathématiques. Pour visualiser l'effet de variation paramétrique sur le comportement d'un système mécatronique il nous faut un outil qui fait du calcul paramétrique, qui permet la modélisation des systèmes multiphysique et qui a une bonne interface graphique.

On a choisi Mathematica comme outil de programmation de nos algorithmes. Il existe aussi des langages de transfert comme Mathmodelica qui est une extension de Modelica dans le langage de Mathematica et le SystemModeler qui fait partie de Mathematica et qui inclut toutes les librairies de Modelica.

L'exemple choisi pour simuler les différents algorithmes est un système de pilotage dont on modifie à chaque fois son comportement pour être continu ou hybride.

On a commencé par présenter l'implémentation de l'algorithme des inclusions différentielles, puis on a appliqué des petites et grandes variations à la résistance et au pas de la vis. Selon la grandeur de variation, la forme générale du domaine atteignable peut varier.

Pour compléter nos travaux, on est passé des systèmes continus avec des paramètres variables à des systèmes hybrides. Pour modéliser ce problème, on a choisi l'inclusion différentielle impulsionnelle.

La programmation de l'algorithme d'inclusion différentielle impulsionnelle a nécessité l'appel des fonctions spécifiques de Mathematica qu'on a détaillées au fur et à mesure de la programmation.

On a obtenu les résultats des programmes d'inclusions différentielles impulsionnelles pour 1 et n paramètre(s) variable(s).

Sur l'exemple choisi, on observe des résultats semblables pour les deux programmes. On note que la date de franchissement des transitions n'est pas unique. Ceci est dû au fait que l'évènement déclencheur de transition est interne au système.

En perspective, on compte prendre en compte des exemples tests réels où la variation de n paramètres influence le fonctionnement.

Après avoir développé ces approches d'étude de variation paramétrique des systèmes mécatroniques continus et hybride, on a souhaité comparer avec d'autres approches et résolutions de ce problème à savoir les méthodes implicites et les approches géométriques.

Puis on abordera la combinaison de nos outils pour notre approche du tolérancement .

Chapitre 5 : Comparaison avec d'autres outils et approches de résolution

Introduction

Dans ce chapitre on va ouvrir le champ sur d'autres outils de résolution d'inclusion différentielle et des systèmes hybrides et les comparer avec notre approche. On va commencer par définir les automates hybrides à invariant polyèdre et la possibilité de les utiliser pour les systèmes hybrides avec du tolérancement. Ensuite, on définit les inclusions différentielles polygonales. Puis, on présente un autre outil de résolution d'inclusion différentielle appelé l'algorithme pratique. Celui-ci présente une approche implicite de résolution qui se base sur la méthode de Heun et sur la discrétisation de l'espace d'état et du temps.

1- Automate hybride à invariant polyèdre

Un automate hybride à invariant polyèdre (PIHA) (polyhedral invariant hybrid automata) est un ensemble H(X, X_0, F, E, I, G) avec

- $X = X_C \times X_{D,}$ *avec* $X_C \subseteq R^n$ est l'espace d'état continu et X_D est un ensemble fini de localisations discrètes.

- F est une fonction qui à chaque localisation discrète $u \in X_D$ affecte un champ de vecteur $f_u(.)$ dans X_C.
- $I : X_D \to 2^{X_C}$ affecte à $u \in X_D$ un ensemble invariant de la forme $I(u) \subseteq X_C$ avec $I(u)$ un polyèdre convexe.
- $E \subseteq X_D \times X_C$ est un ensemble de transitions discrètes.
- $G : E \to 2^{X_C}$ attribue à $e = (u,u') \in E$ un ensemble de gardes (il est l'union des faces de I(u)).
- $X_0 \subseteq X$ est l'ensemble des états initiaux de la forme $X_0 = \bigcup_i (P_i, u_i)$ avec $P_i \subseteq I(u_i)$ est un polytope et $u_i \in U$

Les différences entre un PIHA et un automate hybride général :

- Il n'y a pas d'ensemble de « reset » associé aux transitions discrètes (c.à.d. il n'y a pas de discontinuités dans les trajectoires des états continus)
- Les invariants sont définis par des inégalités (invariants polyédrales) .
- Les gardes sont des faces des invariants (c.à.d une transition discrète est franchie quand la trajectoire de l'état continu atteint un ensemble de garde).

L'espace des transitions discrètes d'un système PIHA est associé aux bornes de l'invariant des états continus. Pour vérifier les propriétés d'un système de transitions infini il faut trouver une bi-simulation c.à.d trouver un système de transition fini équivalent au système de transition original en satisfaisant le problème de vérification. Cette bi-simulation, ou le quotient du système des transitions, est construite à partir d'une partition de l'espace d'état du système des transitions original. On contrôle alors le quotient du système des transitions pour vérifier l'existence d'une bi-simulation. Si la bi-simulation n'est pas vérifiée alors on raffine la partition de l'espace et un nouveau quotient du système des transitions est construit. On répète cette itération jusqu'à ce qu'on trouve une bi-simulation.

Pour construire le quotient du système de transition (quotient transition system QTS) pour une partition du système des transitions, il est nécessaire de calculer les transitions entre les éléments des partitions. Ceci nécessite le calcul des états continus atteints après un départ d'un élément de la partition (l'ensemble des états continus sur les bornes de

l'invariant) et de trouver les points d'intersection entre le domaine atteignable et d'autres éléments de la partition (autre sous ensemble des bornes de l'invariant).

Ce domaine atteignable est nommé tube « flow pipe » pour l'évolution continue qui correspond à l'invariant.

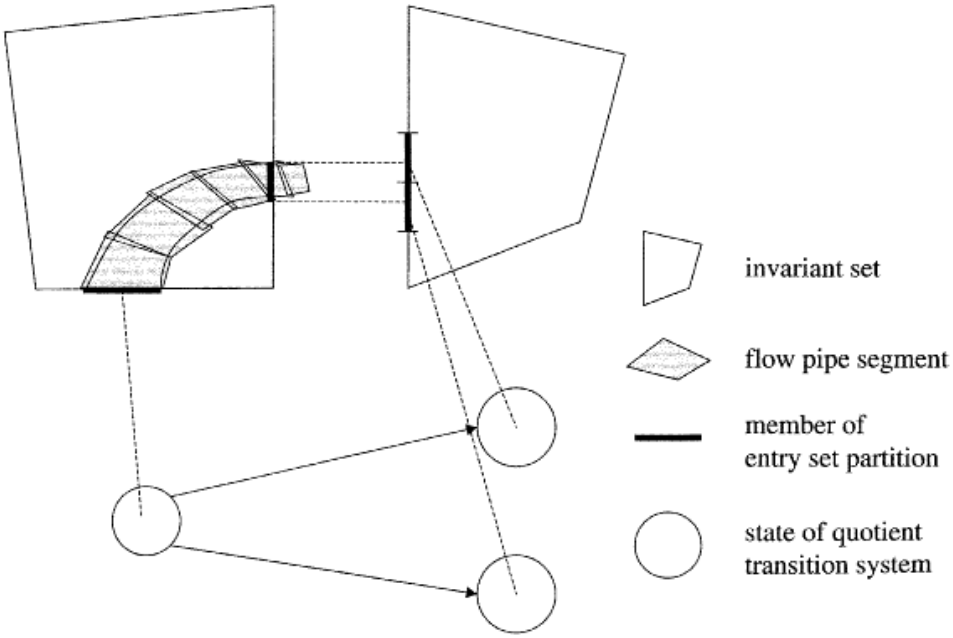


Figure 42 Illustration de construction des transitions de AQTS [Chutinan, 2003]

Dans le cadre de nos études de recherche on veut étudier le tolérancement des systèmes mécatronique et ceci en définissant un intervalle d'incertitude pour les paramètres. On a commencé par utiliser la méthode des inclusions différentielles pour aborder le problème des systèmes dont le comportement est purement continu. Le système est alors défini par une inclusion différentielle de la forme :

$$\frac{dx}{dt} \in F(t, x(t)), \qquad x(0) \in X_0 \qquad (5.1)$$

avec F est une application de $R \times R^n$ dans R^n, X_0 est le l'ensemble de l'état initial et $C \in R$

$$F(t, x(t)) = \left\{ z = f(t, x, v) \,\middle|\, v \in C(t, x) \right\} \qquad (5.2)$$

Le tolérancement d'un paramètre est défini par un intervalle dont on ne connait que les extrémités. Le résultat étant le domaine atteignable.

Pour les systèmes hybrides avec des paramètres variables on a opté pour la modélisation d'inclusion différentielle impulsionnelle. Cette modélisation se base sur un automate hybride dont l'évolution continue dans les états est décrite par une inclusion différentielle. Dans le cadre des automates hybrides à invariant polyèdre, l'invariant est défini par une inéquation. L'état est viable si l'invariant est vrai.

Pour les automates hybrides l'invariant peut être soit déclaré (en extension) comme par exemple le définir par un polyèdre, soit en compréhension c'est-à-dire défini par une équation ou une inéquation de type : $\|x(t)\| \leq a$.

Pour l'inclusion différentielle, le domaine de variation peut être défini, par exemple, par un polygone ou un cercle, ou en compréhension c'est-à-dire défini par une équation de type : $2 \leq \dot{x} \leq 3$.

Pour les automates hybrides à invariant polyèdre, l'espace d'état continu est divisé en un ensemble de parties disjointes. L'union de ces parties couvre tout l'espace d'état. L'invariant est écrit sous la forme d'inéquations linéaires, les gardes qui sont les faces du polyèdre peuvent être considérées comme des hyper plans qui partitionnent l'espace d'état continu en états discrets. Ces états discrets, sont appelés des *locations*. Les locations peuvent changer sous l'évolution des états continus. Quand la trajectoire d'un état continu coupe un hyper plan de garde, un intervalle d'évènements est franchi et la location en cours est changée. La location de transitions est possible seulement entre deux modes de transitions adjacents.

L'automate hybride à invariant polyèdre constitue un formalisme de modélisation des systèmes hybrides. Les variations paramétriques peuvent être exprimées en utilisant un polyèdre. Cet outil de modélisation peut modéliser aussi bien un système avec du tolérancement sur les paramètres qu'un système avec du tolérancement sur les performances. Le calcul de cet outil de modélisation peut être compliqué c'est pourquoi on trouve une méthode géométrique pour la détection des évènements proposée par [Baniardalani, 2012].

Dans [Baniardalani, 2012] on propose une méthode de détection des événements avec une méthode géométrique en calculant l'angle entre le champ de vecteur et la normale de surface de commutation au point de garde.

Lorsqu'on impose un tolérancement sur les paramètres du système, généralement ces paramètres sont internes au système, et cette variation est plutôt subie. Donc l'objectif est de déterminer l'influence de cette variation sur le fonctionnement global du système et de décider l'action à prendre par conséquence.

Si on prend le cas de notre exemple d'application mécatronique (présenté dans le chapitre inclusion différentielle), une variation de la résistance et du pas de vis peuvent faire varier le temps de réponse du système ainsi que la position atteinte. Dans ce cas si notre intérêt se porte sur la position par exemple (précision de position de 5%) on peut poser un invariant qui exprime cette condition.

Un automate hybride à invariant polyèdre peut satisfaire cette contrainte. En effet, le tolérancement sur les performances pose des contraintes sur les critères de performance (gabarit de réponse) et cela revient à poser des conditions sur les variables d'état.

Un outil idéal sera celui qui tient compte à la fois du tolérancement des paramètres en respectant le tolérancement des performances.

Les automates hybrides à invariant polyèdre peuvent être utilisés dans le cadre de développement et de perspectives de nos travaux sur le tolérancement. En effet, cet outil permet de tenir compte aussi bien des variations sur les paramétriques que des variations sur les performances du système. Ce qui est une extension de l'inclusion différentielle impulsionnelle qui se limite à l'étude des systèmes hybrides à paramètres variables.

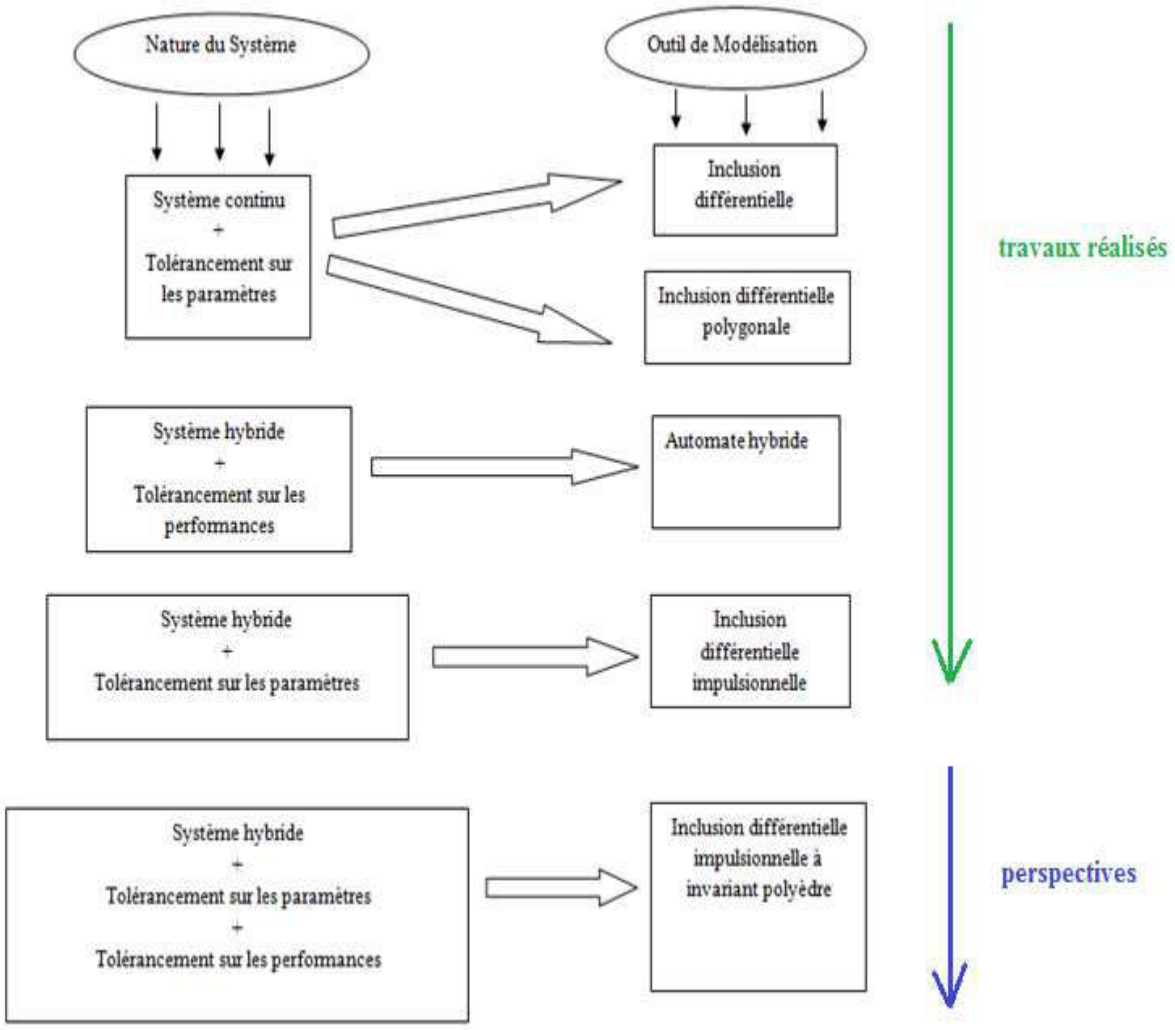


Figure 43 Les différents systèmes avec tolérancement et les outils de prise en charge

2- Inclusion différentielle polygonale

Un système d'inclusion différentielle polygonale (SPDI) consiste en une partition d'un sous-ensemble du plan en des régions polygones et convexes avec une inclusion différentielle constante associée à chaque région [Schneider, 2002], [Asarin, 2007], [Asarin, 2003].

Un système d'inclusion différentielle est défini par :

$$x' \in D_i \ avec \ x \in P_i \subseteq R^2 \quad (5.3)$$

avec D_i et P_i sont des polygones et $\{P_i\}_{i \in I}$ est une partition finie de R^2.

Un segment de trajectoire d'un tel système est une fonction :

$$\xi : [0,T] \rightarrow R^2$$

Tel que pour $t \in [0,T], si\ \xi(t) \in P_i\ et\ \xi'(t)$ est définie alors $\xi'(t) \in D_i$. Si T est infini, on appelle alors ξ une trajectoire. Un segment de trajectoire détermine une signature σ définie par la suite ordonnée des points d'intersection des arêtes de polygone traversée par elle. La figure ci-dessous montre un exemple d'inclusion différentielle polygonale et d'un segment de trajectoire.

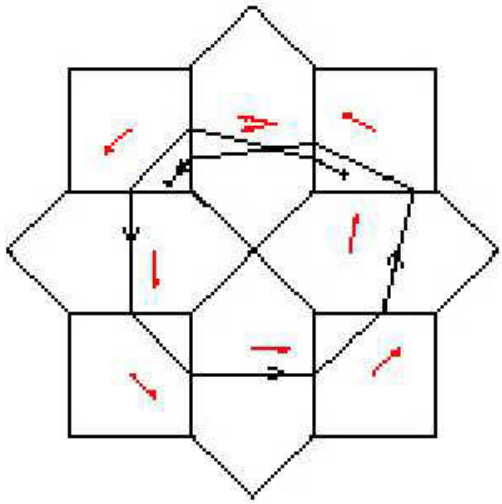


Figure 44 Un segment de trajectoire d'une inclusion différentielle polygonale [Schneider 2002]

Dans le domaine du contrôle, les inclusions sont utilisées pour modéliser des systèmes comprenant des incertitudes ou du bruit. On pourra modéliser ces systèmes en utilisant les équations différentielles de la forme :

$$\dot{x} = f(x,u)\ avec\ u \in U$$

u est la commande ou le bruit. Une autre représentation est l'inclusion différentielle :

$$\dot{x} \in g(x)\ avec\ g(x) = \{\ f(x,u) \mid\ u \in U\ \} \quad (5.4)$$

L'inclusion différentielle permet d'avoir toutes les valeurs possibles de f ; de plus, les inclusions différentielles polygonales permettent d'avoir une approximation conservative des dynamiques complexes non linéaires.

La représentation d'un système d'inclusion différentielle polygonale pour un automate hybride linéaire non déterministe avec des trajectoires continues est possible comme le montre la figure ci-dessous.

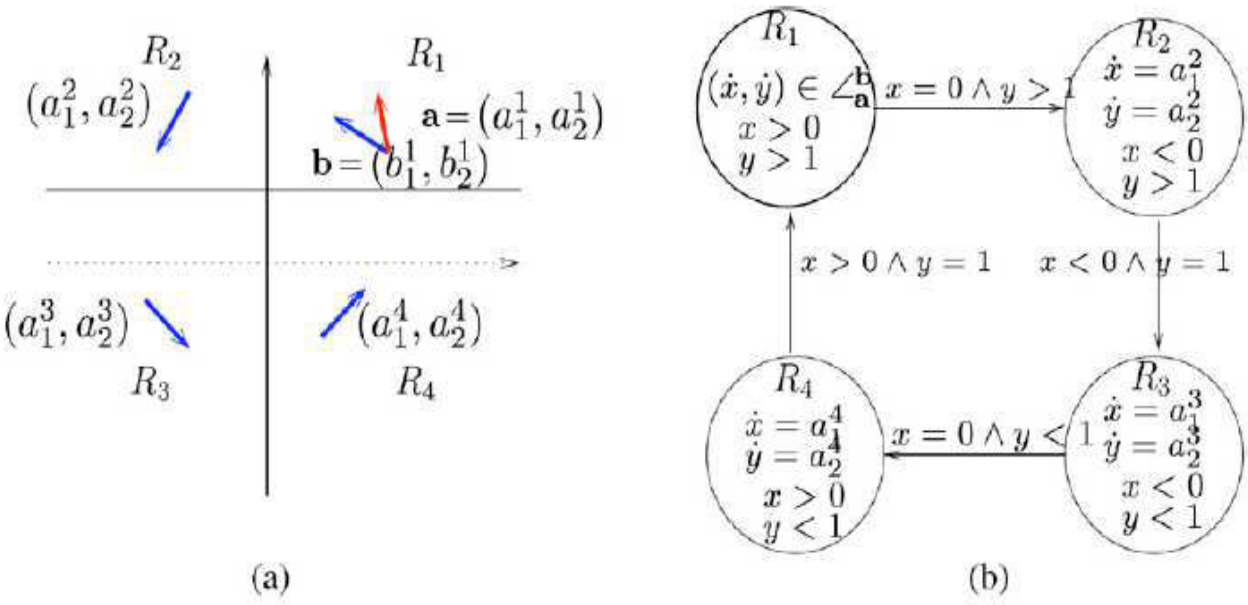


Figure 45 Représentation d'un système d'inclusion différentielle polygonale (a) par un automate hybride (b) [Schneider 2002]

Pour l'inclusion différentielle polygonale l'évolution se fait de passage d'une zone à une autre alors que dans notre cas (l'inclusion différentielle classique) le système évolue dans le temps ou par rapport à une variation paramétrique et non par rapport à sa position dans l'espace d'état.

- **Accessibilité :**

Le problème d'accessibilité entre les états p et q consiste en la question suivante : existe-t-il un segment de trajectoire qui a comme état initial p et état final q ? [Asarin, 2001]

Le résultat est basé sur la représentation des trajectoires par des fonctions multivaluées affines par morceaux. Etant donné un polygone P et deux de ses arêtes e et f, l'ensemble de tous les points accessibles de n'importe quel point $x \in e$, est défini par la fonction multivaluée

$$\Phi_{e,f}(x) = [ax+b, cx+d] \quad (5.5)$$

Les coefficients dépendent de e, f et du polygone D qui détermine la dynamique du système dans le polygone P.

Cette technique permet de définir un système dynamique unidimensionnel dont le comportement qualitatif est équivalent au comportement d'un système dynamique bidimensionnel. Etant donnée une signature $\sigma = e_0 \ldots e_k$, les points accessibles à partir d'un point x par un segment de trajectoire dont la signature est σ sont définis par :

$$\Phi_\sigma(x) = \Phi_{e0,e1} \circ \ldots \circ \Phi_{ek-1,ek} = [\alpha x + \beta, \gamma x + \delta] \quad (5.6)$$

Ceci permet de déterminer les limites d'un segment de trajectoire dont la signature est σ par un calcul de point fixe. Une analyse qualitative des segments de trajectoires est effectuée ce qui permet de classifier les signatures de trajectoires cycliques en un certain nombre de types différents. Cette topologie est utilisée pour déterminer si une signature donnée peut être répétée un nombre infini de fois par une trajectoire où au bout d'un certain temps elle sera abandonnée.

Dans [Schneider 2002] et [Gordon, 2006] on trouve un algorithme qui permet de répondre à la question d'accessibilité d'un point q du plan à partir d'un point p en utilisant l'inclusion différentielle polygonale implémentée sous un outil nommé SPEEDI [Schneider 2002a] , [Pace, 2004] .

La figure ci-dessous montre la sortie graphique générée par SPEEDI lorsque la réponse à la question d'accessibilité est positive. La zone colorée en vert représente les points atteignables.

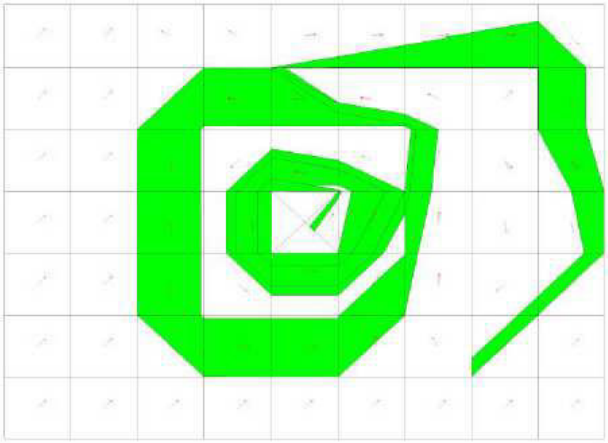


Figure 46 Implémentation de l'algorithme d'inclusion différentielle polygonale sous SPEEDI [Schneider 2002]

Le problème d'accessibilité est très présent aussi bien dans les systèmes continus que hybrides. La notion de portrait de phase été reprise dans ce contexte (inclusion différentielle par morceaux). Dans [Asarin, 2008] on a repris les notions de noyau de viabilité et de noyau de contrôlabilité pour un ensemble de trajectoires ayant un comportement cyclique.

On dit qu'une trajectoire ζ est viable dans un ensemble K si pour tout $t \geq 0$, $\xi(t) \in K$. On dit que K est un domaine viable si pour tout $x \in K$ il existe une trajectoire viable dans K qui part de x. On appelle noyau de viabilité d'un ensemble K le plus grand domaine viable dans K.

Dans [Schneider 2002] et [Schneider 2003] on trouve une étude du noyau de viabilité pour les trajectoires avec signature cyclique (c.à.d l'ensemble de point à partir desquels on peut « tourner » dans un nombre d'arrêts). Il montre que cet ensemble est un polygone non convexe.

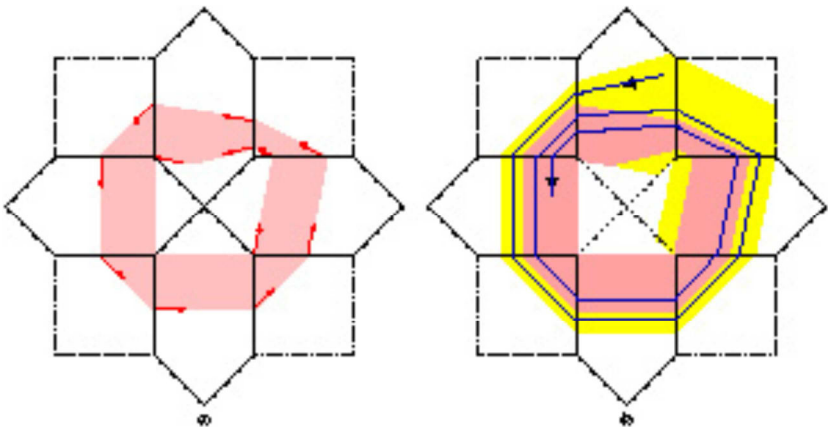


Figure 47 Exemple du noyau de viabilité (colorié en jaune) et le noyau de contrôlabilité (colorié en rose) [Schneider 2002]

On dit qu'un ensemble K est contrôlable s'il existe un segment de trajectoire qui relie un voisinage ouvert K à un autre de K sans jamais sortir de K. On appelle noyau de contrôlabilité le plus grand sous ensemble de K qui est contrôlable.

La notion de noyau de contrôlabilité est l'analogue de cycle limite. En effet, toute trajectoire ayant une signature cyclique converge vers le noyau de contrôlabilité du cycle.

Les noyaux de contrôlabilité se sont avérés de très grande importance dans le portrait de phase d'une inclusion polygonale par morceaux. En effet, toute trajectoire sans auto intersection converge vers l'un de ces noyaux.

Dans l'inclusion différentielle polygonale on trouve une application de la théorie de viabilité établie par Aubin et définie dans le chapitre d'inclusion différentielle impulsionnelle. On définit les noyaux de contrôlabilité et de viabilité en se basant sur les parcours des trajectoires et aux contraintes. Ces contraintes peuvent exprimées le tolérancement sur les paramètres ou sur les performances. On utilise ces notions pour définir différents types de systèmes.

L'inclusion différentielle polygonale est un outil géométrique de résolution des inclusions différentielles. Sa résolution se base sur la partition de l'espace d'état en polygone et l'intersection du résultat avec ces polygones. Cet outil peut s'avérer intéressant pour des applications dans le domaine de la robotique par exemple pour déterminer la trajectoire. Mais il ne peut pas satisfaire nos critères de résolution pour les systèmes mécatroniques pour deux raisons :

1- Cette méthode est limitée aux espaces d'état de dimension 2 alors que nous avons posé un espace d'état de dimension n comme critère essentiel pour le choix de l'outil.
2- L'inclusion différentielle polygonale évolue par passage d'une zone à une autre alors que pour l'inclusion différentielle classique le système évolue par rapport au temps ou par rapport à une variation paramétrique.

Une autre approche de résolution d'inclusion différentielle est présentée ci-dessous. Il ne s'agit pas de méthode géométrique mais le principe de résolution nécessite la partition de l'espace d'état et du temps.

3- Autre approche de construction de l'inclusion différentielle

3.1- La méthode de Heun

Définition générale :

Etant donné :

- $I = [t_0 , T]$ un intervalle réel fini avec $T > t_0$

- $X_0 \subset R^n$ un ensemble non vide
- $F : I \times R^n \to 2^{R^n}$ est une application de $I \times R^n$ dans tous les sous ensembles de R^n.

On cherche une fonction continue $x(.): I \to R^n$ qui satisfait la condition initiale : $x(t_0) = x_0 \in X_0$ et l'inclusion différentielle $\dot{x}(t) \in F(t, x(t))$, $\forall t \in I$

L'ensemble χ est l'ensemble de toutes les solutions ou trajectoires x(.) de l'inclusion différentielle dans l'intervalle [t_0 , T].

Le domaine atteignable $R(\tau)$ à un instant $\tau \in [t_0, T]$ de l'inclusion différentielle est défini par :

$$R(\tau) := \{x(\tau) : x(.) \in \chi\}$$

Des méthodes de discrétisation ont été utilisées pour trouver une approximation du domaine atteignable d'une inclusion différentielle. L'estimation de l'erreur de premier ordre a été obtenue dans [Artstein, 1994], [Dontchev, 1989], [Dontchev,1994], [Hackle, 1993]et [Kurzhanski, 1992]. Ces méthodes ont été employées pour essayer de résoudre le problème de solution des inclusions différentielles. Par contre, [Veliov, 1997] a établi une approximation d'ordre supérieur du domaine atteignable des systèmes de contrôle applicable à une classe de l'inclusion différentielle non linéaire de la forme suivante :

$$\dot{x}(t) \in \left\{ \underbrace{f_0(t, x(t)) + F_0(t, x(t), u)}_{= f_u(t, x(t), u)} : u \in U \right\}, x(t_0) \in X_0 , t \in [t_0, T] \qquad (5.7)$$

Avec :

- $x(.): [t_0, T] \to R^n$, $f_0 : [t_0, T] \times R^n \to R^n$ sont des fonctions vectorielles
- $F_0 : [t_0, T] \times R^n \to R^n \times R^r$ est une fonction matricielle de dimension $(n \times r)$
- $U \subset R^r$ est un ensemble convexe et compact.

Dans [Veliov, 1997], on a montré qu'il existe pour l'inclusion différentielle exprimé ci-dessus, une version de la méthode de Heun qui donne une approximation du domaine atteignable d'ordre $O(h^2)$ ou $O(h^{1.5})$ et qui satisfait les conditions aux limites pour toutes les solutions, la condition de continuité lipschitzienne pour toutes les dérivées premières de f_0 et F_0 et la condition de structure pour les colonnes de F_0. Le calcul du domaine atteignable utilisant cette méthode est très difficile de point de vue pratique et ceci est dû à la complexité,

à la difficulté de calcul et de la représentation des ensembles infinis résultants de chaque itération de la méthode.

La méthode présentée dans [Veliov, 1997] est basée sur la discrétisation du système dans l'espace-temps seulement. Alors que dans [Barrios, 2012], on propose une discrétisation de l'espace-temps et de l'espace d'état en utilisant des mailles pour la discrétisation de ce dernier.

3.2- L'algorithme pratique basé sur la méthode deHeun :

Cet algorithme est une révision pratique de la méthode de Heun proposé par [Veliov, 1997] pour une approximation du domaine atteignable d'une inclusion différentielle. Il établit une discrétisation complète de l'inclusion différentielle tout en contrôlant la mémoire utilisée lors du calcul des itérations. Il propose une méthode qui se base sur l'approximation de l'ensemble U par une discrétisation et une approximation successive de l'ensemble discrétisé résultant par des projections.

La méthode de Heun :

Pour $n \in N$, on choisit une grille $t_0 = t_0 < t_1 < \cdots < t_n = T$ avec un pas $h = \frac{T - t_0}{n} = t_K - t_{k-1}\ (k = 1, \ldots, n)$, on suppose que $R_N(0) = X_0$, pour $k = 1, \ldots, n-1$, on calcule :

$$R_N(k+1) = \bigcup_{y \in R_N(k)} \left\{ y + \frac{h}{2}(f_u(t, y, u) + f_u(t, y + hf_u(t, y, u), u)) : u \in U \right\} \qquad (5.8)$$

L'algorithme pratique : [Barrios, 2012]

Pour $n \in N$, on choisit une grille $t_0 = t_0 < t_1 < \cdots < t_n = T$ avec un pas $h = \frac{T - t_0}{n} = t_K - t_{k-1}$ $(k = 1, \ldots, n)$. Soit $M_1, M_2 \in N, M_1 \geq 2^r, M_2 \geq 2^n$

on suppose que $R_n(0) = X_0$,

- On calcule la projection de l'ensemble U sur $\Delta_{x,p}$.
- $C_{U,x,\rho}$: est une maille d'origine x et de largeur ρdans R^n.
- Pour $k = 1, \ldots, n-1$, on calcule $R_N(k+1)$

1. On calcule $\tilde{R}_n(k+1)$

$$\tilde{R}_N(k+1) = \bigcup_{y \in R_N(k)} \left\{ y + \frac{h}{2}(f_u(t,y,u) + f_u(t, y + hf_u(t,y,u),u)) : u \in C_{U,x,\rho} \right\} \qquad (5.9)$$

2. On calcule la projection de l'ensemble $\tilde{R}_n(k+1)$ sur $\Delta_{x,p}$

$$R_{n(k+1)} = C_{\tilde{R}_{n(k+1),c,\rho}}$$

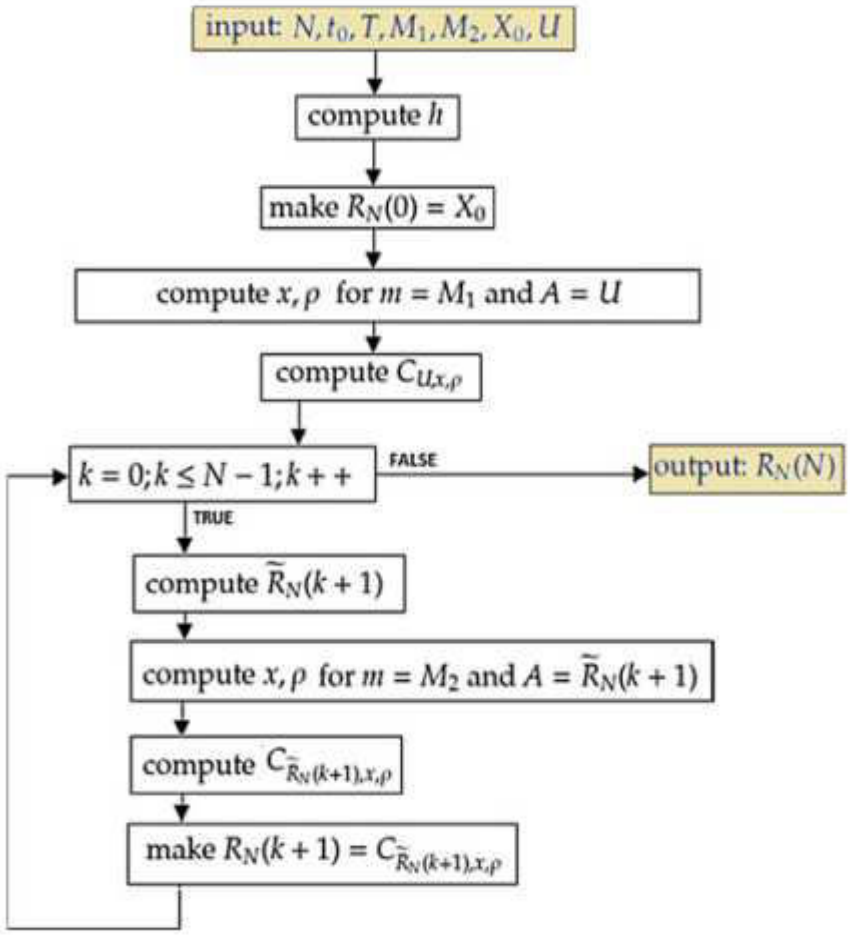


Figure 48 Le diagramme de l'algorithme pratique [Barrios, 2012].

Cet algorithme pratique permet une résolution de l'inclusion différentielle en utilisant la méthode de Heun. Cette méthode se base sur la discrétisation de l'espace d'état et du temps pour avoir en résultat une approximation du domaine atteignable par projection. Cette méthode est différente de celle utilisée lors de nos recherches. En effet, dans le cadre de cette thèse on s'est basé sur l'algorithme de Raczynski pour résoudre l'inclusion différentielle. Le tableau qui suit illustre les différentes étapes de chaque algorithme de résolution.

4- Comparaison de l'algorithme pratique et de notre algorithme de résolution de l'inclusion différentielle

	Notre algorithme de résolution de l'inclusion différentielle	**L'algorithme pratique de résolution de l'inclusion différentielle**
Entrée	- Equations différentielles - Intervalle de variation des paramètres - Les conditions initiales	- N : nombre total de pas - t_0 : temps de départ, T : le temps final - $M_1, M_2 \in N$ (pour le calcul de dimension de maille) - X_0 : conditions initiales - U : domaine de variation
1ére étape	Générer le vecteur adjoint initial	Calcul du pas de variation (fixe)
2éme étape	Algorithme de calcul de l'Hamiltonien qu'on maximise à chaque pas de variation	Approximation du premier domaine atteignable $R_n(0) = X$
3éme étape	Enregistrer le pas et toute la trajectoire	Calcul des mailles : discrétisation de l'espace d'état et du temps
4éme étape	Modifier le pas selon la densité des points sur le domaine atteignable	Calcul du $C_{\tilde{R}_{n(k+1),c,\rho}}$: la projection du domaine atteignable
5éme étape	S'il existe assez de trajectoires pour avoir le domaine atteignable, on enregistre sinon on reboucle à l'étape 1	Approximation du domaine atteignable par la projection $R_{n(k+1)} = C_{\tilde{R}_{n(k+1),c,\rho}}$

Tableau 6.1. Comparaison entre notre algorithme et l'algorithme pratique de résolution de l'inclusion différentielle

Dans ce tableau on a détaillé les différentes étapes de résolution de l'inclusion différentielle pour les deux algorithmes.

- Dans l'algorithme pratique on ne voit pas les paramètres d'entrée, alors qu'avec notre algorithme on fait entrer les équations ainsi que les paramètres physiques du système.

-Dès la première étape on fixe le nombre de pas de variation pour l'algorithme pratique alors dans notre cas le pas de variation est variable.

- Dans notre algorithme le calcul de pas de variation se fait à travers un algorithme de maximisation de l'Hamiltonien, alors que dans l'algorithme pratique la discrétisation de l'espace d'état et du temps se fait avec un calcul de projection et après avoir établir différentes définitions et théorèmes.

- l'algorithme pratique donne une approximation du domaine atteignable dans la dimension 2, alors que notre algorithme permet de déterminer le domaine atteignable dans R^n et d'avoir sa projection sur le plan ou sur l'espace R^3.

L'algorithme pratique nécessite des données assez précises à l'entrée comme le nombre total d'itérations N, les données M_1 et M_2 pour le maillage de l'espace d'état et du temps. La méthode de calcul de la projection du domaine atteignable nécessite l'utilisation des définitions et du théorème pré-établi par les chercheurs (se basant sur la méthode de Heun et utilisant la notion de distance de Haussdorf) . Alors que dans notre algorithme qui se base sur l'algorithme de Raczynski, on n'a besoin que des équations différentielles et des intervalles de variation des paramètres, le nombre d'itérations est à déterminer suivant la forme du domaine atteignable obtenu, le pas de variation est variable suivant les concentrations des points. L'implémentation de notre algorithme fait appel à des notions d'Hamiltonien , de commande optimale et des algorithmes d'optimisations pour avoir au final comme résultat un domaine atteignable dans l'espace R^n .

Nous avons présenté deux méthodes d'approche de résolution de l'inclusion différentielle. Chaque méthode a ses inconvénients et ses complications. Dans l'exemple de la fièvre de dengue [Barrios, 2012],on ne fait pas apparaître un système physique avec des entrées mais on fait face à un système implicite par rapport aux paramètres variables. Ce modèle nécessite le recours à une transformation des données pour voir les variations.

Notre algorithme est plus simple à implémenter car on n'a pas besoin de déterminer des données à l'entrée. On étudie un système mécatronique dont les variations paramétriques sont explicites (exprimées en entrée du système). Aussi le résultat est un domaine atteignable dans R^n alors que pour l'algorithme pratique on obtient en résultat une approximation du domaine atteignable par projection sur R^2. L'algorithme de Raczynski est alors, plus adapté à ce type de problème.

Conclusion

Ce chapitre présente une extension aux chapitres précédents 2 et 3 car on présente d'autres outils de résolution de l'inclusion différentielle et d'automate hybride. On commence par définir les automates hybrides à invariant polyèdre. Pour les automates hybrides à invariant polyèdre, l'espace d'état continu est divisé en un ensemble de parties disjointes. L'union de ces parties couvre tout l'espace d'état. L'invariant est écrit sous la forme d'inéquations linéaires, les gardes qui sont les faces du polyèdre peuvent être considérées comme des hyper plans qui partitionnent l'espace d'état continu en états discrets. Cet outil représente une généralisation des automates hybrides. Il peut être considéré comme un outil de résolution des systèmes hybrides avec du tolérancement aussi bien sur les paramètres que sur les performances. Cet outil pourra bien être utilisé dans le cadre de développement de ces travaux pour l'étude du tolérancement.

Ensuite on présente les inclusions différentielles polygonales. L'inclusion différentielle polygonale est un outil géométrique de résolution des inclusions différentielles. Sa résolution se base sur la partition de l'espace d'état en polygone, et sur l'intersection du résultat avec ces polygones. Cet outil peut s'avérer intéressant pour des applications dans le domaine de la robotique par exemple pour déterminer la trajectoire. Mais, ce outil se limite aux cas d'espace d'état de dimension 2.

On a présenté ensuite un algorithme de résolution de l'inclusion différentielle appelé l'algorithme pratique. Il s'agit d'une méthode implicite dont l'algorithme se base sur la méthode de Heun pour la résolution d'inclusion différentielle en discrétisant l'espace d'état et le temps ; il fait appel à différentes définitions et théorèmes établis dans [Barrios, 2012]. On a établi une comparaison entre cet algorithme et notre algorithme utilisé pour la résolution de l'inclusion différentielle (développé au chapitre 2) et on a montré que notre algorithme est plus simple de point de vue donnée d'entrée et son résultat est plus complet car il nous fournit le domaine atteignable dans l'espace R^n , alors que l'algorithme pratique donne comme résultat une projection du domaine atteignable dans l'espace R^2.

La comparaison des méthodes adoptées pour l'étude des variations paramétriques avec d'autres approches existantes montre que notre choix (méthode d'inclusion différentielle et méthode d'inclusion différentielle impulsionnelle) est le plus approprié pour l'étude des systèmes mécatroniques par rapport à nos critères de choix énoncés initialement dans la

problématique (chapitre introduction paragaphe 3.2). On va voir ensuite comment on peut utiliser ces méthodes pour développer un outil d'étude de tolérancement mécatronique.

Chapitre 6: Développement du tolerancement et perspectives sur les propriétés des systèmes à paramètres variables

Introduction

Dans ce chapitre on reprend les différents outils et résultats, présentés dans les chapitres précédents, pour le calcul de domaine atteignable. On exprime une propriété du système pour définir le tolérancement dans l'espace d'état. On définit les zones du tolérancement et on établit un outil métrique pour le calcul des distances entre ces zones. On définit la distance de Hausdorff et on l'utilise pour montrer les différents cas de figures qu'on peut avoir par l'intersection entre la zone de fonctionnement et le domaine atteignable. On finit par des applications dans le plan en utilisant les notions de distances de Hausdorff, du domaine atteignable et du noyau de viabilité. Nous illustrons ces concepts sur l'exemple d'un système de pilotage défini au chapitre 2.

1- Définition des zones du tolérancement

Définition du tolérancement (Rappel)

Le tolérancement du comportement d'un système mécatronique est l'activité du concepteur consistant à définir un sous ensemble S de l'espace d'état R^n, dans lequel le vecteur d'état respecte une certaine propriété globale du système.

Les dimensions de ce sous ensemble sont les résultats de distribution par le concepteur d'intervalles de R associés aux valeurs des paramètres physiques de nature différentes (mécanique, électrique, …).

Les intervalles des paramètres peuvent être choisis (en pré-conception) ou subis (en production) ou en exploitation par dérive dans le temps.

Définition des zones de fonctionnement

- Une zone du fonctionnement désiré (ZFD) : est une zone de l'espace déterminée par des contraintes posées sur les variables d'état (par exemple $\|x_1\| \leq v_1$, $\|x_2\| \leq v_2$).
- L'intersection entre la ZFD et le domaine atteignable (DA) solution de l'inclusion différentielle qui regroupe toutes les trajectoires possibles constitue la zone du fonctionnement possible du système (ZFP).

Le cas où le domaine atteignable est inclus complètement dans la zone du fonctionnement désiré ($DA \subset ZFD$) est le cas idéal car dans ce cas le vecteur d'état est inclus dans la ZFD quelque soient les variations des paramètres.

Application à l'exemple mécatronique

Pour illustrer ce concept, on reprend l'exemple du système de pilotage présenté dans le chapitre 2.

Comme on l'a défini précédemment (dans le chapitre introduction et problématique) il existe deux types d'incertitude :

- Variation paramétrique qui n'est pas en fonction du temps. L'incertitude est invariante avec le temps. On ne connaît pas la valeur initiale du paramètre. On a dans ce cas, un échantillon parmi n échantillons. On ne connait ni la valeur initiale ni la valeur finale du paramètre.
- Variation paramétrique qui évolue en fonction du temps. Dans ce cas on connait la valeur initiale du paramètre. Dans ce cas on étudie le cas d'un seul exemplaire dont les paramètres évoluent dans le temps.

Dans notre exemple on prend le cas d'un système unique à variation paramétrique qui évolue dans le temps. On connait les valeurs extremum du paramètre mais on n'a aucune idée sur sa loi d'évolution dans l'intervalle de variation.

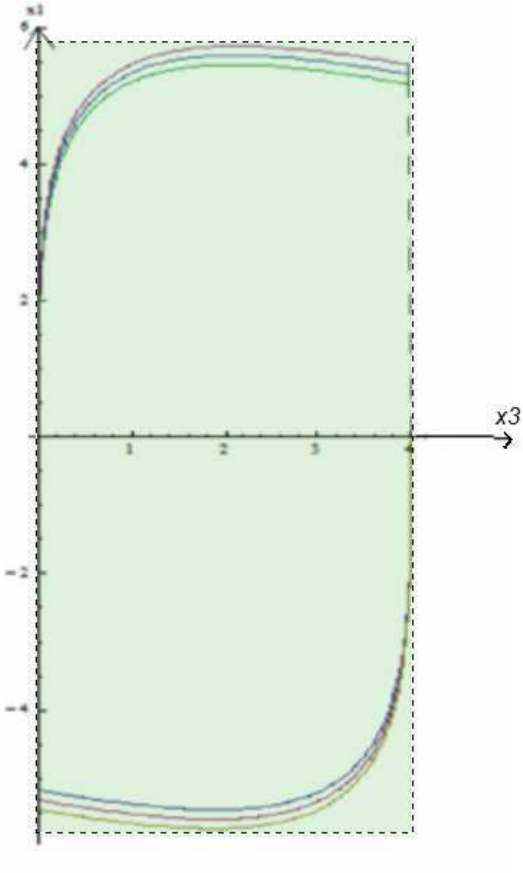


Figure 49 Le domaine atteignable est inclus complètement dans la zone du fonctionnement désirée ZFD

Dans ce premier cas, le domaine atteignable est complètement inclus dans la ZFD. Le vecteur d'état du système est alors inclus dans la ZFD, donc il satisfait toutes les contraintes imposées sur les variables d'état et par conséquence le système assure toutes les fonctionnalités en répondant aux exigences du cahier des charges.

Un autre cas de figure est quand l'intersection entre le domaine atteignable et la zone du fonctionnement désiré est non vide : ($DA \bigcap ZFD \neq \Phi$). Dans ce cas, le système pourrait assurer certaines de ses fonctions mais en plus des zones de fonctionnement désiré(ZFD) il présente des zones de fonctionnement non désiré ZFND.

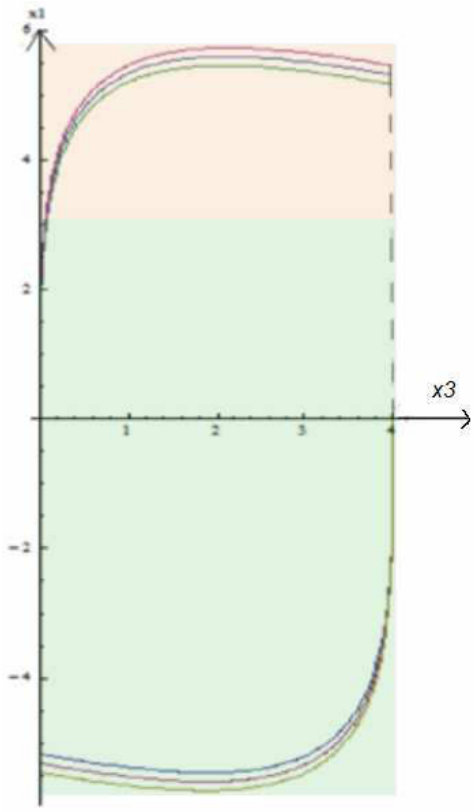


Figure 50 Les zones de fonctionnement désiré et non désiré du système

La zone verte désigne la zone du fonctionnement désiré.

La zone marron désigne la zone du fonctionnement non désiré.

La partie du domaine atteignable qui appartient à la zone marron représente une zone de fonctionnement non désiré par le système. Cette zone correspond à une certaine variation des paramètres ; dans ce cas il faut revoir les intervalles du tolérancement des paramètres et les redéfinir de sorte que le système corresponde aux exigences du cahier des charges et reste dans la ZFD.

2- Les propriétés des systèmes à paramètres variables

Dans le cadre des travaux de cette thèse qui se portent sur le tolérancement et l'incertitude paramétrique, on définit les propriétés du système comme étant l'appartenance de son vecteur d'état à des zones simples de R^2 ou complexes de R^n de l'espace d'état à un instant donné.

Il faut mesurer les distances entre ces zones pour assurer le bon fonctionnement du système et pour respecter les propriétés (ou les contraintes du cahier des charges). Cela nécessite un outil métrique pour mesurer et visualiser les écarts entre ces ensembles.

3- Application de la distance de Hausdorff sur les variations paramétriques de systèmes

3.1- Rappel de définitions

Soit les ensembles S, $\dot{S}$et S_1 :

- Conv(S) : enveloppe convexe de S, c'est le plus petit ensemble convexe qui contient S.
- S convexe $\stackrel{def}{=}\forall$ (x, y) $\in$S,$\forall$ t $\in$ [0,1] , (tx +(1-t) y) $\in$S

Soit :

- $S_1 \subset$ Conv(S)
- $\dot{S}$ intérieur de S alors $\dot{S} \subset S$,
- $x \in \dot{S}$ si $S \in \nu(x)\ voisinage\ de\ x \stackrel{def}{=} \exists\ \Omega \in$ O ouverts de $\mathbb{R}^n$, $\{x\} \subset \Omega \subset \nu(x)$
- Chemin $\mathcal{C}$(x,y) : est une application continue f : [0,1] $\rightarrow \mathbb{R}^n$, tel que f(0)= x et f(1)= y ,(x,y) = Im([0,1])

3.2- La distance de Hausdorff :

En topologie la distance de Hausdorff est un outil topologique qui mesure l'éloignement de deux sous-ensembles d'un espace métrique sous-jacent. Elle porte le nom du mathématicien allemand Felix Hausdorff.

Soit un espace métrique (*E*,*d*). Soient S et S_1 deux sous-ensembles compacts non vides de *E*.

Soit U la boule unité

- $h_{\infty}^{0}(S, S_1) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, S \subset S_1 + d\,U \}$
- $h_{\infty}^{0}(S_1, S) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, S_1 \subset S + d\,U \}$ (6.1)

La distance de Hausdorff entre S et S_1 est définie comme étant le plus petit nombre réel d tel que :

Distance de Hausdorff : $h_{\infty}(S, S_1) \stackrel{def}{=} \max \{ h_{\infty}^{0}(S, S_1), h_{\infty}^{0}(S_1, S) \}$ (6.2)

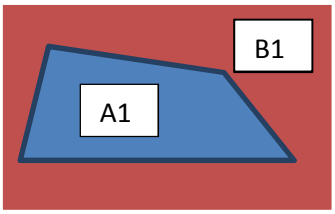
Rappel des propriétés d'une distance :

La distance de Hausdorff $h_{\infty}(S, S_1)$ est nulle si et seulement si $S = S_1$ et elle augmente lorsque des différences de plus en plus importantes apparaissent entre S et S_1.

- $h_{\infty}^{0}(S_1, S_1) = 0$ (6.3)
- $h_{\infty}^{0}(S, S_1) = h_{\infty}^{0}(S_1, S)$ (6.4)
- $h_{\infty}^{0}(S, S_2) \leq h_{\infty}^{0}(S, S_1) + h_{\infty}^{0}(S_1, S_2)$ (6.5)

3.3- Application au Tolérancement

Cas 1 : l'inclusion différentielle est dans le gabarit de tolérance



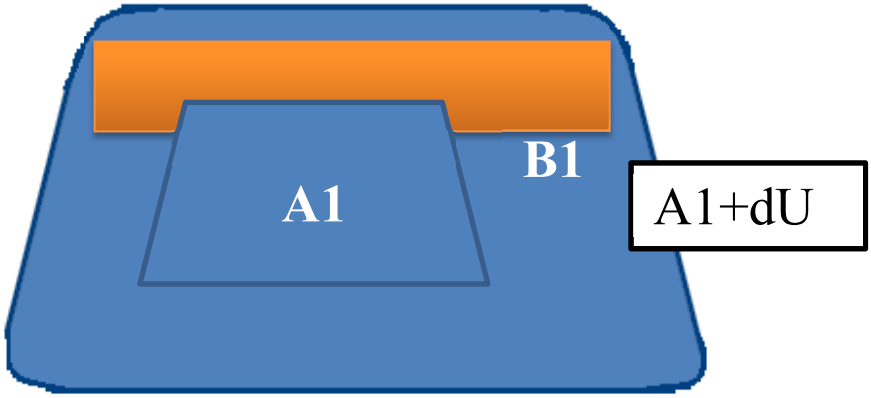
La zone bleue (A1) représente le domaine atteignable solution de l'inclusion différentielle, alors que la zone rouge (B1) représente la zone du fonctionnement désiré ZFD.

Rappel

$h_{\infty}^{0}(S, S_1) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, S \subset S_1 + d\,U \}$ (6.6)

ici

$h_{\infty}^{0}(A_1, B_1) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, A_1 \subset B_1 + d\,U \} = 0$ (6.7)

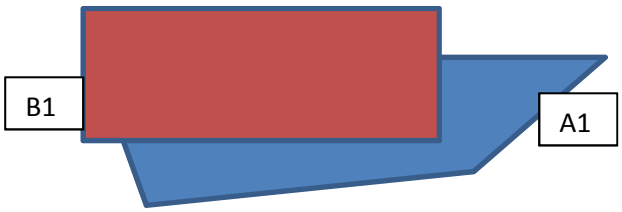


Si on souhaite que le domaine atteignable englobe toute la ZFD alors on cherche la distance de Hausdorff nécessaire pour atteindre cet objectif.

$h_{\infty}^{0}(B_1, A_1) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, \; B_1 \subset A_1 + d\,U \} = d_1$ (6.8)

Il faut augmenter la zone (A1) d'un rayon d pour inclure la ZDF (B1).

Cas 2 : l'inclusion différentielle n'est pas dans le gabarit de tolérance



$h_{\infty}(A_1 , B_1) \stackrel{def}{=} \max \{ h_{\infty}^{0} (A_1 , B_1) , h_{\infty}^{0} (B_1, A_1) \}$ (6.9)

$h_{\infty}^{0}(A_1, B_1) \stackrel{def}{=} \inf \{ d \in \mathbb{R}^{+}, \; A_1 \subset B_1 + d\,U \}$ (6.10)

d représente la distance pour que l'IDI contienne la zone de fonctionnement définie par le gabarit B dans le cahier des charges. On traite d'atteignabilité.

4- Application des métriques sur l'inclusion différentielle impulsionnelle au dimensionnement

L'inclusion différentielle impulsionnelle est un outil qui permet de déterminer le domaine atteignable par le système pour ses différents états. On illustre ce concept sur notre exemple mécatronique système de pilotage (défini chapitre 2) avec une variation sur la résistance. On pose des contraintes sur les variables d'état ; ces contraintes supplémentaires délimitent la zone du fonctionnement désiré ZFD du système. On fait face à trois cas :

- Zone de fonctionnement désiré fixe et intervalle de variation d'un paramètre à bornes variables.
- Zone de fonctionnement désiré variable et intervalle de variation d'un paramètre à bornes fixes.
- Zone de fonctionnement désiré fixes et intervalle de variation d'un paramètre à bornes fixes.

4.1- Zone de fonctionnement désiré fixe et intervalle de variation d'un paramètre à bornes variables :

Dans le cas de zone de fonctionnement désiré fixe et intervalle de variation d'un paramètre à bornes variables, le cahier des charges fixe la zone de fonctionnement désiré du système (ZFD) par des contraintes sur les variables d'état. Quand à l'intervalle du tolérancement, il est à bornes variables, on peut agir sur les paramètres pour élargir ou diminuer l'intervalle de variation (dans la phase de conception ou du pré-dimensionnement par exemple).

Pour ce premier cas on a une zone de fonctionnement désiré (ZFD) déterminé par les valeurs maximales que peut prendre le courant et la position, la résistance du moteur varie dans un intervalle de $r_n \pm 25\%$.

Avec r_n la valeur nominale de la résistance r.

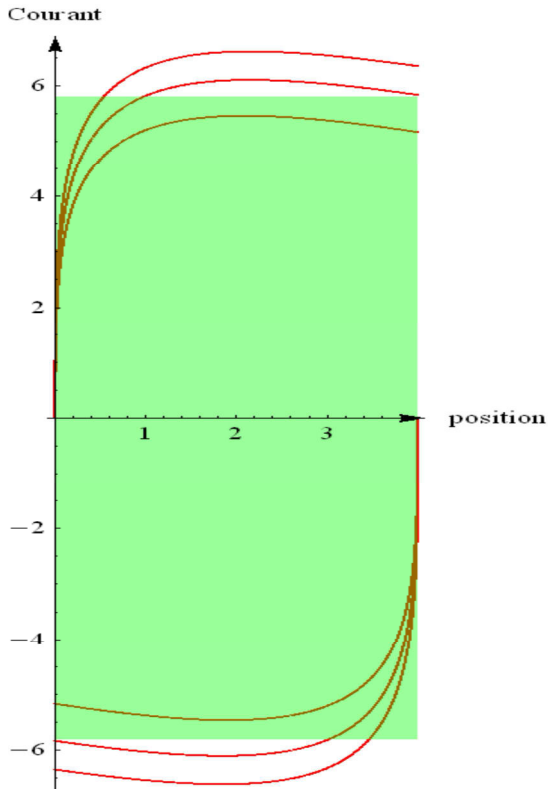


Figure 51 Projection de l'inclusion différentielle impulsionnelle sur le plan (position / courant) et ZFD (zone verte)

Cette variation des paramètres mène le système à dépasser la ZFD lors des différents états. Une solution est de faire varier l'intervalle de variation des paramètres. Par exemple si on réduit la variation de $r_n \pm 25\%$ à $r_n \pm 5\%$ on obtient le résultat suivant.

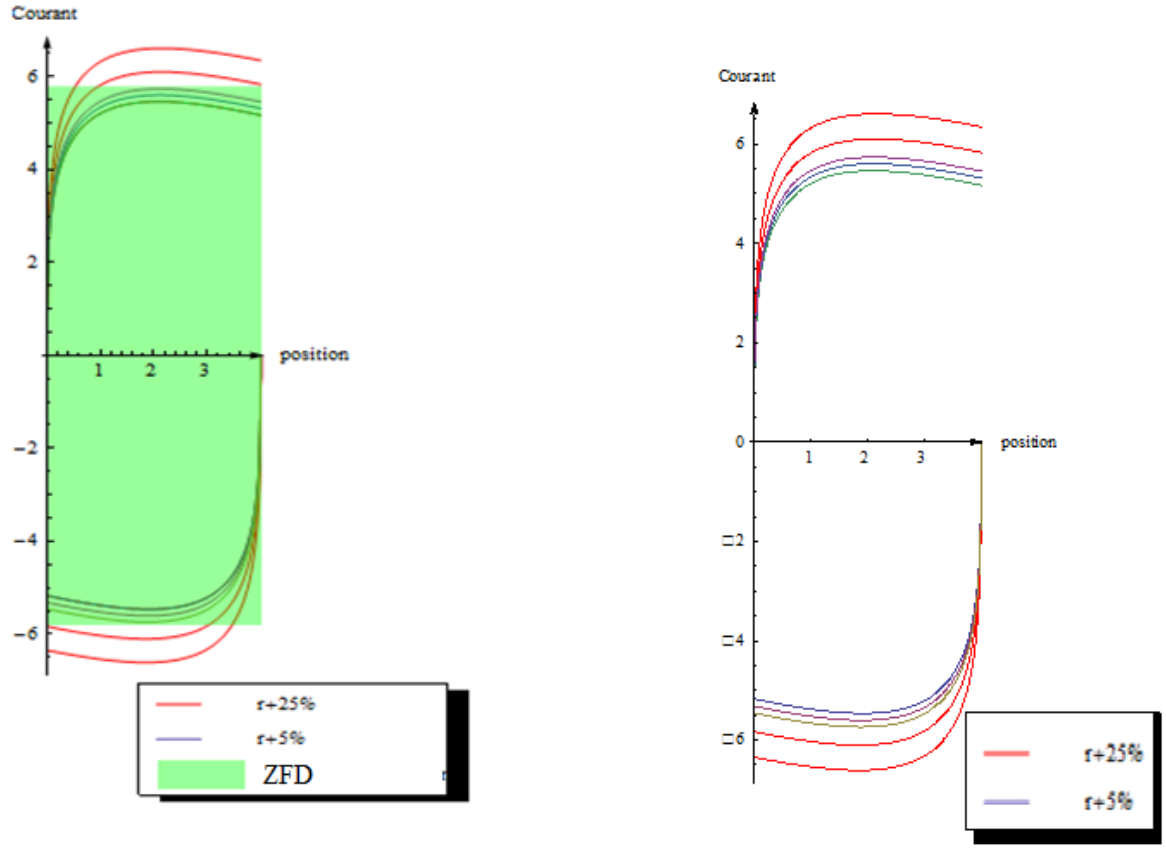


Figure 52 Projection de l'Inclusion différentielle impulsionnelle avec variation de la résistance de $\pm 25\%$ et de $\pm 5\%$

Pour une variation de la résistance r de $\pm 25\,\%$, le domaine atteignable occupe une certaine zone de l'espace qui dépasse la ZFD. Étant donné que les composants du système ne peuvent pas assurer un tel fonctionnement, on a choisi de modifier les paramètres pour réduire le domaine atteignable par l'inclusion différentielle impulsionnelle et pour assurer le bon fonctionnement du système (ne pas dépasser la ZFD).

La réduction de l'intervalle de variation des paramètres se traduit par un rétrécissement du domaine atteignable de l'IDI par le système.

Chaque domaine atteignable correspondant à une variation donnée est limité par un « polygone » qui représente la ZFD.

La distance entre les deux « polygones » correspondants respectivement aux variations de la résistance r par $\pm 25\%$ et $\pm 5\%$ mesurée par la méthode du Hausdorf est « d ».

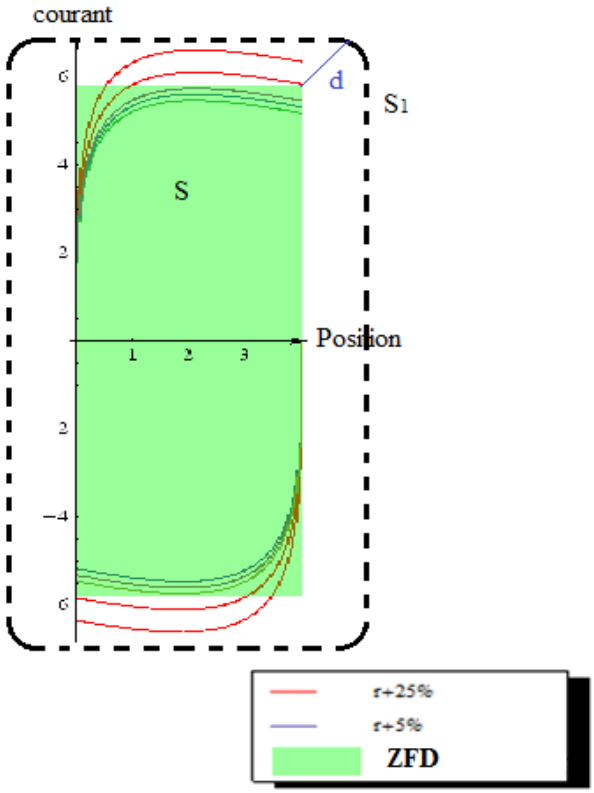


Figure 53 Projection des domaines atteignables du système pour différentes variations de la résistance et la distance de Hausdorf .

Notons la ZFD par S,

S1 est le rectangle représenté par des pointillés noir.

$h_{\infty}^{0}(S_1, S) \overset{\text{def}}{=} \inf \{ d \in \mathbb{R}^{+}, \ S_1 \subset S + d\,U \}$ (6.11)

L'objectif est d'augmenter la ZFD ; la méthode consiste à établir une boucle d'ajustement.

A chaque itération de la boucle, d augmente (ou diminue) et sera noté par d_i.

A chaque pas :

On fait varier le paramètre v ;

1- On augmente l'intervalle de variation $[v_{\min}, v_{\max}] := [v_{\min} - \delta, v_{\max} + \delta]$

2- On calcule l'inclusion différentielle impulsionnelle

3- On calcule d

$$d = \inf\{d \in R^+, \quad S1 \subset S + dU\} \qquad (6.12)$$

- Si d converge $\underset{i=\infty}{Lim}(r_i - r_{i-1}) \to 0$ alors on continue l'augmentation de v $[v_{\min} - \delta, v_{\max} + \delta]$ jusqu'à d $< \xi$
- Sinon (d diverge) On diminue la variation de v ($[v_{\min} + \delta, v_{\max} - \delta]$)

4- On répète les étapes 1 ,2 et 3 jusqu'à d $< \xi$

Remarque : pour que la distance d converge (soit inférieur à une certaine valeur ξ) , il faut tester une variation croissante ou décroissante du paramètre v.

Dans le cas où la zone du fonctionnement désiré est fixe et l'intervalle de variation d'un paramètre est à bornes variables, le cahier des charges est prédéterminé, les contraintes sont fixées mais on peut changer les caractéristiques des composants pour satisfaire les exigences. Sur le cycle de vie, cela revient à faire des micro-boucles au niveau de la conception détaillée.

4.2- Zone du fonctionnement désiré variable et intervalle de variation à bornes fixes :

Supposons le cas où la zone du fonctionnement désiré est variable et l'intervalle de variation est à bornes fixes quand, par exemple, le système est déjà fabriqué et qu'on ne peut pas changer ses composants. Les intervalles de variation des paramètres sont fixes. La ZFD est variable, on peut modifier les conditions sur les variables d'état pour la modifier. L'agrandissement et le rétrécissement de la ZFD sont faits de façon à inclure le domaine atteignable de l'inclusion différentielle impulsionnelle.

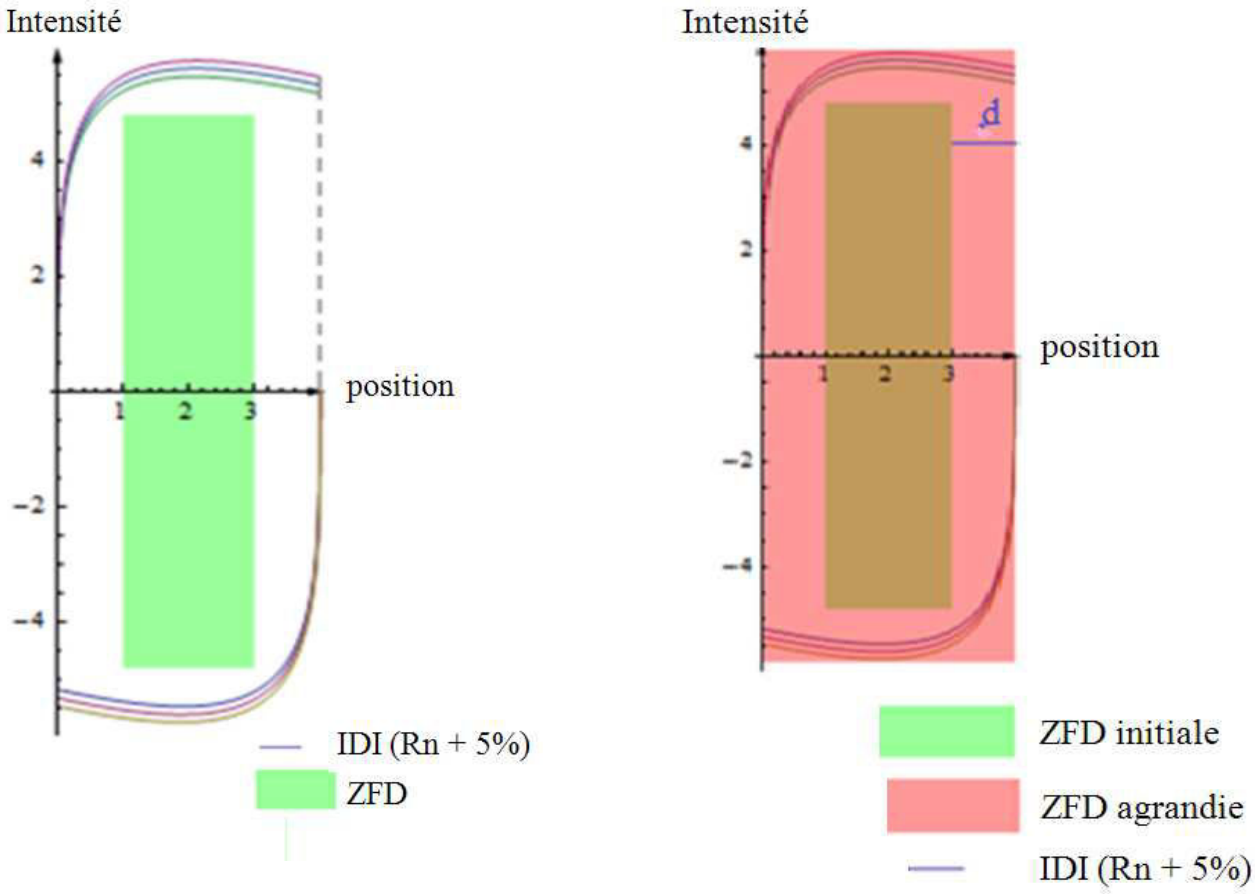


Figure 54 Projection de l'inclusion différentielle impulsionnelle et ZFD variables

Dans la figure 54 on montre que l'agrandissement (ou éventuellement le rétrécissement) du ZFD par changement des conditions du vecteur d'état permet d'inclure le domaine atteignable de l'inclusion différentielle impulsionnelle.

La distance entre les deux zones de fonctionnement est « d ». A chaque itération de la boucle, d augmente (ou diminue) et sera noté par d_i.

A chaque pas :

On fait varier les conditions sur le vecteur d'état ;

1- On augmente la ZFD en modifiant les conditions sur les variables d'état
2- On calcule l'inclusion différentielle impulsionnelle
3- On calcule d

- Si d converge : $\lim_{i=\infty}(d_i - d_{i-1}) \to 0$ alors on continue l'augmentation de

la ZDF $(x_i + \delta)$ jusqu'à d $< \xi$

- ❖ Sinon (d diverge) On diminue la ZFD $(x_i - \delta)$ en modifiant les conditions sur les variables d'état

4- On répète les étapes 1 ,2 et 3 jusqu'à d $< \xi$

Cette approche peut être utilisée en deux cas :

- soit la technologie n'existe pas, les composants qui peuvent satisfaire les conditions du cahier des charges n'existent pas.

- soit le coût des composants est trop élevé, dans ce cas on met en cause le cahier des charges et on modifie les contraintes.

Sur le cycle de vie cela correspond à faire des boucles entre la phase de conception et la phase de réalisation du cahier des charges.

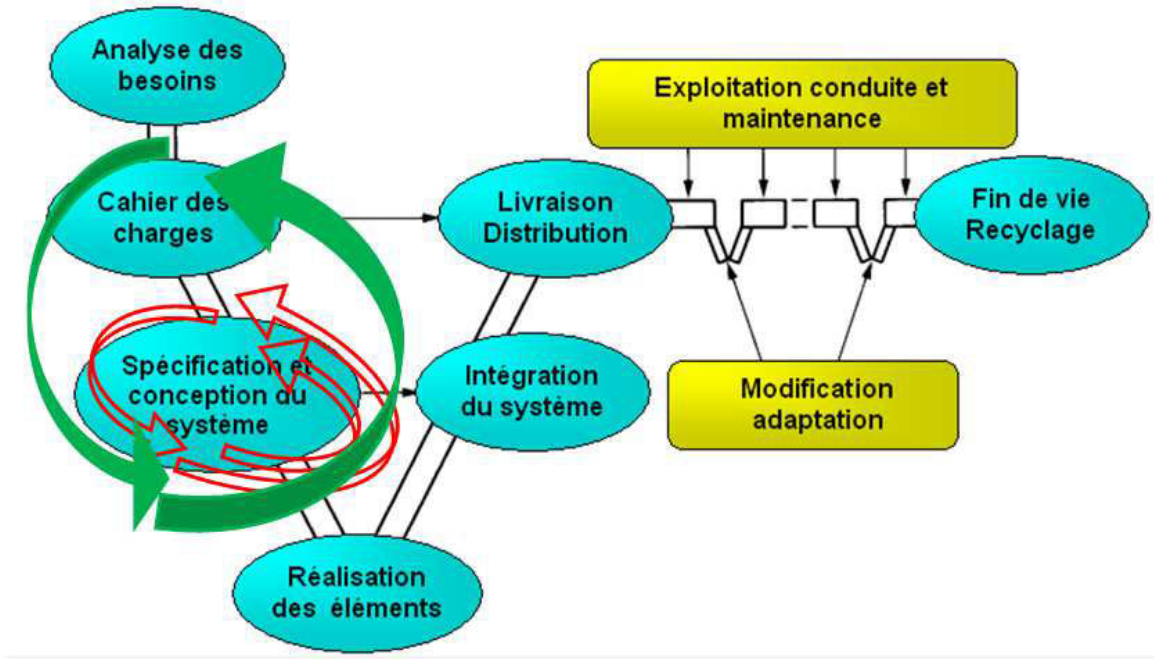


Figure 55 Cycle de vie et les boucles de conception

4.3- Zone du fonctionnement désiré fixe et intervalle de variation à bornes fixes :

On se place dans le cas où la ZFD est fixe et les paramètres de variation sont à bornes fixes aussi. Si on veut chercher la zone du domaine atteignable qui satisfait certaines contraintes du

cahier des charges sur les performances ou sur les paramètres on trouve une approche dans la théorie de viabilité. En effet, on suppose qu'on a un certain domaine atteignable. La ZFD est donnée par des contraintes sur les performances. Donc le système est dit viable selon Aubin, s'il évolue dans cette zone délimitée par les contraintes. On cherche un point initial appartenant au domaine atteignable et dont au moins une trajectoire au départ de ce point ne viole pas les contraintes et évolue dans la ZFD au cours du temps. Le noyau de viabilité correspond au plus grand ensemble qui englobe toutes les trajectoires inclues dans le domaine atteignable et la ZFD à la fois. La distance entre le noyau de viabilité et le domaine atteignable peut être calculé en utilisant la distance de Hausdorff.

$$h_{\infty}^{0}(\text{Viab}, \text{DA}) = \inf\{ d \in \mathbb{R}^{+}, \ \text{Viab} \subset \text{DA} + d\,U \} = 0 \qquad (6.13)$$

$$h_{\infty}^{0}(\text{DA}, \text{Viab}) = \inf\{ d \in \mathbb{R}^{+}, \ \text{DA} \subset \text{Viab} + d\,U \} = d \qquad (6.14)$$

D'où la distance de Hausdorff est

$$d = \max\{ h_{\infty}^{0}(\text{Viab}, \text{DA}), h_{\infty}^{0}(\text{DA}, \text{Viab}) \} \qquad (6.15)$$

Dans le chapitre « les inclusions différentielles impulsionnelles » on a présenté la théorie de viabilité et on a présenté un algorithme de construction de noyau de viabilité. On donne ici un schéma représentatif du résultat (figure 56).

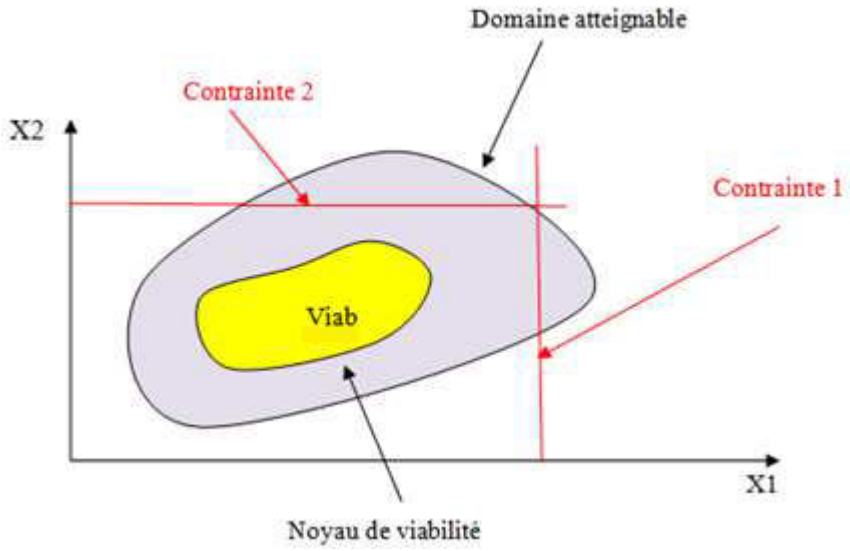


Figure 56 Domaine atteignable et noyau de viabilité

Remarque : dans le cas où la ZFD est variable et les paramètres de variation sont aussi à bornes variables, on se ramène à l'un des cas précédents c'est à dire soit on fixe la ZFD et on fait varier les paramètres soit l'inverse. Le choix se fait selon le contexte technologique et économique.

4.4- Conclusion sur les deux approches

On conclue que chaque variation sur les variables d'état ou sur les paramètres du système se traduit par une variation sur le domaine atteignable du système. Cette variation est désormais calculable en utilisant la distance de Hausdorff.

Cette double approche peut être appliquée au tolérancement. En effet, suivant la phase du cycle de vie du système on peut agir soit sur les composants soit sur les conditions d'utilisation (conditions sur les variables d'état) ; cette variation affecte l'atteignabilité du système.

5- Application dans le plan :

5.1- 1er cas : délimitation du domaine atteignable

On se place dans le cas où on dispose d'une inclusion différentielle ou d'une inclusion différentielle impulsionnelle d'un système ; le domaine atteignable résultant est le plus petit domaine qui englobe toutes les trajectoires possibles que peut avoir le système pour l'ensemble des variations.

Soit Tn la trajectoire nominale du système avec la valeur nominale du paramètre (on suppose que la variation est de la forme $v = v_n \pm \delta$) cette trajectoire peut être limitée par un ensemble nominal (dans la figure ci-dessous noté par Ens1)

Soit Ens 2 le plus petit ensemble qui inclue l'inclusion différentielle.

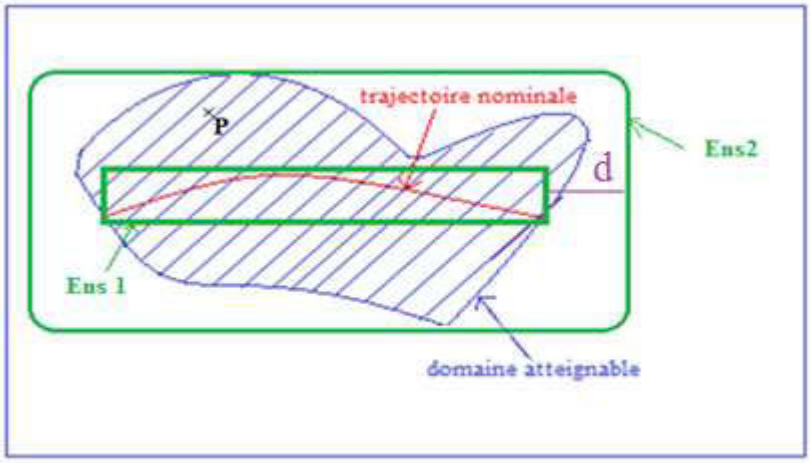


Figure 57 Domaine atteignable et distance de Hausdorff

Soit un point P qui appartient au domaine atteignable alors :

La distance entre tout point P et Ens1 est inférieure ou égale à la distance de Hausdorff d.

$$d(P, Ens1) = \inf_{y \in DA} Py \leq d$$

La semi distance de Hausdorff :

$$h_{\infty}^{0}(\text{Ens2, Ens1}) = \inf\{ d \in \mathbb{R}^{+},\ \text{Ens2} \subset \text{Ens1} + d\,U \} = d \qquad (6.16)$$

$$h_{\infty}^{0}(\text{Ens1, Ens2}) = \inf\{ d \in \mathbb{R}^{+},\ \text{Ens1} \subset \text{Ens2} + d\,U \} = 0 \qquad (6.17)$$

D'où la distance de Hausdorff est

$$d = \max\{ h_{\infty}^{0}(\text{Ens2, Ens1}),\ h_{\infty}^{0}(\text{Ens1, Ens2}) \} \qquad (6.18)$$

Ce dernier résultat nous donne un outil pour le tolérancement. En effet, il suffit d'avoir la trajectoire nominale (qui correspond à la trajectoire du système sans prendre en compte les variations paramétriques) ; on délimite cette trajectoire par un premier polygone (« Ens1 »). Ce dernier n'est pas forcément inclus dans le domaine atteignable ; il pourrait bien y avoir un ensemble de points qui appartiennent à « Ens1 » mais qui n'appartiennent pas au domaine atteignable. On utilise la notion de distance de Hausdorff pour déterminer le plus petit ensemble qui inclut à la fois « Ens 1 » et le domaine atteignable. Cette distance de Hausdorff notée « d » est la plus grande entre tout point du domaine atteignable et « Ens1 ».

Ainsi on pourra limiter tout domaine atteignable dans R^2 ou sa projection dans un plan par un ensemble moyennant la distance de Hausdorff.

5.2- 2éme cas application dans le cas d'automate hybride à invariant polyèdre

Dans le cas d'un automate hybride à invariant polyèdre, les états sont généralement disjoints. Pour assurer la continuité des états, on peut calculer la distance entre deux états disjoints déterminés par des invariants. Cela revient à chercher comment augmenter l'invariant d'un état pour que lorsqu'on sorte d'un état 1 on soit directement dans un état 2. La distance de Hausdorff dans ce cas est la plus petite distance entre les invariants qui permet la continuité des états et qui évite les sauts.

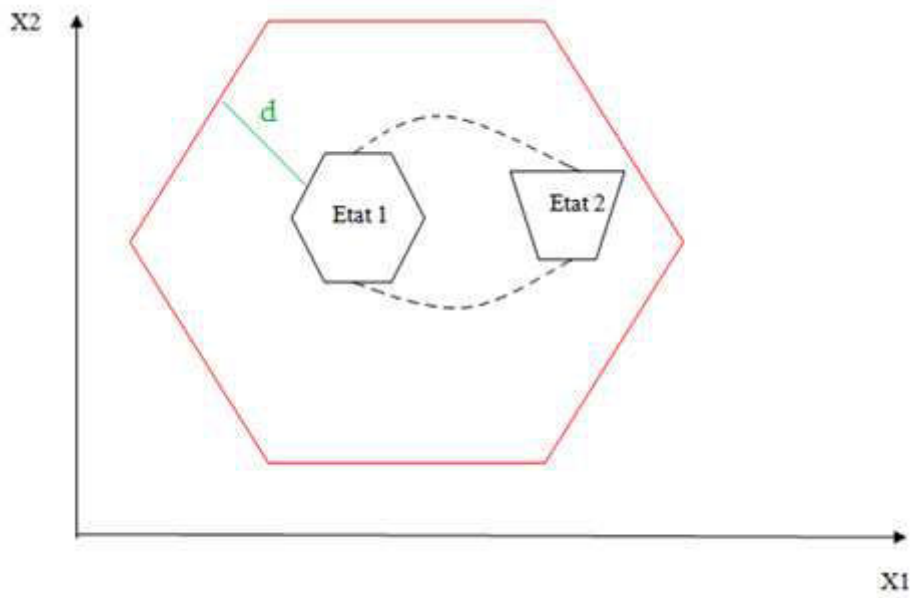


Figure 58 Application des distances de Hausdorff à l'automate hybride à invariant polyèdre

Cette approche nous ouvre les champs de calcul de distance entre les états et les ensembles, il propose une utilisation de la métrique pour le calcul des chemins des états.

Conclusion

Ce chapitre est une application générale des différents outils utilisés et résultat obtenus dans les chapitres précédents pour le tolérancement dans l'espace d'état. On a commencé par établir une définition de la zone du fonctionnement désiré et les différents cas de figure qu'on peut avoir de son intersection avec le domaine atteignable. Puis, on a proposé des propriétés des systèmes liées à ces zones de l'espace d'état.

Nous avons introduit la distance de Hausdorff, et son application dans le domaine du tolérancement. Pour cela on a étudié les cas suivants : un système avec une zone de fonctionnement désiré fixe et des intervalles de variations à bornes variables, un système avec une zone de fonctionnement désiré variable et des intervalles de variations à bornes fixes et le cas où la zone de fonctionnement désiré est fixe et des intervalles de variations à bornes fixes aussi. Pour respecter la propriété du système on peut varier soit ses performances soit ses paramètres. On a montré que chaque variation de la ZFD ou de l'intervalle de variation est mesurable en utilisant la distance de Hausdorff.

Ensuite, nous avons donné quelques exemples d'application dans le plan. On a commencé par délimiter le domaine atteignable entre deux ensembles en utilisant la distance de Hausdorff. Enfin, on a présenté un cas de possibilité de futur développement associant les inclusions différentielles impulsionnelles, les automates hybrides et la distance de Hausdorff.

Conclusion et perspectives

Dans cette thèse nous avons proposé une méthode d'étude de variation paramétrique pour les systèmes mécatroniques continus et hybrides puis une approche du tolérancement mécatronique.

Le problème de variations paramétriques a suscité l'intérêt de différents chercheurs car c'est un problème présent dans différents domaines mais l'appellation diffère ; dans le domaine mécanique on peut avoir un problème de jeu, ou de tolérancement géométrique ; ou des perturbations et incertitude si on étudie des systèmes dans le domaine fréquentiel.

On a défini des critères et comparé quelques méthodes existantes pour l'étude de variation paramétrique. La méthode stochastique de Monté Carlo s'est avérée non satisfaisante pour notre cas d'étude, en effet elle ne présente que 2 à 5% du résultat réel comparée à la méthode déterministe des inclusions différentielles.

La méthode de l'inclusion différentielle est une extension des équations différentielles. C'est un outil mathématique utilisé dans l'automatique et plus précisément dans la simulation par Raczynski. Cette méthode consiste à résoudre des équations différentielles dont un ou plusieurs de ses paramètres varie(nt) dans un intervalle. La solution fournie par cette méthode est le domaine atteignable. Ceci représente un domaine qui englobe toutes les solutions possibles. Chaque solution dans ce domaine est une solution optimale pour une instance de variation. Pour résoudre le problème d'inclusion différentielle nous avons développé un algorithme d'optimisation qui se base sur la méthode du steepest descent, mais de plus le notre permet d'avoir l'optimum global. Cet algorithme a été utilisé pour la maximisation de l'Hamiltonien à chaque pas de variation dans l'intervalle du tolérancement. Cette méthode permet l'étude de variation paramétrique pour un système continu.

Dans la pratique tous les systèmes ne sont pas purement continus. Un système mécatronique peut avoir une évolution continue interrompue par des sauts discrets. Il s'agit dans ce cas de système dynamique hybride. Le formalisme d'automate hybride est un outil de modélisation de système hybride qui ne présente pas de variation paramétrique. Mais dans le cas où ce système présente des paramètres variables, l'outil de modélisation le plus approprié devient selon notre approche l'inclusion différentielle impulsionnelle.

L'inclusion différentielle impulsionnelle présente une extension des automates hybrides. Nous avons repris ce formalisme et identifié ses ensembles sur un système mécatronique. Nous avons développé des algorithmes de résolution des inclusions différentielles impulsionnelles pour un puis pour n paramètres variables.

Ensuite, on a introduit quelques outils de programmation comme Modelica et Mathematica. Le premier est un langage de programmation orientée objet. C'est un langage multi physique qui contient des librairies des différents domaines : mécanique, électronique, thermodynamique… cet outil est très performent pour la programmation et la simulation. Mais dans notre algorithme on manipule plusieurs équations différentielles, des inclusions différentielles des systèmes hybrides et, pour cela, on a besoin d'un langage qui combine aussi bien les possibilités de modélisation , de simulation et des performances mathématiques comme le calcul symbolique par exemple. On a choisi Mathematica comme outil de programmation de nos algorithmes. Cet outil présente de plus une passerelle entre Modelica et Mathematica à travers le langage MathModelica.

Après avoir fixé notre choix sur le logiciel de Mathematica on a présenté une implémentation de nos algorithmes développés dans les chapitres précédents.

On a choisi un exemple d'application simple. Il s'agit d'un système de pilotage qui comporte un moteur à courant continu, une masse et une vis sans fin.

On a commencé par programmer l'algorithme des inclusions différentielles en appliquant de petites et de grandes variations à la résistance et au pas de la vis. Selon la grandeur de variation, le fonctionnement du système pourra être à l'intérieur ou à l'extérieur d'une certaine zone de l'espace d'état appelée zone du fonctionnement désiré (ZFD).

Pour programmer les algorithmes développés pour l'inclusion différentielle impulsionnelle, nous avons modifié le fonctionnement de notre système d'application. Nous lui avons ajouté des sauts pour qu'il ait un fonctionnement hybride.

La programmation de l'algorithme de l'inclusion différentielle impulsionnelle nécessite l'appel des fonctions spécifiques de Mathematica qu'on a détaillées au fur et à mesure de la programmation.

On a présenté les résultats des programmes de l'inclusion différentielle impulsionnelle pour un puis n paramètres variables.

Sur les résultats de simulation de l'algorithme de l'inclusion différentielle impulsionnelle aussi bien pour un ou plusieurs paramètres variables la date de franchissement de transition discrète n'est pas unique. Ceci est dû au fait que les événements déclencheurs de transition sont internes au système d'application et dépendent de ses variables d'états. Ces aspects pourront donné lieu à des analyses ultérieurs.

Ensuite, on a comparé notre approche avec d'autres outils de résolution d'automates hybrides et d'inclusion différentielle. On a commencé par les automates hybrides à invariant polyèdre. L'espace d'état continu est divisé en un ensemble de parties disjointes qui couvrent tout l'espace d'état. L'invariant est écrit sous la forme d'inéquations linéaires. Il peut être considéré comme un outil de résolution des systèmes hybrides avec du tolérancement aussi bien sur les paramètres que sur les performances.

On a également présenté les inclusions différentielles polygonales qui sont un outil géométrique de construction des inclusions différentielles. Sa définition se base sur la partition de l'espace d'état en polygone et l'intersection du résultat avec ces polygones

Puis, on a présenté un algorithme de résolution de l'inclusion différentielle appelé l'algorithme pratique ; il se base sur la méthode de Heun en discrétisant l'espace d'état et le temps.

On a établi une comparaison entre cet algorithme et notre algorithme utilisé pour la résolution de l'inclusion différentielle et on a montré que notre algorithme est plus simple de point de vue des données d'entrée et son résultat est plus complet car il nous fournit le domaine atteignable dans l'espace R^n, alors que le résultat obtenu par l'algorithme pratique est une projection du domaine atteignable dans l'espace R^2.

En dernière partie, on a repris les différents outils utilisés et résultats obtenus dans les chapitres précédents pour définir et affiner notre approche du tolérancement. On a défini la zone du fonctionnement désiré (ZFD) et les différents cas de figures qu'elle peut avoir en son intersection avec le domaine atteignable. On a présenté un outil métrique pour le calcul des distances entre ces différents ensembles.

Dans le cas où la ZFD et les intervalles de variations sont à bornes fixes. Nous nous sommes intéressés à un cas de figure où le domaine atteignable n'est pas entièrement inclus dans la ZFD. On a eu recours dans ce cas à la théorie de viabilité pour trouver le noyau de viabilité.

En fin du chapitre 6 on a donné quelques exemples d'application dans le plan. On a commencé par délimiter le domaine atteignable entre deux ensembles en utilisant la distance de Hausdorff. Enfin, on a présenté un cas d'application pour les automates hybrides à invariant polyèdre où on peut augmenter l'invariant pour assurer une continuité des états initialement disjoints .

Perspectives

On pourra développer notre méthode en l'appliquant à un système mécatronique plus complexe. Notre exemple d'application constitue le système de base d'un système de freinage utilisé dans le domaine de l'aéronautique.

On pourra aussi transférer les programmes de Mathematica vers Modelica. Il existe aussi des langages de transfert comme Mathmodelica qui est une extension de Modelica dans le langage de Mathematica et le SystemModeler qui fait partie de Mathematica et qui inclut toutes les librairies de Modelica/Dymola.

L'objectif de l'étude du tolérancement mécatronique est de prévoir le fonctionnement des systèmes mécatroniques munis d'incertitudes. Avec les méthodes présentées ci-dessus on pourra avoir un domaine atteignable qui englobe toutes les solutions possibles. Nous proposons de déterminer des surfaces qui limitent le domaine atteignable. Ceci permettra de limiter le fonctionnement du système dans un volume ou une surface si on travaille dans un plan 2D. Ce travail nécessite le calcul des distances minimales qui séparent le domaine atteignable du « volume du fonctionnement ».

On a montré qu'une variation dans l'intervalle du tolérancement peut être traduite par une variation sur le domaine atteignable ; cette variation peut être mesurée en utilisant les distances de Hausdorff. Un algorithme pourra être développé et implémenté pour visualiser les résultats.

En perspective, ces travaux de thèse peuvent être poursuivis et complétés dans le cadre du projet européen Eurêka ITEA2 MODRIO. L'objectif du projet MODRIO est d'étendre les environnements de modélisation et de simulation basés sur les standards ouverts pour accroître la sécurité des systèmes complexes d'énergie et de transport, leur fiabilité et leurs performances (de la conception à l'exploitation pendant toute leur durée de vie). Une des tâches de ce projet concerne la modélisation et la simulation des systèmes hybrides à paramètres variables [MODRIO 2013].

Bibliographie

[Alazard et al, 1999] D. Alazard, C. Cumer, P. Apkarian, M. Gauvrit, G. Ferreres, (1999) "*Robustesse et commande optimale*". Cépaduès-Edition.

[Alur,1999] R. Alur, S. Kannan, Salvatore La Torre "Polyhedral flows in hybrid automata", *Hybrid Systems: Computation and Control, Proceedings of the Second International Conference*, (HSCC'99), Lecture Notes in Computer Science 1569, pages 5--18, Springer-Verlag, 1999.

[Alur, 2000] R. Alur, T A. Henzinger. G. Lafferriere. and G J. Pappas. (2000) "*Discrete abstractions of hybrid systems*," *Proc. IEEE*, vol. 88, pp. 971–984, July 2000.

[Artstein, 1994] Artstein Z, "*First order approximations for differential inclusions*". Set-Valued Anal 2:2 17, 1994

[Asarin, 2001] E. Asarin, G. Schneider, and S. Yovine. "*On decidability of the reachability problem of planar differential inclusions*".In HSCC'01.LNCS 2034, Springer, 2001.

[Asarin,2003] E. Asarin, G. Schneider, and S. Yovine "*Towards computing phase portraits of polygonal differential inclusions*" , Technical Report 2003-042, Department of Information Technology, Uppsala University, August 2003

[Asarin, 2007] Eugene Asarin, Gerardo Schneiderb, Sergio Yovinec "*Algorithmic analysis of polygonal hybrid systems, part I: Reachability"*, Theoretical Computer Science 379, pp 231–265, (2007)

[Asarin, 2008] Eugene Asarin,Gordon Pace, Gerardo Schneiderb, Sergio Yovinec "*Algorithmic analysis of polygonal hybrid systems, part II: Phase portrat and tools"*, Theoretical Computer Science 390, pp 1–26, (2008)

[Aubin, 1984] J.P Aubin, A. Cellina, . " *Differential Inclusions, Set-Valued Maps And Viability Theory"*. Grundl.der Math. Wiss. 264. Berlin: Springer, 1984

[Aubin, 1990] J.-P. Aubin , H. Frankowska. "Set Valued Analysis". Birkh¨auser, Boston, 1990.

[Aubin, 1991] J.P.Aubin."Viability Theory". Birkh¨auser, Boston, 1991.

[Aubin,1997] J P. AUBIN, *Dynamic Economic Theory : a Viability Approach*, Springer-Verlag,1997.

[Aubin,1998] J P. AUBIN, L. NAJMAN , « The Russian Mountain Algorithm for Global Optimisation», *J. Mathematical methods of Operations Research*, , 1998.

[Aubin,1999] J P AUBIN, *Mutational and Morphological Analysis : Tools for Shape Regulationand Morphogenesis*, Birkhäuser, 1999.

[Aubin, 2001] J. P Aubin, J. Lygeros, M. Quincampoix, S. Sastry S. & N. Seube "*Viability and invariance kernels for impulse differential inclusions*" Proceedings of the 40th IEEE conference on decision and control, 2001

[Aubin, 2001 a] J P. Aubin, G. Haddad *"Detectability through Measurements under Impulse Differential Inclusions"* Cahier de la MES vol. a.2001, n.30, 2001 .

[Aubin, 2002] J. P Aubin., J. Lygeros ,M. Quincampoix, S. Sastry S. & N. Seube « Impulse Differential Inclusions : A Viability Approach to Hybrid Systems", IEEE Transactions on Automatic Control, 47, 2-20,2002

[Aubin, 2006] J.P Aubin, "*Boundary-Value Problems for Systems of Hamilton-Jacobi-Bellman Inclusions with Constraints*" SIAM Journal on Control and Optimization , SIAM J. Control Optim., 41(2), 425–456. (32 pages), 2006

[Aubin, 2011] J.P Aubin, A. Bayen, P. Saint-Pierre , "Viability Theory. New Directions » Springer , 2011

[Aubin, 2013] J.P Aubin, A. Bayen, P. Saint-Pierre , "Viability Theory. New Directions , second edition » Springer , 2013

[Aublin, 1992] Aublin M., Boncompain R., Boulaton M., Caron D., 1992. *Systèmes mécaniques, théorie etdimensionnement*. Dunod. Paris

[Aubry et al., 2011] Aubry, C., Desmare, R., and Jaulin, L. (2011). "*Loop detection in a mobile robot trajectory using interval analysis"*. In SWIM 2011, Bourges, France.

[Baniardalani, 2012] Sobhi Baniardalani, Javad Askari "A Geometric Method for Generating Discrete Trace Transition System of a Polyhedral Invariant Hybrid Automaton", *Intelligent Control and Automation*, vol 3, pp: 197-206 **,**2012

[Bardi, and all, 1997] M. Bardi and I. Capuzzo-Dolcetta. Optimal Control and Viscosity Solutions of Hamilton- Jacobi-Bellman Equations, volume 17 of Systems & Control: Foundations & Applications. Birkauser, 1997

[Barrios, 2012], Jorge Barrios, Alain Piétrus, Gonzalo Joya, Aymée Marrero, Héctor de Arazoza"A differential inclusion approach for modeling and analysis of dynamical systems under uncertainty", Soft Comput, DOI 10.1007/s00500-012-0889-2, Springer-Verlag , 2012

[Bellman,1957] R. Bellman. Dynamic Programming.Princeton University Press, 1957.

[Bemporad, 2000]Alberto Bemporad, Giancarlo Ferrari-Trecate, Manfred Morari "Observability and Controllability of Piecewise Affine and Hybrid Systems", IEEE Transaction on Automatic Control, vol. 45, no. 10, october 2000

[Ben Abdessalem, 2011] Ben Abdessalem M A; (2011), « *Optimisation avec prise en compte des incertitudes dans la mise en forme par hydroformage* ». thèse, école doctorale SPMII, Rouen ,France

[Bergounioux, 2001] Bergounioux M.,2001*Optimisation et introductionau contrôle des systèmes linéaires*

[Bertsekas,1984] D.P. Bertsekas. Dynamic Programming and Optimal Control, volume 1 & 2. Athena Scientific, 1984.

[Bernard, 2011], Bernard.C « La théorie de la viabilité au service de la modélisation mathématique du développement durable. Application au cas de la forêt humide de Madagascar », L'Ecole doctorale des sciences fondamentales (Clermont-Ferrand), thèse de doctorat, 2011

[Bonneuil, 1997] BONNEUIL N., MULLERS K., « Viable Populations in a predator-prey system », *J. Mathematical Biology*, vol. 35, p. 261-293.1997

[Bouskela , and al 2011] Bouskela D, Jardin A, Benjelloun-Touimi Z, Aronsson P, and Fritzson P. (2011) « *Modelling of uncertainties with Modelica* ». proceedings of the 8th Modelica conference, Drsden, Germany,

[Brun-Picard, 2004] Brun-Picard,D, Consortium FOR2M (2004) « La mécatronique en tant que demarche de conception intégrée de systèmes à hautes performances dynamique ».COMEFIM conférence Bucarest.

[Cardaliaguet, 1998] Cardaliaguet P., Quincampix M., Saint-PierreP., « Set-Valued NumericalAnalysis for Optimal Control and Differential Games », *Stochastic and DifferentialGames*, Birkhäuser, 1998.

[Chabert, 2007] Chabert, G. (2007), « *Techniques d'intervalles pour la résolution de systèmes d'équations* ». PhD dissertation, Université de Nice-Sophia Antipolis, Nice-Sophia Antipolis, France. Available at : www.emn.fr/z-info/gchabe08/.

[Chabert and Jaulin, 2009] Chabert, G. and Jaulin, L. (2009). "*Contractor Programming. Artificial Intelligence* », 173 :1079.1100. WOS.

[Chutinan, 2003] Alongkrit Chutinan , Bruce H. Krogh, Fellow, IEEE, "*Computational techniques for hybrid system verification"* IEEE Transaction on automatic control, vol. 48, no. 1, 2003

[Colin, 2008] « Chaotic dynamics in hybrid systems », Nonlinear Dynamics and Systems Theory, 8 (2) , 169–194, 2008

[Craig, 1999] Craig Kevin « *Mechatronics at Rensselaer : a two-course senior elective sequence in Mechanical engineering»* Proceeding of the 1999 IEEE/ASME International conference on advanced intelligent Mechatronics, 452, 458

[Delanoue et al., 2006] Delanoue, N., Jaulin, L., and Cottenceau, B. (2006).“*Using interval arithmetic to prove that a set is path-connected.*Theoretical Computer Science, Special issue : Real Numbers and Computers,” 351(1) :119.128.

[Devictor, 2004] N. DEVICTOR “Advances in methods for uncertainity and sensitivity analysis” Proceedings of the workshop “Level 2 PSA and Severe Accident Management”, OCDE/AEN/CSNI/WGRISK, Köln, Mars 2004.

[Dontchev, 1989] Dontchev A, Farkhi EM “*Error estimates for discretized differential inclusions*”. Computing 41:349–358, 1989

[Dontchev, 1994] Dontchev A, Hager W “*Euler approximation to the feasible set*”. Numer Funct Anal Optim 15:245–262, 1994

[Donchev, 2006] Tzanko Donchev« *Impulsive differential inclusion with constraints* », Electronic Journal of Differential Equations, Vol. 2006, No. 66, pp. 1–12. (2006)

[Donchev, 2007] “*Fixed time impulsive differential inclusions*”, Surveys in Mathematics and its Applications, Volume 2,pp: 1 – 9, 2007

[Doyen, 1996] Doyen L., Gabay D., « Economie des ressources renouvellables et viabilité », *Actes des journées Vie, Environnement et Sociétés*, 1996.

[Doyen 97] Doyen L., Gabay D., « Viabilité et Régulation d’un modèle de croissance prenant en compte le risque climatique », *Journées du PIRPVS, Toulouse*, 1997.

[Drocourt et al., 2005] Drocourt, C., Delahoche, L., E. Brassart, B. M., and Clerentin, A. (2005) “*Incremental construction of the robots environmental map using interval analysis*”.Global Optimization and Constraint Satisfaction : Second International Workshop, COCOS 2003, 3478 :127.141.

[Duc, 1999] Duc .G, Font, S (1999) « *commande H∞ et µ-analyse des outils pour la robustesse* » Hermes Science publiacation , Paris.

[Ellouze, 2010] Imen Ellouze« *Etude de la stabilité et de la stabilisation des systèmes à retard et des systèmes impulsifs* », thèse, l’Université Paul Verlaine de Metz, 2010

[Es Sadek, 2009] M.Z Es Sadek *« Contribution à l'optimisation globale. Approche déterministe , stochastique et application »,* INSA de Rouen, 2009

[Fritzon, 1998] P.Fritzon, V. Engelson, J. Gunnarsson *'An Integrated Modelica Envirement for Modeling, Documentation and Simulation'* The 1998 Summer Computer Simulation Conference (SCSC '98) ; 1998

[Fritzon, 2003] P. Fritzon., *"Principles of Object-Oriented Modeling and Simulation with Modelica 2.1"*,IEEE press,Wiley-interscience, 2003.

[Fritzon, 2007]P. Fritzon, al, "*OpenModelica System Documentation, Preliminary Draft, 2007-06-20 for OpenModelica 1.4.3"*, june 2007.

[Global methodology of an uncertainty study (Site web openturns)] site web : http://doc.openturns.org/openturns-latest/html/ReferenceGuide/cid2.xhtml

[Gordon, 2006] Gordon J. Pace, Gerardo Schneider , "*Static analysis of SPDIs for state-space reduction*", Technical Report 336, Department of Informatics, University of Oslo, PO Box 1080 Blindern, N-0316 Oslo, Norway, April 2006

[Gorre, 1996] Gorre A., « The "Montagnes Russes" Algorithm in mutational spaces», *Parametric Optimization IV*, , 1996, Springer-Verlag.

[Gratton , 2002] Gratton Y., « *Krigeage : la méthode optimale d'interpolation spatiale* » (2002) les articles de l'Institut d'Analyse Géographique, ,www.iag.asso.

[Hackl, 1993] Hackl G "*Numerical approximation of reachable sets and control sets*". Rand Comput Dyn 1(4):371–394, 1993

[Hadj Amor, 2008] H J. Hadj-Amor « Contribution au prototypage virtuel de systèmesmécatroniques basé sur une architecture distribuée HLA. Expérimentation sous les environnements OpenModelica-OpenMASK », Thèse , Supméca Toulon, 2008

[Hadj Amor, 2012] H J. Hadj-Amor, T. Soriano "A contribution for virtual prototyping virtuel of mechatronic systems based on HLA real time distributed high level architecture" *Journal of Computing and Information Science in Engineering JCISE* , vol. 12, no. 1, ed. ASME American Society of Mechanical Engineer, 2012

[Harashima, 1996]F. Harashima, M. Tomizuka, T. Fukuda, "*Mechatronics – What is it, why and how? Aneditorial*", IEEE/ASME Transactions on Mechatronics 1, 1-4, 1996.

[Henzinger, and al 1995] T.A. Henzinger, P.W. Kopke, A. Puri, P. Varaiya, (1995) "*What's decidable about hybrid automata ?*",27th Annual ACM Symp. On Theory and Computing (STOCS),

[Henzinger, 1997] T.A. Henzinger, P.-H.Ho, and H. Wong-Toi. *"HyTech: A Model Checker for Hybrid Systems."*, Lecture Notes In Computer Science; Vol. 1254, Proceedings of the 9th International Conference on Computer Aided Verification, pp 460 – 463, 1997, ISBN:3-540-63166-6.

[Henzinger, 1997 a] Thomas A. Henzinger, Pei-Hsin Ho, Howard Wong-Toi "HyTech: A Model Checker for Hybrid Systems" software Tools Technol. Transfer, vol 1, numéro 1/2 , pp 110-122, 1997

[Hien, 2001] N. V. Hien*," Une Méthode Industrielle de Conception de Commande par Automate Hybride Développée en Objets"*, Thèse, AIX-MARSEILLE III, 18 DEC 2001.

[Hospital, 2012] F. Hospital '*Conception préliminaire des actionneurs électromécaniques basée sur les modèles : lois d'estimations et règles de conception pour la transmission de puissance mécanique'* Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse), 2012.

[Ibn Taarit, 2010] Kaouther Ibn Taarit, « *Contribution à l'identification des systèmes à retards et d'une classe de systèmes hybrides* »,Thèse , LAGIS de l'Ecole Centrale de Lille , LACS de l'Ecole Nationale d'Ingénieurs de Tunis, 2010

[Isermann, 2000] R. Isermann, *« Mechatronic systems : concepts and Applications »*, Transactions of the Institute ofMeasurement and control 22,1 (2000), pp.29-55.

[Jabeen, and all, 2010] Jabeen S.D, Bhunia A.K , 2010, '*Population based steepest descent method'*AMO advanced modeling and optimization.

[Jantschi, 2009] L. Jantschi, S. D. Bolboaca « *Distribution Fitting 2. Pearson-Fisher, Kolmogorov-Smirnov, Anderson-Darling, Wilks Shapiro, Cramer-von-Misses and Jarque-Bera statistics », 2009*

[Jaulin L, et al, 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001) ."*Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics"*. Springer-Verlag, London,

[Jaulin and Chabert, 2008] Jaulin, L. and Chabert, G. (2008).QUIMPER : « *un langage de programmation pour le calcul ensembliste »* ; Application à l.automatique. In CIFA 2008, Bucharest, Romania.

[Juarez and all, 2008] Z . JUÁREZ, B. DENIS, J-J. LESAGE « Réseaux d'automates hybrides à synchronisations »typées pour la modélisation des Systèmes dynamiques hybrides » "Conférence Internationale Francophone d'Automatique, CIFA 2008, Bucarest : Roumanie (2008)"

[Kirkpatrick,1983] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. "*Optimization by Simulated Annealing*". Science, May 1983

[Koutsoukos, 2008] Xenofon D. Koutsoukos, *Senior Member, IEEE*, and Derek Riley"Computational methods for verification of stochastic hybrid systems",IEEE Transaction on systems, man, and cybernetics- Part A: system and human, vol. 38, no. 2, march 2008

[Kryszewski, 2006] Wojciech Kryszewski, Sławomir Plaskacz "Periodic solutions to impulsive differential inclusions with constraints", Nonlinear Analysis Vol, 65, pp: 1794–1804, 2006

[Kucharavy, 2007] Kucharavy,D (2007) « *TRIZ instruments for forcasting : past, present and future»;* LGECO - design engineering laboratory INSA Starsbourg Graduate school of science and technology.

[Kurzhanski, 1992] Kurzhanski A, Valyi I "*Ellipsoidal techniques for dynamic systems: control synthesis for uncertain systems*." Dyn Control 2:87–111, 1992

[Laroche, 2007] Laroche, E (2007) « *Identification et commande robuste de systèmes électromécaniques* »HDR , université Louis Pasteur de Stasbourg, France.

[Le Bars, 2011] Le Bars, F, (2011), « *Analyse par intervalles pour la localisation et la cartographie simultanées; Application à la robotique sous-marine* », université de Bretagne occidentale, France, thèse.

[Linde et al, 2006] Linde H, Herr,G Rehklau,A, (2006), « *INNOVATION of the INTEGRATED PRODUCT and PROCESS DEVELOPMENT by WOIS* », TRIZ conference 2006 Japan.

[Liu, 2008] Bin Liu, David J. Hill "*Stability of Discrete Impulsive Hybrid Systems via Comparison Principle*", Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, July 6-11, 2008

[Lydoire and Poignet, 2003] Lydoire, F. and Poignet, P. (2003) "*Nonlinear predictive control using constraint satisfaction*". In 2nd International Workshop on Global Constrained Optimization and Constraint Satisfaction (COCOS), pages 179.188. 2003

[Lygeros, 2003] John Lygeros, Karl Henrik Johansson, Slobodan N. Simié, Jun Zhang, S.Shankar Sastry "Dynamilcal properties of hybrid automata", IEEE transaction on automatic control, Vol 48, No1, 2003

[Lygeros, 2004] J. Lygeros, "Lecture Notes on Hybrid Systems", *Notes for an ENSIETA short course*, 54-58, 2004

[Lygeros, 2010] J. Lygeros, M. Prandini « Stochastic Hybrid Systems: A Powerful Framework for Complex, Large Scale Applications » *European Journal of Control*, 2010 pp 583-594.

[Lynch and all, 1996] N. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III*. New York: Springer-Verlag, 1996, number 1066 in LNCS, pp. 496–510.

[Mattiolli, 96] Mattiolli J., Doyen L., NajmanL., « Lattice operators underlying dynamic systems », *Set-Valued Analysis*, vol. 4, p. 119-134.1996

[Mattioli, 2003], Mattiolli, J et Artiouchine, K, « Noyau de viabilité : une contrainte globalepour la modélisation de systèmes dynamiques »*JFPLC'2003 Douzièmes Journées Francophones de Programmation Logique et Programmation par Contraintes,* Amiens 2003

[Meizel et al., 1996] Meizel, D., Preciado-Ruiz, A., and Halbwachs, E. (1996). "*Estimation of mobile robot localization : geometric approaches*". In Milanese, M., Norton, J., Piet-Lahanier, H., and Walter, E., editors, Bounding Approaches to System Identification, pages 463.489. Plenum Press, New York, NY.

[Meza, 2010] Juan C.Meza, 2010 '*steepest descent'*

[MODRIO 2013] Eurêka ITEA2 MODRIO deliverable D 6.2.7 « analysis of complex systems with varying parameters » Supmeca-IFPEN 2013 .

[Morton, 1985] Monrton, B. G "*new application of μ to real parameter variation problem* " IEEE conference on decision and control, volume 24, page: 233- 238, 1985

[Pace, 2004] Gordon J. Pace, Gerardo Schneider"*Model checking polygonal differential inclusions using invariance kernels"* VMCAI' 04, numero 2937 in LNCS, 2004

[Pontryagin,1962] L. Pontriaguine, V. Boltiansky, R. Gamkrelidze, and E. Michtchenko. The mathematical theory of optimal processes. Editions de Moscou, 1962.

[Quincampoix, 19 91] Quincampoix M., « Problèmes de cibles en théorie du contrôle et des jeux différentiels », PhD thesis, Paris Dauphine, 1991.

[Rabier, 2007] Rabier, F (2007). « *Modélisation par la méthode des plans d'expériences du comportement dynamique d'un module IGBT utilisé en traction ferroviaire* » , Thèse, ENI de Tarbes.

[Raczynski, 2004] Raczynski, S. (2004) 'Continuous simulation, differential inclusions, uncertainty, and traveling in time', *Simulation*, Vol. 80, No. 2, pp.87–100

[Raczynski , 2006] Raczynski S., « *Modeling and simulation: The Computer Science of Illusion"*, Wiley, England, p115-116.2006

[Raczynski, 2007] Raczynski S., (2007)*'Determination of attainable regions in ship maneuvers : an application of differential inclusions to the modeling of ship dynamics'*.Journal of Marine Science and Technology.

[Raczynski, 2011] Raczynski, S (2011) 'uncertainty, dualism and inverse reachable sets' International journal of simulation and modelling vol. 10, no1, pp. 38-45 .

[Raissi et al., 2004] Raissi, T., Ramdani, N., and Candau, Y. (2004) "*Set membership state and parameter estimation for systems described by nonlinear differential equations*." Automatica, 40 :1771.1777.

[Rubino, 2007] G . Rubino, B. Tuffin '*Simulations et méthodes de Monte Carlo*' techniques de l'ingénieur, Référence AF600, 10 oct. 2007

[Rüdiger, 1998] Rüdiger H, Harald M, Willy S.,(1998). "An approach to a general view on tolerances in mechanical engineering". 2nd International Workshop on Integrated Product Development IPD 98. Magdeburg. 65 - 76

[Saint-Pierre, 1994] Saint-Pierre P., « Approximation of Viability Kernel », *Applied Mathematics and Optimisation*, vol. 29, 1994, p. 187-209.

[Saint-Pierre, 1998] Saint-Pierre P., « Dynamical System with State Constraints, Theory and Applications », *IFIP Conference on Systems Modelling and Optimization*, 1998.

[Samper,2007] , Samper, S (2007) « Tolérancement et analyse des structures au services des sytèmes souples et du défaut de formes » habilitation à diriger les recherches , université de Savoie.

[Schimmerling et al , 1998] Schimmerling P, Sisson J.C, Zaidi A (1998) « pratique des plans d'expérience » techniques & documentation.

[Schneider, 2002] Gerardo Schneider, « *Analyse Algorithmique de Systèmes Hybrides polygonaux* », Université de Grenoble I_ Joseph Fourier, Sciences et Géographie VERIMAG, 2002.

[Schneider, 2002 a] Gerardo Schneider, Eugene Asarin, Godon Pace et Sergio Yovine, "SPeeDi a verification tool for polygonal hybrid system" , Proceedings of the 14th International Conference on Computer Aided Verification page, 354-358, Springer- Verlag London UK , 2002

[Schneider, 2003], *Gerardo Schneider* "Invariance Kernels of Polygonal Differential Inclusions" , Technical report 2003-042, ISSN 1404-, Department of Information Technology Uppsala University Box 337, SE-751 05 Uppsala, Sweden, 2003

[Site Web Mathematica] http://reference.wolfram.com/legacy/v2/contents/whatis.pdf

[Site Web Mathmodelica]http://www.mathcore.com/products/mathmodelica/

[Site Web Modelica] https://www.modelica.org/

[Site web matlab] http://w3.gel.ulaval.ca/~lehuy/intromatlab/doc_1.htm

[Tian, 2000] Tian M.F., Shi R., 2000. Worst case tolerance analysis of linear Analog Circuits using sensitivity*bands* IEEE transactions on circuits and systems vol47, N°8

[Trélat , 2005] Trélat. E (2005), « *Contrôle optimal : théorie et applications* ». Collection Mathématiques Concrètes. Vuibert,

[Turki, 2008] Turki S., *"Ingénierie système guidée par les modèles: Application du standard IEEE15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques",* Thèsesoutenue le 02/10/2008, Université du Sud ToulonVar.

[Veliov, 1997] Veliov V , "On the time-discretization of control systems". SIAM J Control Optim 35(5):1470–1486. ISSN:0363-0129, 1997

[Vinas et al., 2006] Vinas, P. H., Sainz, M. A., Vehi, J., and Jaulin, L. (2006)."*Quantified set inversion algorithm with applications to control"*. Reliable computing, 11(5) :369.382.

[Zaytoon, 2001] (sous la direction de) J. Zaytoon, « Systèmes dynamiques hybrides », HERMES science publication , 157-161, 2001.

Publications

Manel. Zerelli, Thierry. Soriano, Baptiste. Arnault, « Modeling of parameters' variation using differential inclusions", *IEEE Mechatronics French-Japanese conference*, Yokomaha Japan, 2010

Manel. Zerelli, Thierry. Soriano, Hassen Jawhar. Hadj Amor, « Modélisation et analyse de variations paramétriques d'un système mécatronique par l'inclusion différentielle », *CFM congrès français de mécanique,* Besançon France, 2011

Manel. Zerelli, Thierry. Soriano, "Identification of the Impulse Differential Inclusion for the Behavior of a Mechatronic System", IFAC, Mathmod, *7th Vienna International Conference on Mathematical Modelling,* 2012, in press

Manel Zerelli, Thierry Soriano, "Application of Impulse Differential Inclusion for Uncertainty Analysis of Mechatronic Hybrid System", IEEE conference Mecatronics REM 2012.

Manel Zerelli, Thierry Soriano, « An approch of tolerancing of a mechatronic system in state space. », *Journal of « Advances in Applied Mathematics and Mechanics », 2013 ,* sous reviewing

Manel Zerelli, Thierry Soriano, "A parameter variation analysis algorithm for hybrid systems based on impulse differential inclusion", journal of "Computational optimization and application." 2013, sous reviewing

Résumé

Dans cette thèse nous avons proposé une méthode d'étude des variations paramétriques pour les systèmes mécatroniques continus et hybrides puis une approche du tolérancement mécatronique.
Nous avons d'abord étudié les différentes approches existantes pour la prise en compte de la variation de paramètres. Pour les systèmes continus à paramètres variables nous avons choisi la méthode des inclusions différentielles. Nous avons repris l'algorithme de Raczynski et nous avons développé un algorithme d'optimisation qui se base sur la méthode du *steepest descent*, avec une extension permettant d'obtenir l'optimum global. Pour les systèmes hybrides, contenant des évolutions continues et des sauts discrets, et qui présentent des variations paramétriques, nous avons choisi le formalisme de l'inclusion différentielle impulsionnelle comme outil de modélisation. Nous avons repris ce formalisme et identifié ses éléments sur un système mécatronique. Nous avons développé des algorithmes de résolution des inclusions différentielles impulsionnelles pour un puis pour plusieurs paramètres variables. Pour visualiser les résultats, les algorithmes développés ont été implémentés sous Mathématica. Nous avons fini cette partie par une comparaison entre notre approche et d'autres comme celles autour des automates hybrides à invariant polyèdre, les inclusions différentielles polygonales et l'algorithme pratique de résolution des inclusions différentielles. Nous avons montré alors certains avantages de notre approche.

En dernière partie, nous avons repris les différents outils utilisés et résultats obtenus pour définir et affiner notre approche du tolérancement. Nous avons défini la zone du fonctionnement désiré, les différents cas de figures qu'elle peut présenter et son intersection avec le domaine atteignable.
Nous avons présenté un outil métrique basé sur la distance topologique de Hausdorff pour le calcul des distances entre ces différents ensembles. Munis de ces éléments, nous avons proposé une démarche itérative pour le tolérancement dans l'espace d'état.

Mots clés : système hybride, mécatronique, paramètre variable, inclusion différentielle impulsionnelle, tolérancement, distance de Haussdorf.