REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEINEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE A. MIRA-BEJAIA

Mémoire de fin d'études

En vue de l'obtention du diplôme Master

Département : Génie électrique

Filière : Electrotechnique

Option : Automatisme Industriel

Thème

Utilisation de Stateflow (Matlab) pour commander un système à événement discret

Encadré par:

Mr. AZZI Abdelmalek

Préparé par :

Mr : MAMAN ATTAHER Moussa

Examinateurs:

Mr. MELAHI Ahmed

Mr. LARBA Mohamed

Promotion 2023/2024

Remerciement

Je tiens à exprimer mes remerciements et ma gratitude à monsieur AZZI Abdelmalek, professeur à l'université A/Mira Bejaia, pour tous ses efforts, ses conseils et pour la modestie qui le caractérise malgré le grand savoir scientifique qu'il possède, choses qui m'ont permis de mener à bien cette étude.

J'adresse mes chaleureux remerciements aux membres du jury qui ont accepté d'évaluer ce travail.

Mes vifs remerciements vont également à l'ensemble des enseignants de département génie électrique de l'université A/Mira Bejaia pour la formation qu'ils nous ont assuré tout long de notre cursus universitaire et tous ceux à qui ont participé à notre formation.

Pour terminer je tiens à remercier tous mes collègues et mes amis qui m'ont aidé et qui m'ont apporté leur soutien moral ainsi que tous ceux qui ont participé de près ou de loin à l'élaboration de ce travail.

Dédicace

Je dédie ce modeste travail à ma chère mère, à mon cher père, qui m'ont soutenu et encouragé durant toutes mes études et nulle chose ne récompensera leurs sacrifices. Que dieu les garde pour moi.

A tous les membres de ma famille (sur tout mon frère Abbas Maman Attaher) qui m'ont aidé de plusieurs manières et pour leur soutien précieux durant les longues années de ma formation, ce qui leur fait valoir ma grande gratitude. Qu'ils me pardonnent mon manque de disponibilité et mes absences. Que ce travail soit une part de ma reconnaissance envers eux.Je dédie aussi ce travail à mes amis algériens (Naim, Hicham, Ouallam, M6, etc.), à mes compatriotes nigériens (Sani, Douada, Ali, Rachelle, Hadiza, Tidjani, Djafar, Banani, Moustapha, Youssouf, etc.), et à mes amis internationaux (John, Iba, Keita, Awa, Noris, Edenaida, Anthony, etc.). La liste est loin d'être exhaustive, mais sachez que vous méritez mieux que cela.

Sommaire

Liste des figures

Liste des figures

Introduction générale

Introduction générale

Dans le domaine de l'ingénierie des systèmes de contrôle, la modélisation et la commande des systèmes à événements discrets (SED) sont essentielles pour garantir un fonctionnement fiable et efficace. Les SED sont des systèmes dynamiques où les événements se produisent à des instants distincts dans le temps, souvent en réponse à des conditions spécifiques ou à des stimuli externes. Pour concevoir et mettre en œuvre des stratégies de contrôle robustes pour de tels systèmes, les ingénieurs se tournent souvent vers des outils logiciels avancés tels que Stateflow, qui est une extension de MATLAB dédiée à la modélisation, la simulation et la génération de code pour les systèmes à événements discrets.

Stateflow offre une approche graphique et intuitive pour la modélisation du comportement dynamique des SED, en permettant aux ingénieurs de représenter les états, les transitions et les événements de manière visuelle. En utilisant des diagrammes d'états et de transitions, les concepteurs peuvent décrire le comportement discret du système et spécifier les actions à entreprendre en réponse à différents événements. De plus, Stateflow permet l'intégration transparente de la logique de contrôle avec d'autres composants du système, facilitant ainsi le développement de systèmes complexes.

Dans notre travaille, nous explorerons les principes fondamentaux de Stateflow MATLAB et son application dans la conception de contrôleurs pour les SED. Nous discuterons des concepts clés tels que les états, les transitions, les événements, les actions et les hiérarchies de diagrammes d'états, tout en illustrant leur utilisation à travers des exemples concrets. En outre, nous examinerons les techniques avancées telles que la vérification de propriétés, la génération de code automatique et l'intégration avec d'autres outils de conception, soulignant ainsi la puissance et la flexibilité de Stateflow dans la commande des systèmes à événements discrets.

Chapitre I

I.1. Présentation de Stateflow

Stateflow est un produit développé par MathWorks qui propose un langage graphique pour modéliser des diagrammes de transition d'état, des organigrammes, des tables de transition d'état et des tables de vérité.[1][2].

Il est utilisé pour décrire le comportement des algorithmes MATLAB et des modèles Simulink en réponse aux signaux d'entrée, aux événements et aux conditions temporelles. Stateflow permet la conception et le développement de contrôles de supervision, de planification de tâches, de gestion des pannes, de protocoles de communication, d'interfaces utilisateur et de systèmes hybrides.[1]

Le langage Stateflow comprend des éléments tels que des diagrammes de transition d'état, des organigrammes, des tables de transition d'état et des tables de vérité. Ces éléments peuvent être utilisés pour modéliser une logique de décision combinatoire et séquentielle, qui peut ensuite être simulée dans un modèle Simulink ou exécutée en tant qu'objet dans MATLAB.[1][2]

Stateflow propose également une animation graphique pour le débogage et l'analyse pendant l'exécution. Les contrôles au moment de l'édition et de l'exécution garantissent la cohérence et l'exhaustivité de la conception avant la mise en œuvre. [1]

Stateflow est particulièrement utile pour modéliser des systèmes dynamiques en tant que machines à états finis. Il fournit un moyen visuel et intuitif de concevoir et de simuler des systèmes complexes et peut être intégré à d'autres produits MathWorks à des fins de vérification, de validation et de test.[2]

Stateflow prend également en charge la génération de code pour les systèmes embarqués, avec la prise en charge des formats C, C++, VHDL, Verilog et structurés en texte pour les automates.[2]

Stateflow est une boîte à outils logicielle puissante développée par MathWorks, largement utilisée dans l'ingénierie des systèmes embarqués, le contrôle automatique et la robotique. Cette solution permet aux ingénieurs de modéliser et de simuler le comportement des systèmes dynamiques complexes à l'aide de diagrammes d'états et de transitions, ainsi que de diagrammes d'activité. En utilisant Stateflow, les professionnels peuvent concevoir et tester des systèmes réactifs basés sur des états de manière efficace, ce qui accélère le processus de développement et améliore la fiabilité des produits finaux.[3]

Les systèmes à événements discrets sont des systèmes dynamiques où les événements se produisent à des moments spécifiques et distincts dans le temps, entraînant des transitions entre différents états du système. Stateflow offre une approche visuelle et intuitive pour modéliser ces transitions d'état, en utilisant des diagrammes d'états et de transitions, ainsi que des diagrammes d'activité pour spécifier le comportement des états.[3]

Grâce à Stateflow, les ingénieurs peuvent décrire le comportement des systèmes de manière claire et précise, en modélisant les états du système, les événements déclencheurs et les actions associées à chaque transition. Cela permet une conception plus robuste et une validation plus efficace des systèmes avant leur implémentation dans le monde réel.[3]

En combinant la puissance de la modélisation graphique avec la capacité de simuler et de générer du code automatiquement, Stateflow accélère le processus de développement des systèmes à événements discrets, tout en garantissant une meilleure qualité et une meilleure fiabilité des produits finaux.[1]

I.2. Hiérarchie des objets dans Stateflow

Le sommet de la hiérarchie dans Stateflow est occupé par la machine de Statflow, qui englobe tous les autres éléments dans le modèle de Simulink. Les machines de Stateflow organisent les éléments dans une structure hiérarchique basée sur la composition, ce qui signifie que Stateflow peut inclure d'autres éléments (voir Figure 1). De même, les diagrammes peuvent contenir différents types d'éléments tels que des états, des boîtes, des fonctions, des données, des événements, des transitions et des jonctions. Dans la hiérarchie de Stateflow, les états peuvent eux-mêmes contenir tous ces éléments, y compris d'autres états. La structure hiérarchique des états peut être représentée à l'aide de super-états et de sous-états.[4]

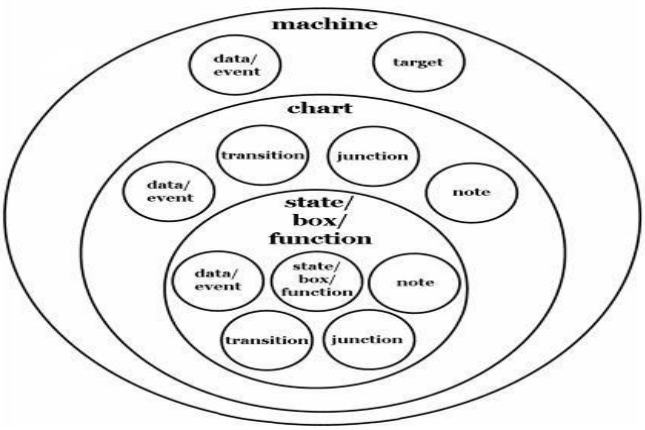


Figure I.1 : La hiérarchie dans Stateflow [4]

I.3. La construction de la machine Stateflow

Pour créer une machine à états finis, Stateflow propose un graphique ou chart que l'on peut intégrer dans une fenêtre Simulink et déplacer à volonté**.** .[4]

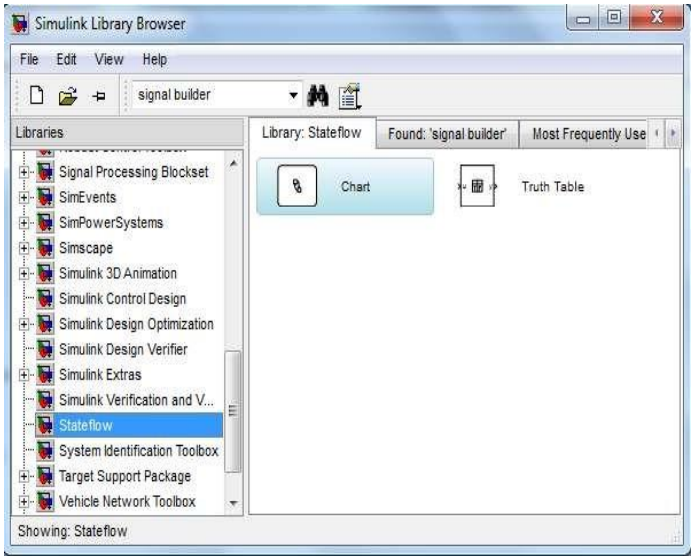


Figure I.2 : Bibliothèque Simulink [4]

D'outils En double-cliquant sur le bloc "Chart" dans la fenêtre MATLAB/Simulink, vous accédez à une fenêtre Stateflow où se trouve une palette indispensable pour construire une machine à états finis. [4]

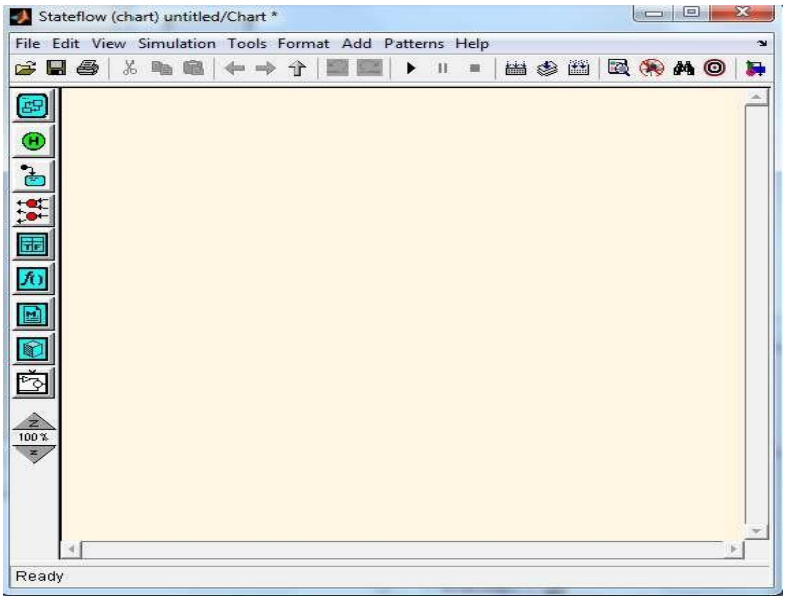
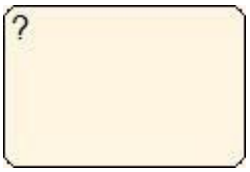
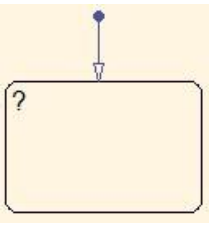
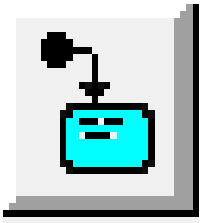
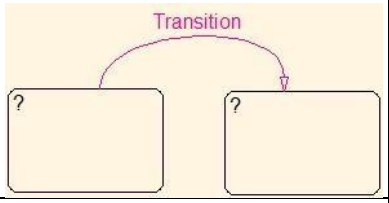
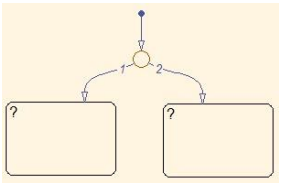
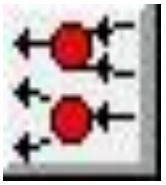
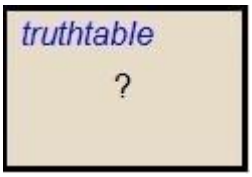
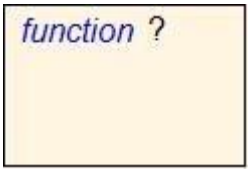


Figure I.3 : Environnement de Stateflow [4]

Nom	Notation et explication	Toolbox Icon
Etat	Rectangle qui définit l'état à qui l'on donne le nom a la position du signe (?) 	
Jonction de l'historique	Elle nous permet d'enregistrer l'activité d'un état.	
Transitions	Transitions de défaut. Objets graphiques qui spécifient quel état exclusif qui est en activité quand il y a deux états exclusifs ou plus au même niveau dans la hiérarchie 	

Jonction	Transition d'un état à un autre sur certaine condition que l'on l'écrit à la place de (Transition). 	
Jonction connective	Jonction ou arrivent et partent plusieurs transitions. Les transitions sont exécutées, soit par défaut selon leur création ou par la condition qui lui associe 	
Fonction de table de vérité		
Fonction graphique	Une fonction graphique est une fonction que vous définissez graphiquement avec un graphique d'écoulement qui inclut la langue d'action de Stateflow . 	
Fonction matlab		

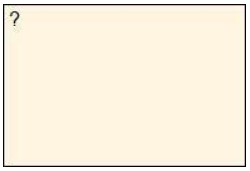
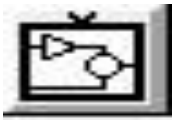
Box	Le Box ou la boite est un objet graphique qui organise d'autres objets dans le diagramme, tel que des fonctions et des états. 	
Fonction de simulink		

Tableau 1 : Présentation des différents outils de Stateflow [5]

I.4. Les Etats

Dans les états, les actions à réaliser sont décrites dans le cadre du graphe représentant la machine à états finis. Un état peut être actif ou inactif en fonction des événements survenant ou des conditions de validation des transitions.

Un état peut également être désigné comme un super-état, ou état parent, par rapport à son contenu, et ceux-ci deviennent des sous-états, ou enfants. Les états "Eteint" et "Allumé" sont des sous-états de l'état "Clignotant", lequel est à son tour un enfant de l'état racine. (Voir Figure 4). [5]

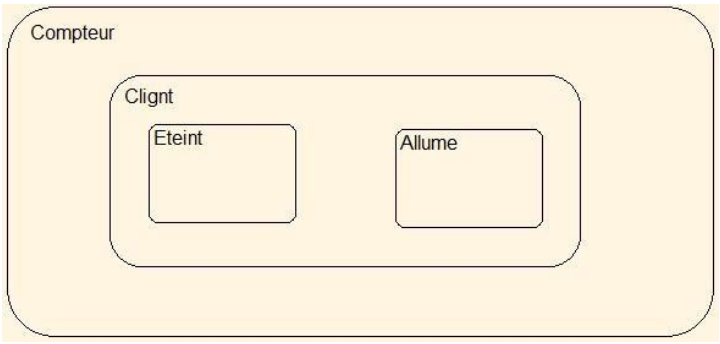


Figure I.4 : La hiérarchie des états ***[5]***

Cette structure hiérarchique est visible dans la fenêtre de gauche de l'explorateur (Outils... Explorer). Il est important de souligner que Stateflow est subordonné à Simulink.

Le bloc Stateflow (compteur) suit la même hiérarchie que les autres objets de Simulink. (Voir Figure 5). [5]

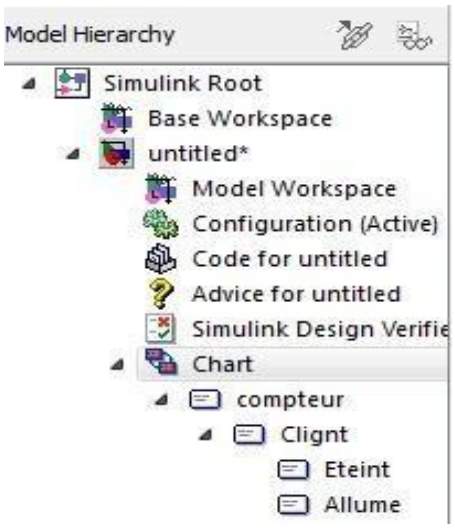


Figure I.5: La hiérarchie du modèle [5]

Le cas présenté dans la Figure 4 comprend des états exclusifs qui ne peuvent pas être actifs simultanément. Cette exclusivité est mise en œuvre via des transitions contenant des conditions permettant le passage d'un état à un autre. Ce type de décomposition est appelé "OU". D'autre part, un graphe Stateflow peut comporter plusieurs états actifs simultanément, définissant ainsi une décomposition "ET". Les états parallèles sont identifiés par des bordures en pointillés. Chaque état est accompagné d'un libellé situé en haut à gauche du rectangle le délimitant. En général, l'état contient :

nom/

entry : « actions à réaliser à l'entrée de cet état »

during : « actions à réaliser durant l'activité de cet état »

exit : « actions à réaliser dès qu'on sort de cet état »

on <nom_evt> : « actions à réaliser à l'occurrence de l'événement « nom_evt » »

bind : « variable » « actions ».

Si aucune spécification n'est fournie, cela est considéré comme une entrée par défaut, ce qui signifie que les actions sont exécutées dès que l'état est atteint.

L'indication "bin: a" implique que seule cet état ou ses états enfants peuvent modifier la valeur de la variable "a", tandis que les autres états peuvent utiliser cette variable mais ne peuvent pas modifier sa valeur.

Il est impératif que chaque état ait un nom unique à l'intérieur de son super-état pour éviter toute ambiguïté.

Les actions à exécuter dans Stateflow sont très similaires au langage "C" ou à Matlab :

Exemple :

- « a++ » : Incrémentation de la variable « a ».
- « abs(a) » : La valeur absolue de la variable « a ».
- « sqrt(a) » : La racine de la variable « a ».

De la même manière que pour MATLAB, une ligne peut être prolongée jusqu'à la suivante en utilisant le symbole... pour indiquer cette continuation.

- Exemple :

« Secondes = (secondes−60*minutes …

− 3600*heures) ; »

Les fonctions MATLAB et les variables de l'environnement de travail peuvent être utilisés en utilisant l'opérateur "ml". [4] [5]

I.5. Les transitions

Une transition est un élément graphique qui connecte un état à un autre. Un libellé décrit les circonstances ou les conditions du passage entre ces états. [5]

I.5.1.Les transitions par défaut

La transition par défaut, qui n'a ni condition ni libellé, spécifie l'état qui doit être actif en cas d'ambiguïté entre plusieurs états en décomposition "OU" ayant le même niveau hiérarchique. Chaque état contenant des sous-états doit avoir une transition par défaut sur l'un d'eux, déterminant ainsi l'état actif dès l'entrée de ce super-état (ou état parent). [5]

I.5.2.Labels des transitions

Une condition est une expression booléenne qui valide le passage d'un état à un autre, et elle est placée entre crochets.

Les conditions peuvent impliquer des variables locales ou des entrées de SIMULINK. Dans une transition, des actions peuvent être effectuées, similaires à celles réalisées à l'intérieur d'un état, et ces actions sont encadrées par des accolades.

Dans l'exemple suivant, la transition se produit lorsque la température dépasse le seuil de "120", et dans ce cas, le moteur de recirculation de l'air est arrêté en définissant la variable "recirc" à 0. [5]

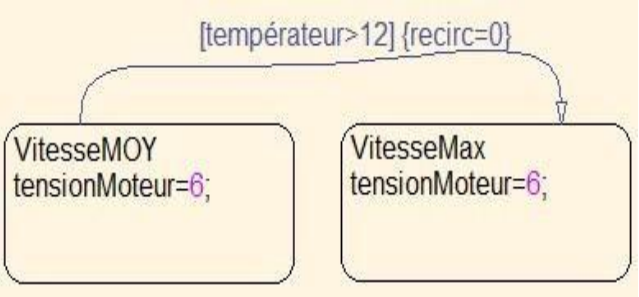


Figure I.6: Condition du passage d'un état à l'autre avec {condition-action} [5]

En général, le label d'une transition est formulé comme suit :

« Événement[condition]{action_de_condition} /action_de_transition »

La transition sera activée lorsque :

• *l'événement "événement" se produit.*

• *la condition est évaluée comme vraie.* [4]

I.6. Les événements

Les événements, qui ne sont pas des éléments graphiques, influent sur l'exécution du graphe Stateflow. Tous les événements doivent être définis en utilisant le menu "Ajouter => Événement" dans "Explorateur de modèles ». [4]

La survenue d'un événement peut être utilisée pour valider une transition d'un état à un autre ou pour déclencher une action spécifique à exécuter. [4]

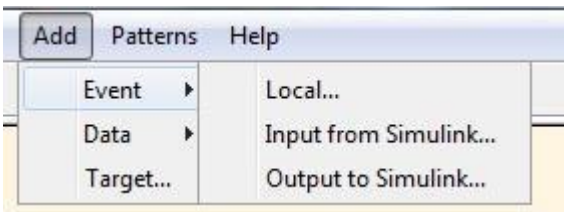


Figure I.7 : Comment ajouter les événements. [4]

I.7. Les Objets Datas

Les objets Data (données) peuvent représenter des variables locales, des entrées ou des sorties vers SIMULINK, des constantes, et ainsi de suite.

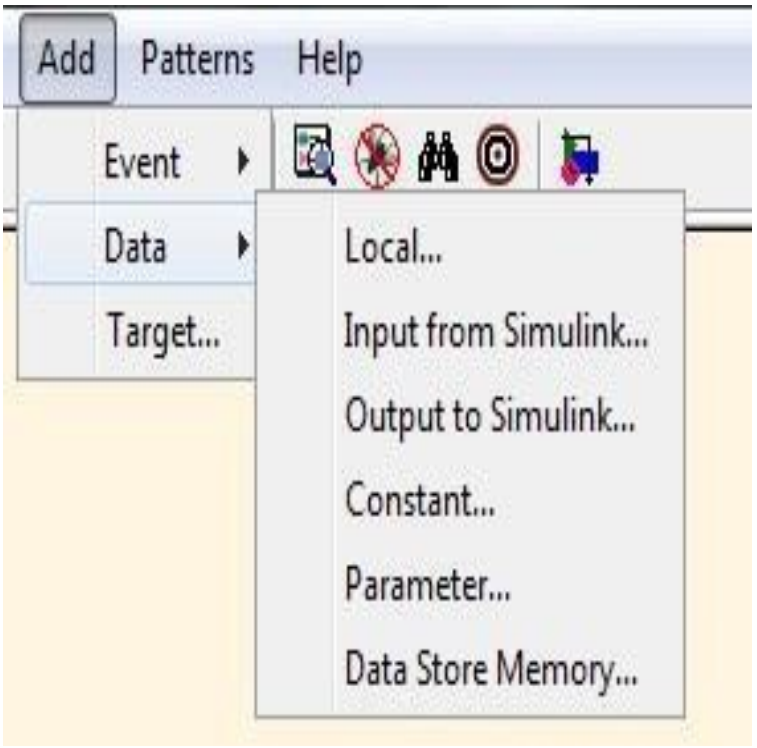


Figure I.8 : Comment ajouter les données (datas) [4]

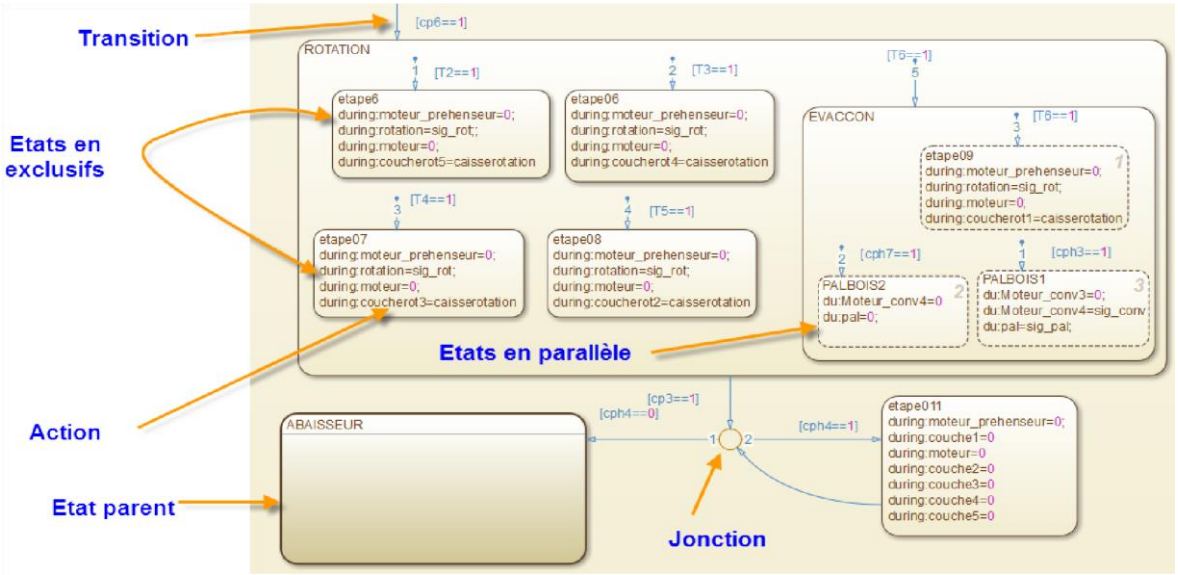


Figure I.9 : Exemple d'un diagramme de Stateflow [4]

I.8. Pourquoi opter une fonction graphique

Il est plus simple de créer, d'accéder et de gérer une fonction graphique qu'une fonction textuelle, comme celles que vous devez définir en langage C ou en MATLAB de manière externe. [5]

Une fonction graphique est une entité propre à Stateflow. Vous utilisez l'éditeur de modèles (patterns) pour créer une fonction graphique qui est intégrée au modèle, avec les diagrammes qui font appel à cette fonction.

I.9. Procédure pour créer Stateflow

Pour créer un diagramme de Stateflow, on présente le schéma fonctionnel qui est sur le schéma ci-dessous [5] :

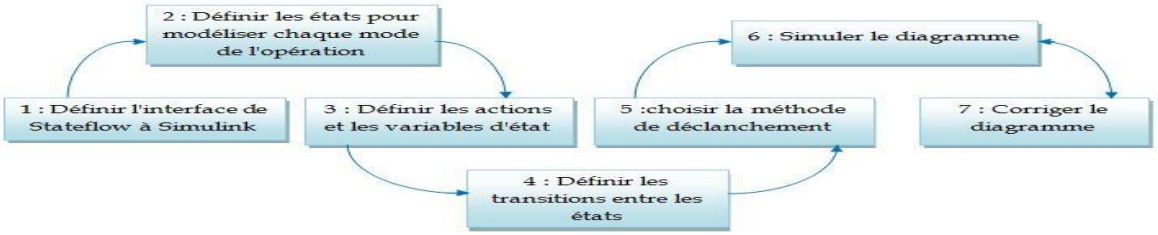


Figure I.10 : Schéma explicatif d'une création de diagramme Stateflow.

Il est plus simple de créer, d'accéder et de gérer une fonction graphique qu'une fonction textuelle, comme celles que vous devez définir en langage C ou en MATLAB de manière externe.

Une fonction graphique est une entité propre à Stateflow. Vous utilisez l'éditeur de modèles (patterns) pour créer une fonction graphique qui est intégrée au modèle, avec les diagrammes qui font appel à cette fonction.[5]

I.10. Conclusion

Les avantages de cet outil résident dans sa capacité à prendre en compte naturellement les aspects de parallélisme et de hiérarchie. Il permet également de réduire le nombre de transitions entre les différents états du système, notamment lorsqu'il s'agit d'une transition vers un état composé (état parent contenant des sous-états) ou lors d'une transition d'un état composé vers un autre. Nous explorerons cela en détail dans l'exemple d'application lors d'utilisation.

Chapitre II

II.1. Introduction

. La simulation d'un modèle dans Simulink génère des données de signal représentant un système dynamique. En intégrant ce modèle à un environnement virtuel, il devient possible d'utiliser ces données pour contrôler et animer ce monde virtuel. Créer un environnement virtuel implique la recherche ou la création d'objets en trois dimensions. Connecter cet environnement virtuel à MATLAB ou Simulink pour l'analyse de chaque étape présente ses propres défis. [3]

II.2. Description du déroulement du système

On a une commande à réaliser via stateflow de trois (3) moteurs qui ont une entrée respective Yref 1 ; Yref 2 et Yref3 et qui sont les sorties du chart matlab qui fera l'objet d'un programme sur Simulink. Le mot1 (moteur 1) doit s'allumer si 0<U<=3, mot 2 (moteur 2) s'allume si 3<U<=6 et mot 3 (moteur 3) s'allume lorsque U>6 avec U est l'entré de chart matlab.

U = |9*cos(pi*t) |

G(z) = $\frac{0.1}{z-0.6}$, chaque moteur a une fonction de transfert discret appelé G(z) du premier ordre. (voir figure II.1)

Yref 1 = (u>0 & u<1) *1+(u>=1 & u<2) *1.5+(u>=2 & u<3) *2

Yref 2 = (u>=3 & u<4) *1+(u>=4 & u<5) *1.5+(u>=5 & u<6) *2

Yref 3 = (u>=6 & u<7) *1+(u>=7 & u<8) *1.5+(u>=8 & u<9) *2

Y1, Y2 et Y3 sont les sorties des trois moteurs.

Les matériels utiliser sont :

Les matériels utiliser sont :

- ✓ Un chart Matlab
- ✓ Trois régulateurs de PI
- ✓ Trois sommateurs
- ✓ Trois fonctions de transfert discret G(z)
- ✓ Quatre scoops
- ✓ Quatre muxes

NB : Si le moteur 1 est activé, il donnera un échelon e1= 1 ; e2= 0 et e3= 0.

Si le moteur 2 est activé, il donnera un échelon e1= 1 ; e2= 1 et e3= 0.

Si le moteur 2 est activé, il donnera un échelon e1= 1 ; e2= 1 et e3= 1.

Les échelons e1 ; e2 ; et e3 sont aussi les sorties au niveau de chart Matlab.

II.3. Programme de Simulink

Dans Simulink, nous allons créer un nouveau fichier pour réaliser la simulation, comme illustré dans la figure ci-dessous :

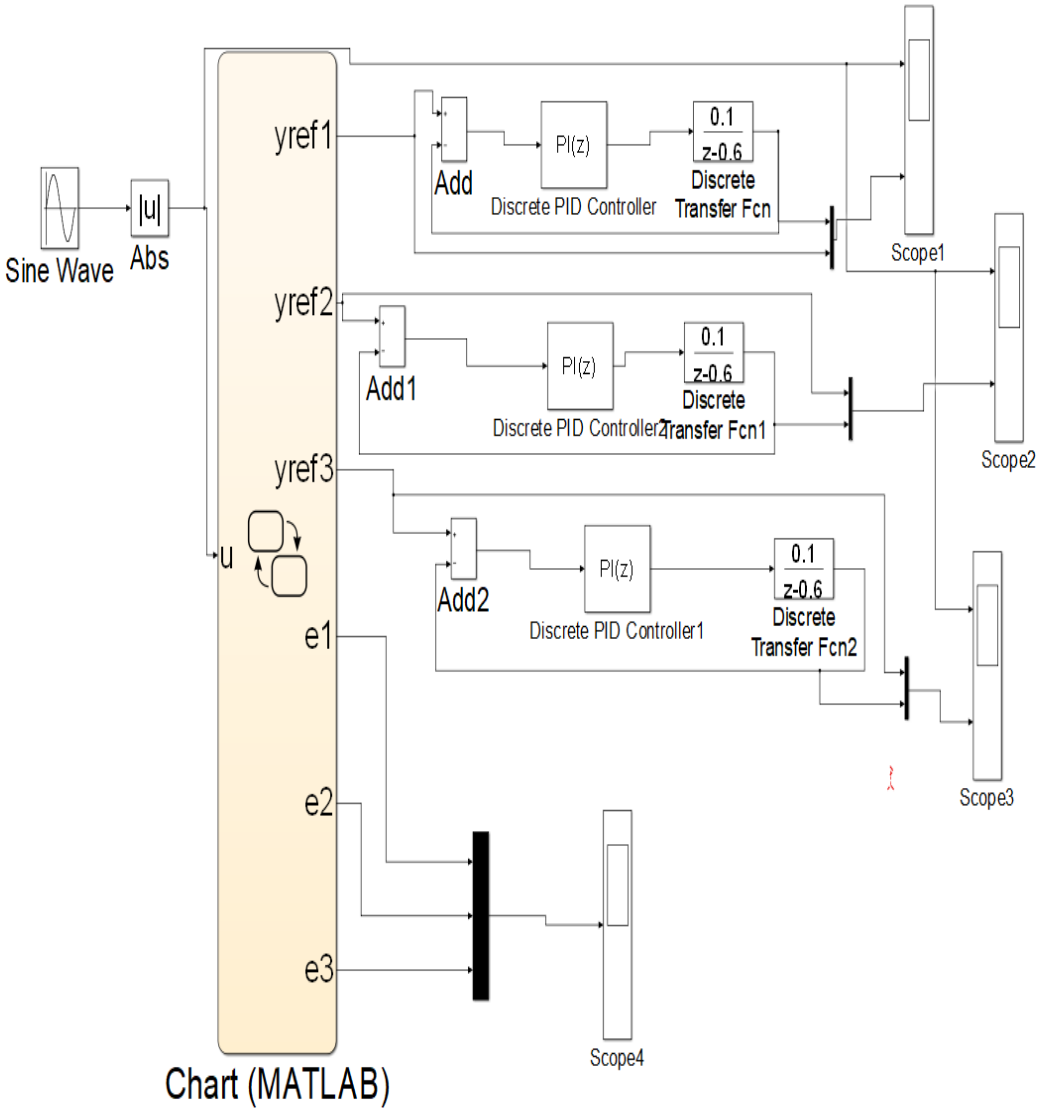


Figure II.1 : *Programme sur Simulink*

II.4. Programme de Stateflow

Ici, nous allons programmer dans l'environnement Stateflow de MATLAB, comme illustré dans la figure suivante :

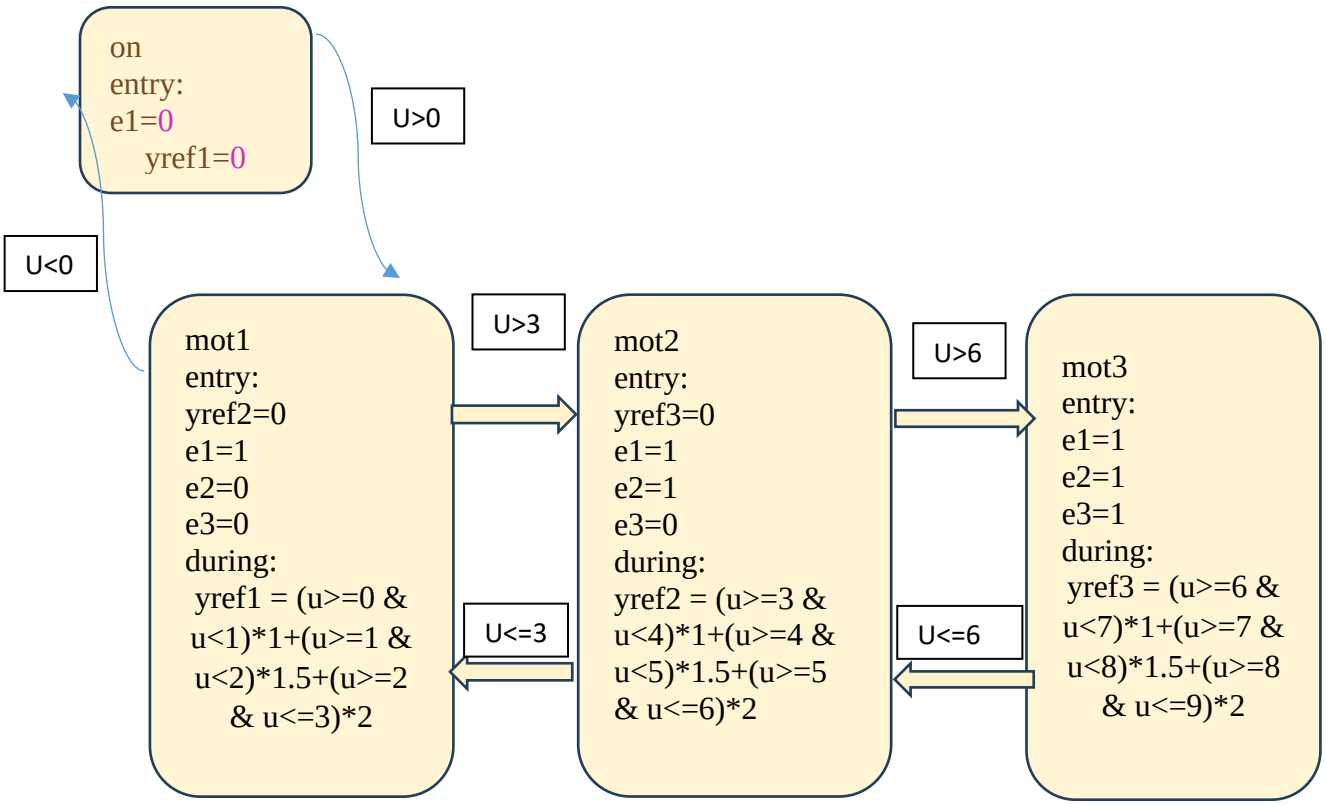


Figure II.2: Programme à l'intérieur de chart Matlab

II.5. Les résultats de simulation

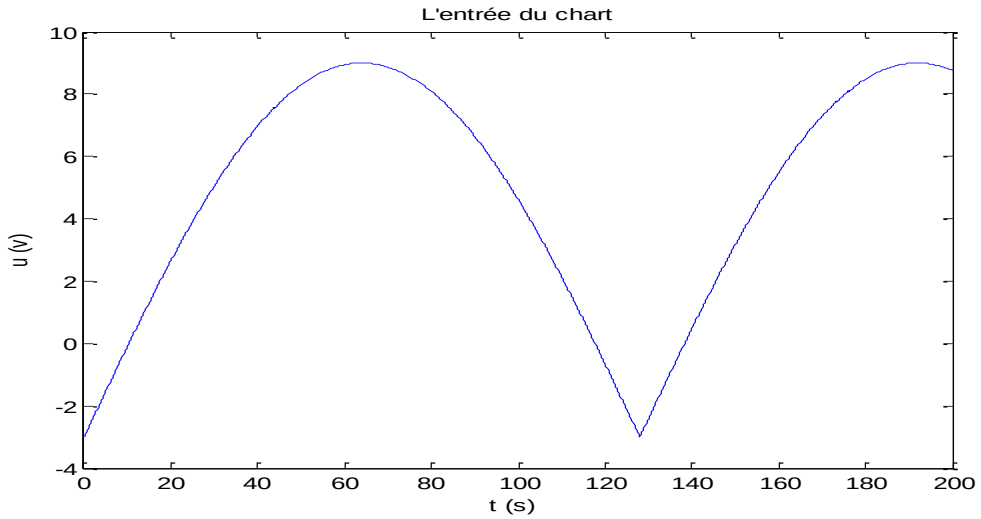


Figure II.3 : Le signal d'entrée U

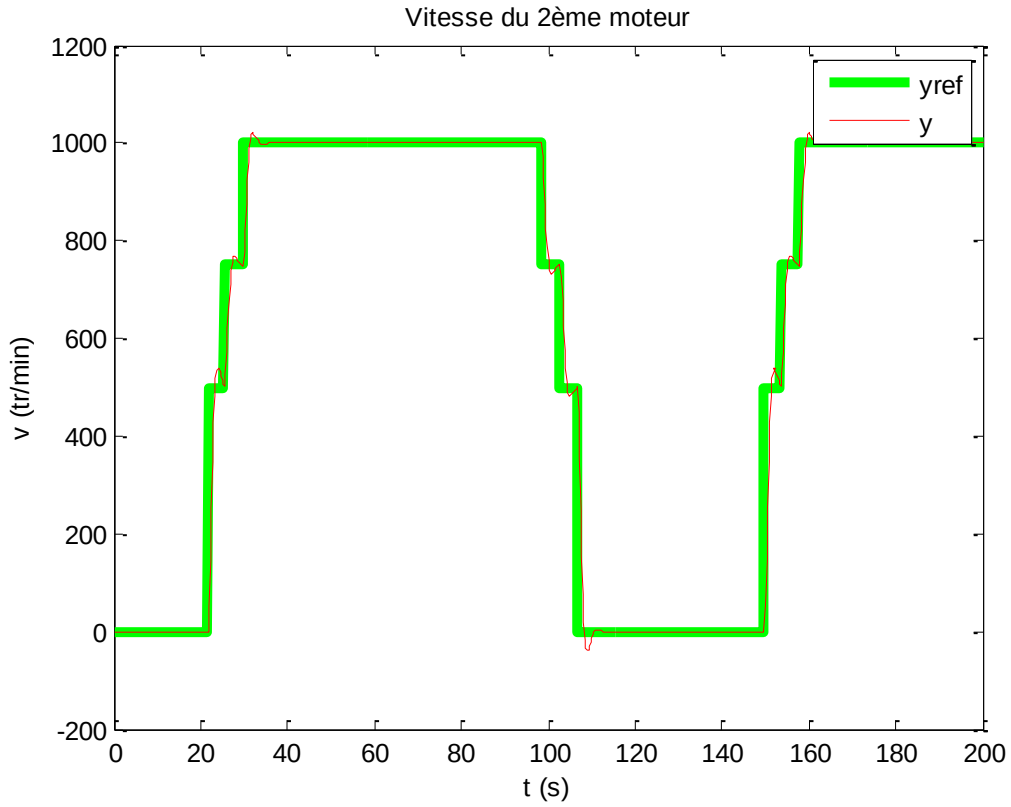


Figure II.4: *Les graphes de Yref et Y du deuxième moteur*

II.6. Conclusion

En conclusion, pour créer un programme de Stateflow et Simulink pour trois moteurs, il est important de comprendre les fonctionnalités de ces outils et de les utiliser ensemble pour modéliser et simuler le comportement des moteurs et leurs interactions.

Chapitre III

III.1. Introduction

Dans cette partie, nous allons faire de l'implémentation du programme réalisé dans la section précédente à la carte dSPACE ds1104 pour une nouvelle simulation en temps réel. La nouvelle génération des DSP de dSPACE se compose des dispositifs les plus compacts à ce jour et est dédiée au développement de prototypes de contrôle rapides. Ils réduisent le temps et le coût de développement des nouveaux algorithmes de contrôle. Leurs principaux avantages sont la simplicité d'utilisation et la commodité. En effet, l'interfaçage direct avec Simulink/MATLAB permet de concevoir les algorithmes de contrôle en utilisant les blocs diagrammes graphiques de Simulink pour ensuite les valider directement en temps réel [7].

Figure III.1 : La carte Dspace

Particulièrement, le kit ACE de dSPACE est un outil très attrayant destiné aux applications universitaires. Il existe deux versions différentes du kit ACE de dSPACE :

- le kit ACE 1104
- le kit ACE 1103

Ces kits offrent de nombreux avantages, notamment la possibilité de :

- Tester en temps réel les méthodes de contrôle les plus complexes
- Se concentrer uniquement sur la conception de la technique de commande désirée
- Optimiser en temps réel les contrôleurs

- Développer une expérience pratique en milieu universitaire
- Travailler avec une interface Windows facile à utiliser
- Implanter directement les modèles créés dans Simulink et les exécuter en temps réel dans dSPACE
- Observer automatiquement le comportement des systèmes lors de changements de paramètres. [8]

III.2. Domaine d'application

Les champs d'application typiques de dSPACE comprennent :

- Les entraînements et les moteurs
- Les moteurs à induction triphasés
- Le contrôle du bruit et des vibrations
- L'hydraulique et le pneumatique
- La robotique
- L'automatique et le traitement du signal.

La facilité d'utilisation de ce kit découle de son interface directe avec Simulink. Une fois la méthode de contrôle développée et conçue dans Simulink, que ce soit par des blocs diagrammes ou des programmes en langage C, elle est compilée. Ensuite, les données sont envoyées à la carte dSPACE pour démarrer l'application en temps réel.

III.3. Implémentation de la commande

Pour l'implémentation de notre commande il faut disposer du matériel suivant :

- Un Pc qui contient les différents logiciels : Matlab v13b, Simulink, ControlDesk.
- Une carte dSPACE "RTI1104".
- 03 moteurs à courant continu.
- Une source de tension variable.

III.4. Real Time Interface (RTI)

Cet outil MATLAB automatise la génération de code C à partir des diagrammes de schéma blocs de Simulink. Il produit un code personnalisable et optimisé, adapté aux modèles Simulink. Il prend en charge tous les types de systèmes, tels que continus, discrets ou hybrides, et permet la simulation externe ainsi que la connexion aux équipements physiques. De multiples cibles sont disponibles pour son utilisation.[7][8]

Grâce à l'Interface Temps Réel (RTI), il est simple de faire fonctionner des modèles de fonctions sur la carte ds1104, de configurer toutes les entrées/sorties graphiquement, d'ajouter des blocs dans un schéma-bloc Simulink, et de générer le code du modèle via Simulink Coder (RealTime Workshop).[11]

III.5. Présentation de la carte *dSPACE "RTI1104"*

La carte DSP utilisée est la dSPACE "RTI1104". Son processeur principal est un MPC8240, doté d'un cSur Power PC 603e et d'une horloge interne à 250 MHz. Elle dispose d'une mémoire Flash de 8 Mo et de 32 Mo de SDRAM.

Elle est équipée de 8 convertisseurs analogiques-numériques (4 en 16 bits, 4 en 12 bits), de 8 convertisseurs numériques-analogiques (CNA) de 16 bits pouvant fournir une tension analogique entre -10V et +10V, d'une liaison série, de 2 codeurs incrémentaux, de 20 entrées-sorties numériques, d'une DSP esclave (TMS320F240) et de 3 timers (32 bits) pouvant fonctionner de manière indépendante.

Pour programmer la DSP, il est nécessaire de créer un schéma dans l'environnement Simulink de MATLAB. Il est recommandé de travailler dans le répertoire approprié pour élaborer le fichier Simulink, car la compilation génère de nombreux fichiers contenus dans un nouveau répertoire à chaque étape de compilation.

Une fois le schéma Simulink terminé et sauvegardé, l'étape suivante consiste à générer le code associé et à le transférer dans la DSP. Dans la bibliothèque Simulink, on trouve une librairie nommée "dSPACE RTI1104", où l'on peut choisir les composants pour communiquer avec la carte DSP. En double-cliquant sur chaque bloc, on peut sélectionner les champs dans lesquels écrire ou lire les données.

La compilation réussit, le chargement du programme et son exécution dans la DSP sont effectués automatiquement. [7] [8]

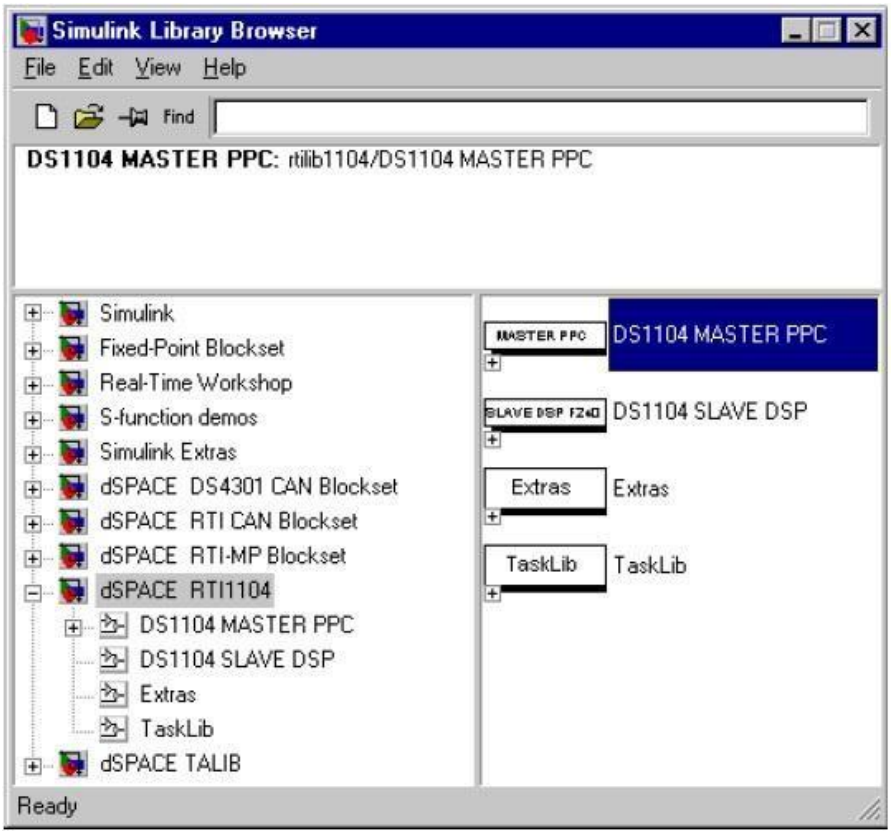


Figure III.2 : Simulink Library Browser

Puisque nos signaux sont numériques, nous utilisons le port des entrées-sorties numériques "I/O" (voir Figure 19).

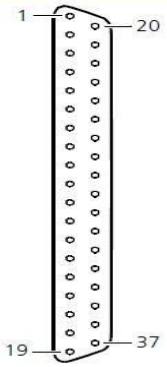
Connector (CP18)	Pin	Signal	Pin	Signal
	1	GND		
	2	SCAP1	20	GND
	3	SCAP3	21	SCAP2
	4	GND	22	SCAP4
	5	ST2PWM	23	ST1PWM
	6	GND	24	ST3PWM
	7	SPWM1	25	GND
	8	SPWM3	26	SPWM2
	9	SPWM5	27	SPWM4
	10	SPWM7	28	SPWM6
	11	SPWM9	29	SPWM8
	12	GND	30	GND
	13	GND	31	GND
	14	GND	32	GND
	15	GND	33	GND
	16	SSIMO	34	SSOMI
	17	SCLK	35	SSTE
	18	VCC (+5 V)	36	GND
	19	VCC (+5 V)	37	GND

Figure III.3 : Des entrées-sorties numériques « I/O »

Pour compiler un schéma Simulink, suivez ces étapes :

- Accédez au menu "Simulation", puis sélectionnez "Simulation Parameters".

- Dans l'onglet "Solver", réglez le "Type" sur "fixed-step" et la "fixed step" sur la période d'échantillonnage choisie (Te). Définissez le "stop time" sur "inf".

- Dans l'onglet "Code Generation", choisissez "System target file" et chargez le fichier "rti1104.tlc".

- Retournez dans le menu "Simulation" et sélectionnez "Start".

La compilation produit plusieurs fichiers, dont deux sont particulièrement importants :

- *.sdf, qui répertorie tous les paramètres du schéma bloc, utilisés dans Control Desk pour établir le lien entre le firmware et l'IHM (Interface Homme Machine).

- *.ppc, le firmware chargé dans la carte DSP.

III.6. Démarrage de ControlDesk

ControlDesk est une interface graphique qui facilite l'administration de la carte dSPACE, la gestion des applications et des fonctions de contrôle. Ce logiciel est compatible avec toutes les cartes dSPACE dsxxxx.

Pour l'utiliser :

i. Ouvrez le menu "Start" et sélectionnez "dSpaceTools-ControlDesk".

ii. Dans la fenêtre ControlDesk, cliquez sur "Platform".

Figure III.4 : ControlDesk en cours de lancement

III.7. Manipulation

Dans cette partie, nous allons procéder à la manipulation en utilisant les trois moteurs répartis dans trois maquettes différentes selon l'ordre suivant : la première maquette avec le premier moteur, la deuxième maquette avec le deuxième moteur et la troisième maquette avec le troisième moteur, qui sont reliées par des fils la figure ci-dessous représente le montage de la commande.

Figure III.5 : *Montage de la commande*

III.7.1. Programme de la commande

Pour l'implémentation nous utiliserons la carte Dspace, le programme se fera dans Matlab-Simulink, nous utiliserons le même programme utilisé pour la simulation avec quelques changements, les signaux issus des fonctions de transfert seront issus des trois moteurs via la carte d'acquisition de données, et les signaux de commandes qui sont connectés aux fonctions de transfert seront connectés aux entrées des trois moteurs via la carte d'acquisition de données.

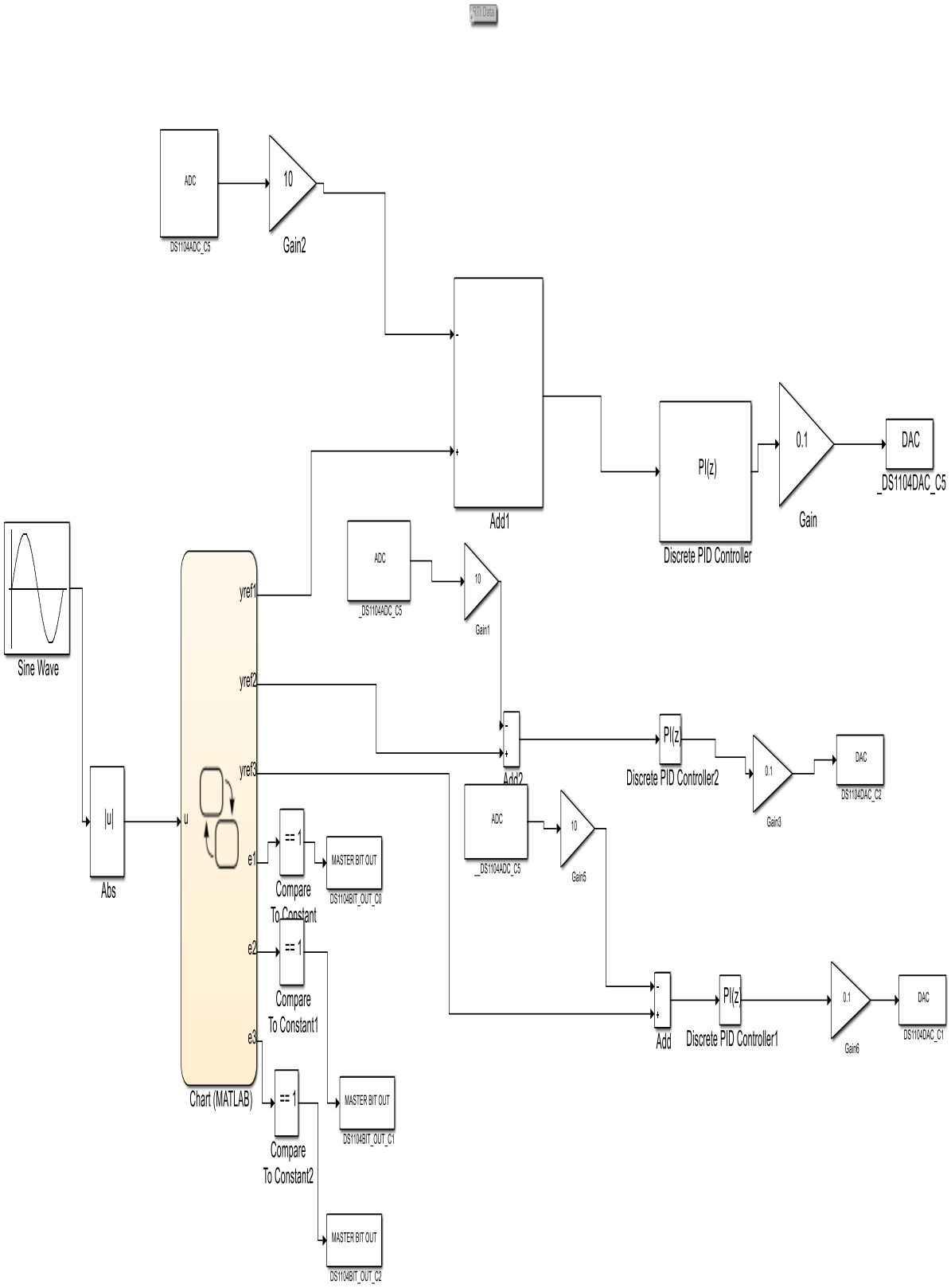


***Figure III.6**:* Programme *de la commande réelle*

III.7.2.Identification des paramètres des moteurs

Pour identifier les paramètres du moteur 2 nous utiliserons « Identification Toolbox » de Matlab dont voici les étapes :

On lance l'application « identification toolbox »

On importe les deux signaux d'entrée u et sortie y du moteur 2 trouvées lors d'un essai pratique. Ces deux vecteurs doivent être sous forme colonne

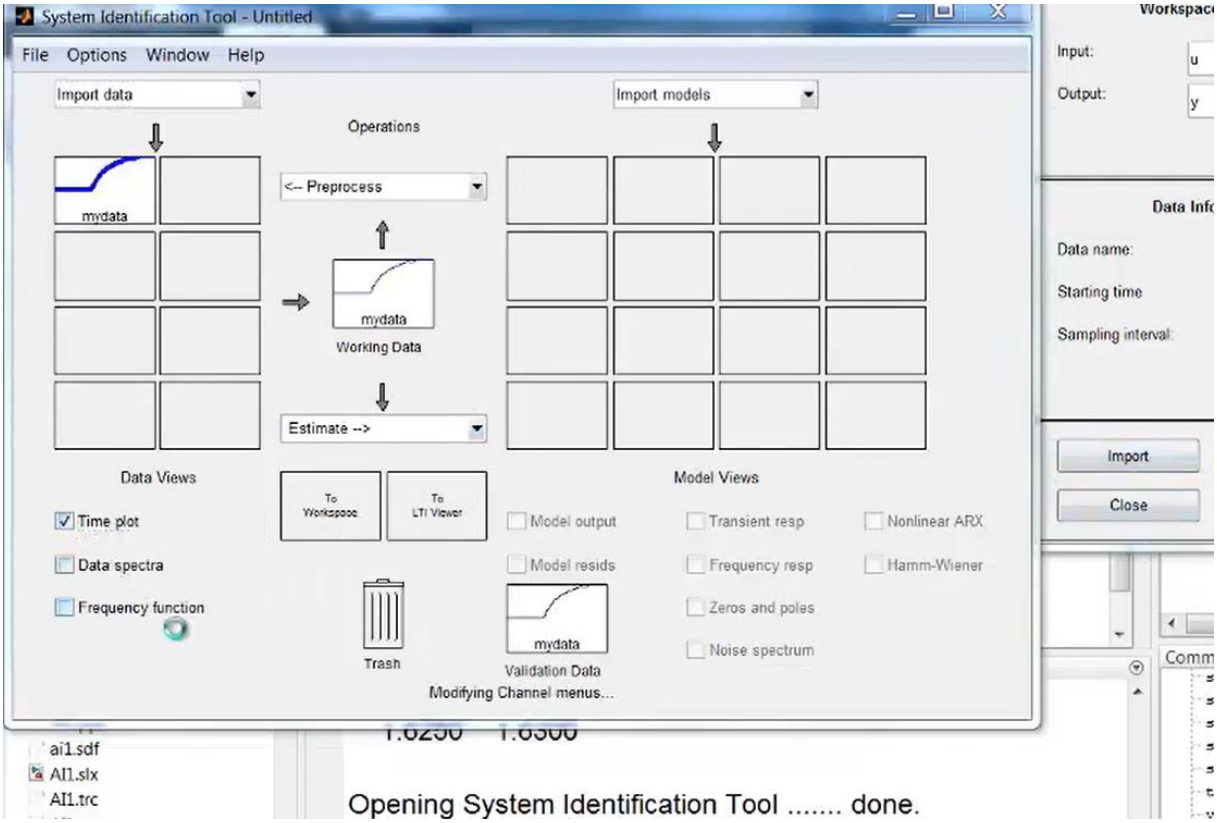


***Figure III.7** : Fenêtre d'identification Toolbox*

Après avoir introduit la première valeur du temps ainsi que la période d'échantillonnage, on clique sur 'Estimate', et on choisit le nombre de pôles et de zéros.

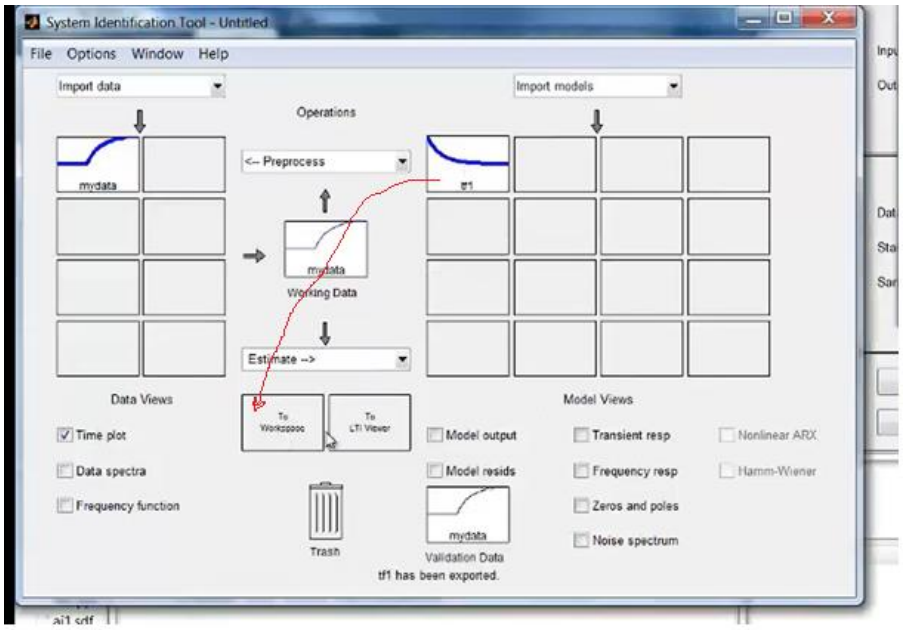


Figure III.8: *Le glissement de fonctionn de transfert vers « to workspace »*

On fait glisser la fonction de transfert obtenue tf1 vers 'to workspace' comme indiqué par la flèche rouge

On obtient la fonction de transfert suivante pour le moteur 2

G2(z) =)= $\frac{0.55}{z-0.54}$, te=0.25

On répète la même procédure pour les deux autres moteurs

G1(z) , $\frac{z}{z^2-0.18z+0.016}$, te=0.25 pour le moteur 1

G3(z) = $\frac{0.2z}{z^2-1.24z+0.4}$, te=0.25,pour le moteur 3

III.7.3. Calcul de paramètres des régulateurs du système par toolbox

Nous allons expliquer les méthodes à suivre pour calculer les paramètres du moteur

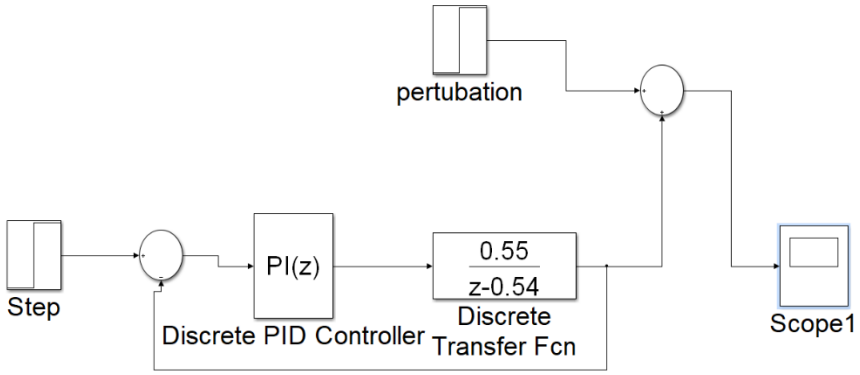


Figure III.9: Programme avec PID

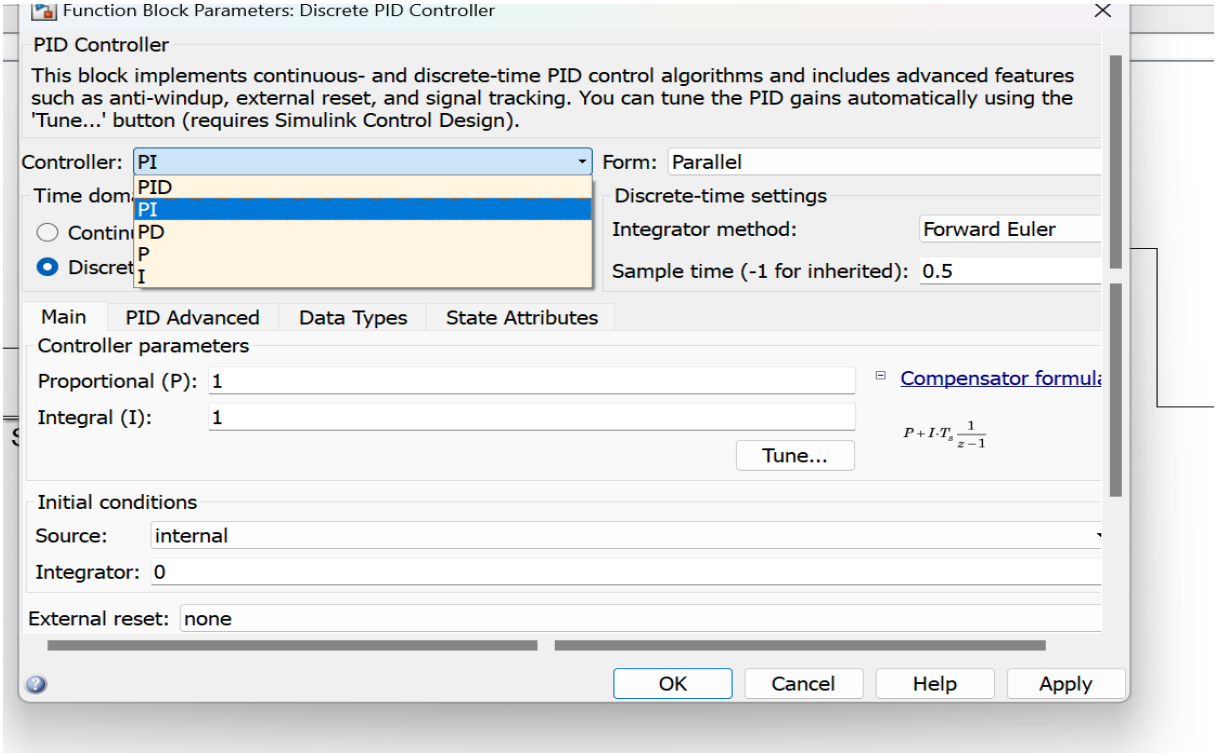


Figure III.10: *Choix des paramètres du régulateur PI*

Pour trouver les paramètres du PI, nous utilisons PID Tune toolbox

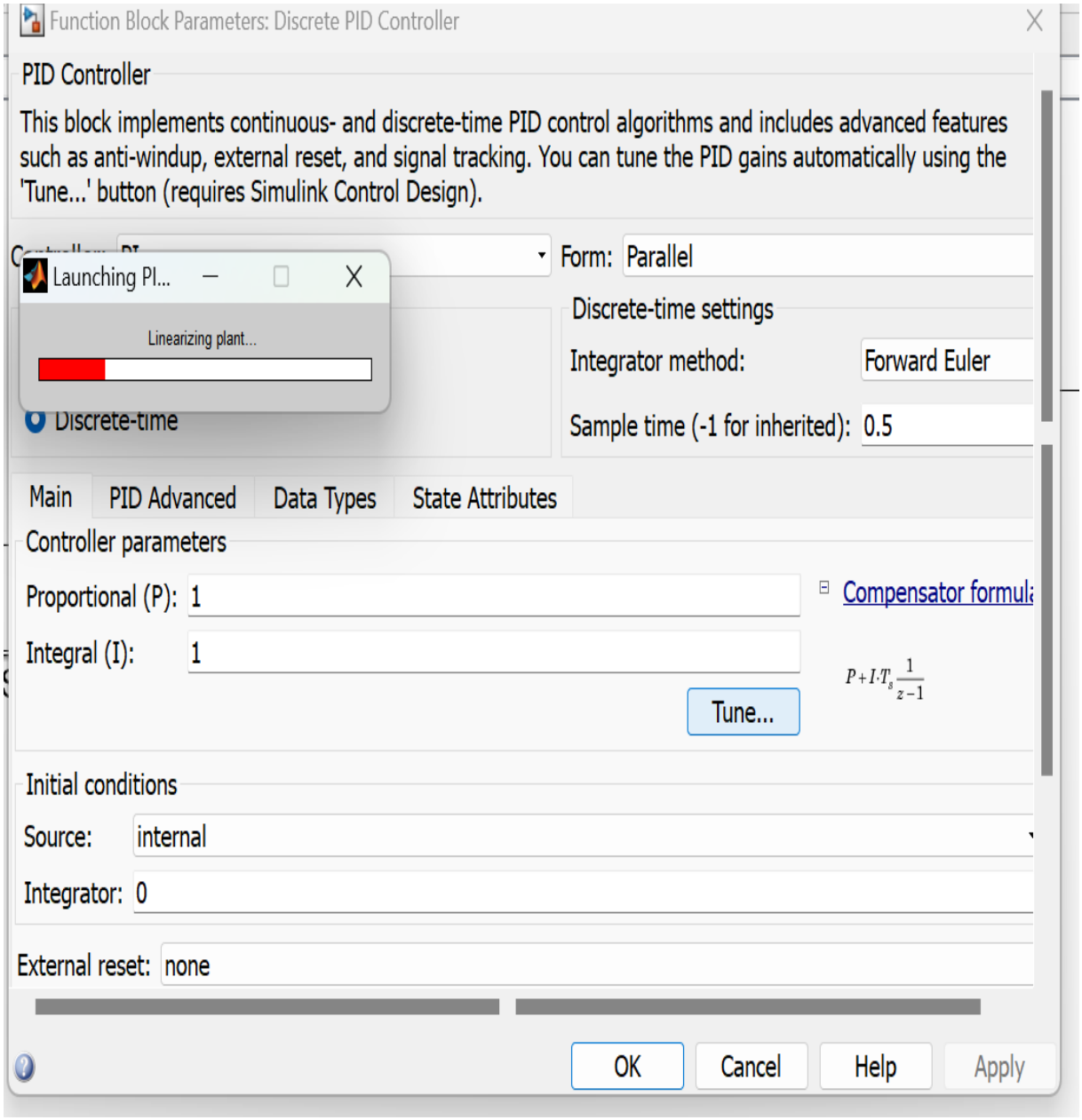


Figure III.11 : Lancement toolbox « Tune »

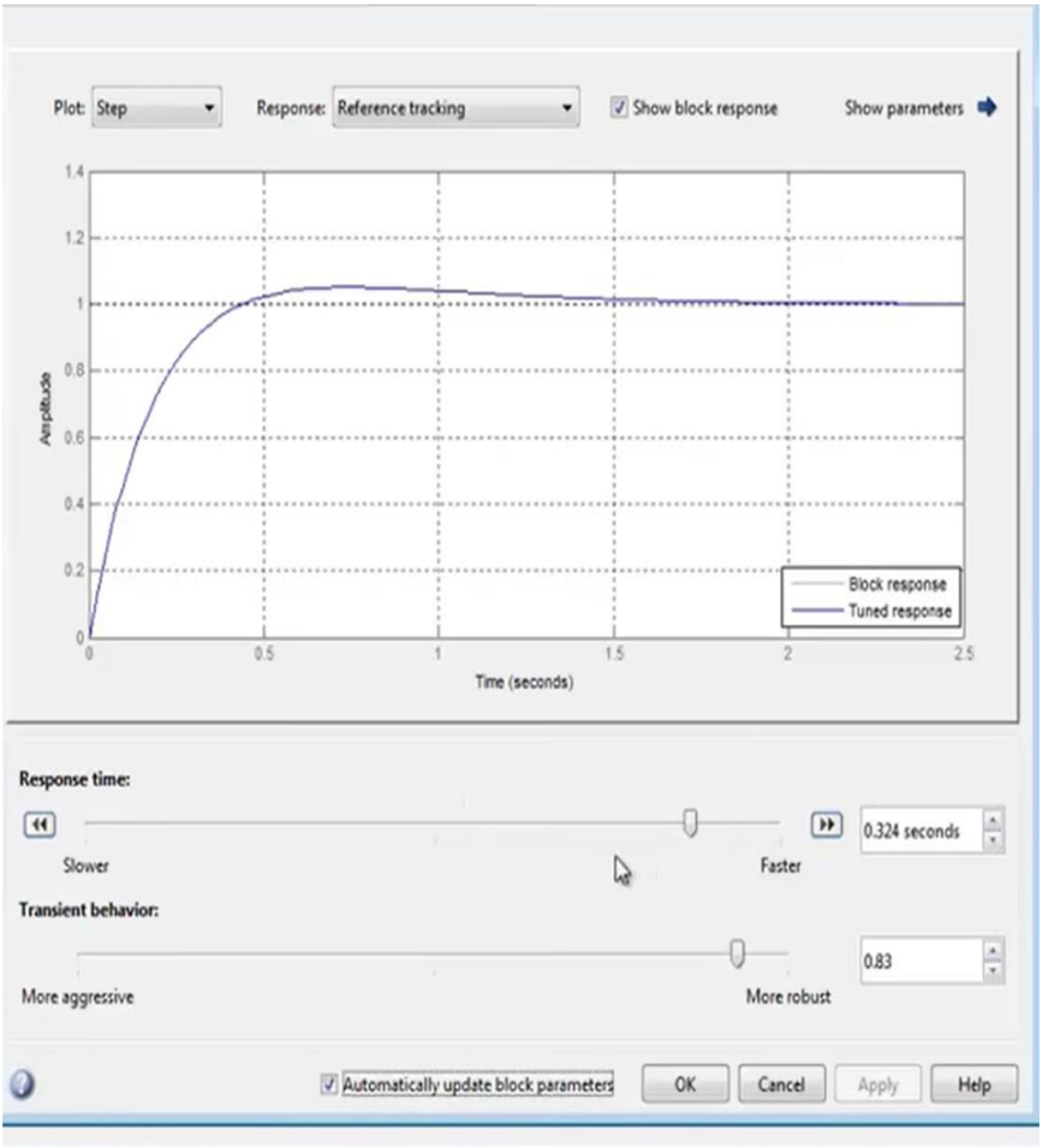


Figure III.12: Fenêtre d'ajustement les paramètres du PID en fonction du temps de réponse

On peut ajuster les paramètres du PID en fonction du temps de réponse souhaité

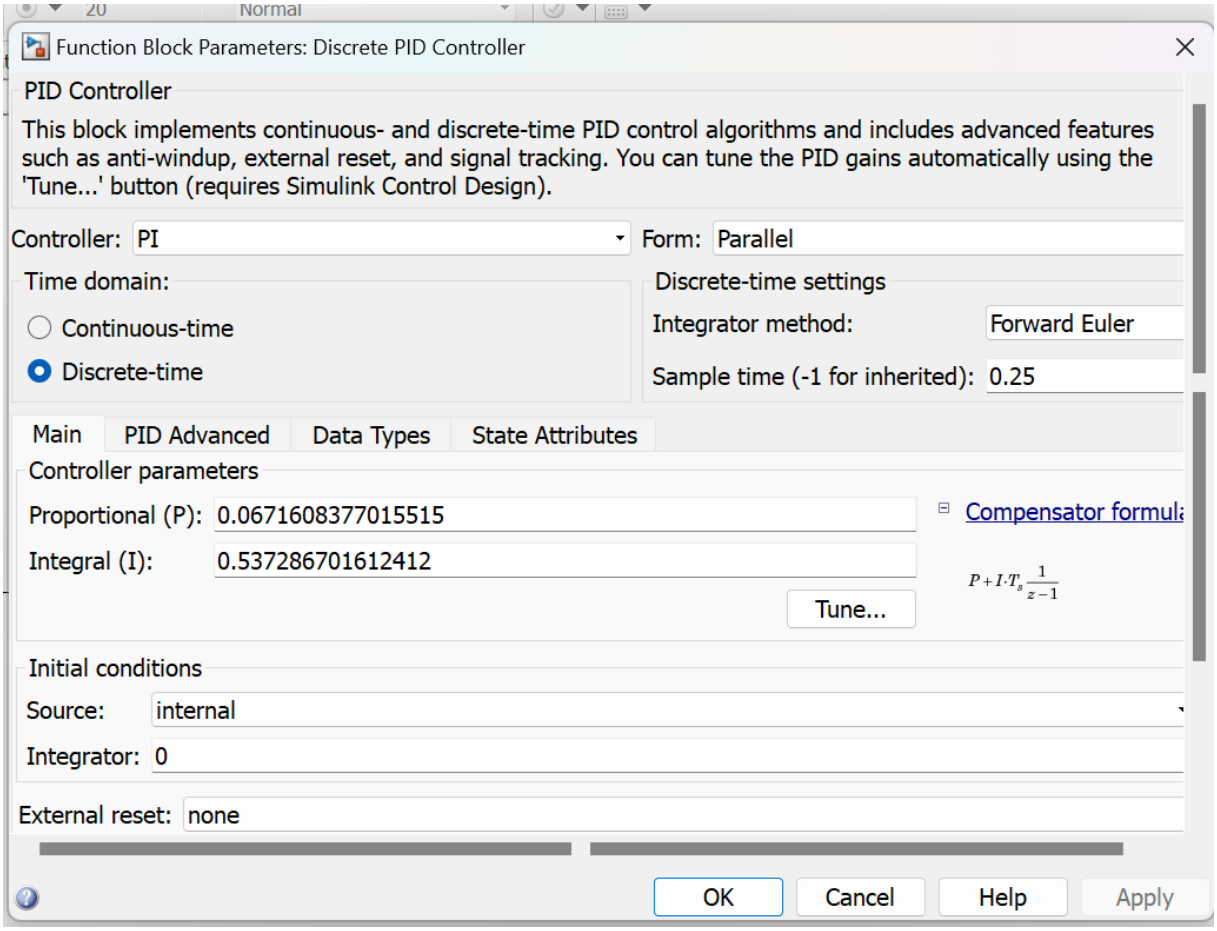


Figure III.13: Résultats des paramètres du régulateur PI

Nous allons maintenant définir directement les paramètres du PID pour les deux autres moteurs sans montrer les étapes à l'aide de captures d'écran, en sachant que les étapes sont identiques à celles du premier moteur. En plus, les deux autres moteurs 1 et 3

Voici les paramètres du régulateur PI pour le premier moteur 1.

Kp=0.029, Ti=1/0.23

Pour le 3ème moteur

Kp=0.11, Ti=1/0.41

III.7.4.Les résultats pratiques

Après identification et calcul des paramètres PID de nos trois moteurs, nous allons maintenant piloter directement les moteurs depuis le PC via le convertisseur analogique-numérique (envoi du signal du PC vers moteur ou vice-versa). Ensuite, nous avons enregistré les courbes des résultats de nos moteurs qui tournent réellement à l'aide de ControDesk avec le plotter. Les mesures seront ensuite transférées vers la fenêtre de commande en exécutant un script pour tracer les graphes dans MATLAB.

- Le premier moteur est en marche et les deux autres moteurs sont à l'arrêt. Voir annexe 2.
- Les deux premiers moteurs sont en marche et le troisième moteur est à l'arrêt. Voir annexe 3.
- Le script à réaliser pour tracer les graphes à l'aide des mesures enregistrées au ControDesk via le plotter porte sur le fichier nommé 'mesure1'
- Résultat pratique des états des moteurs qui sont sorties de chart Matlab

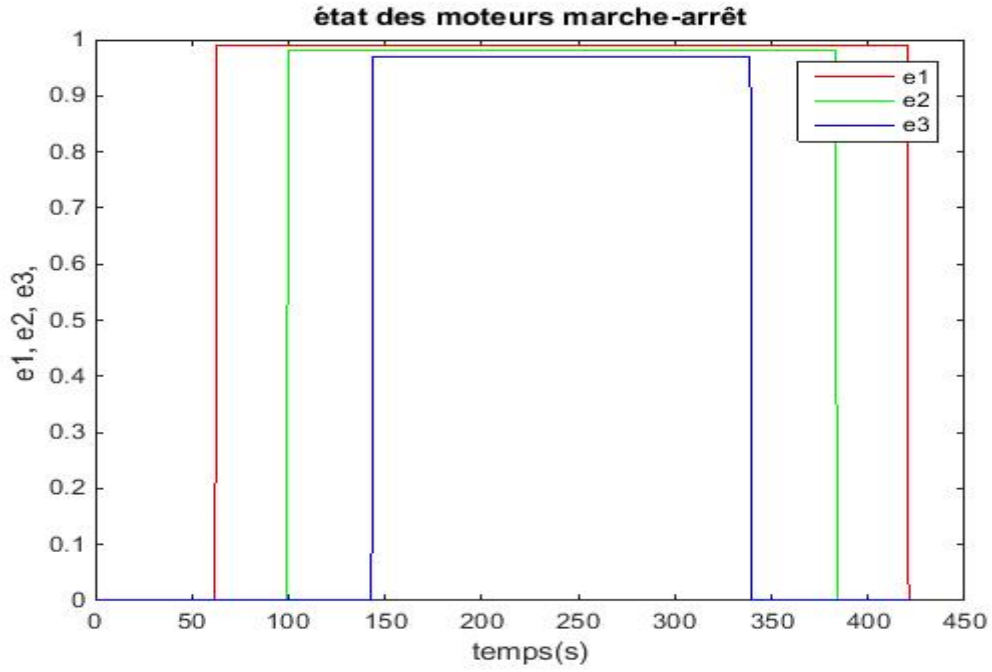


Figure III.14: *Les états des trois moteurs*

- Résultat pratique de Yref2 et Y2

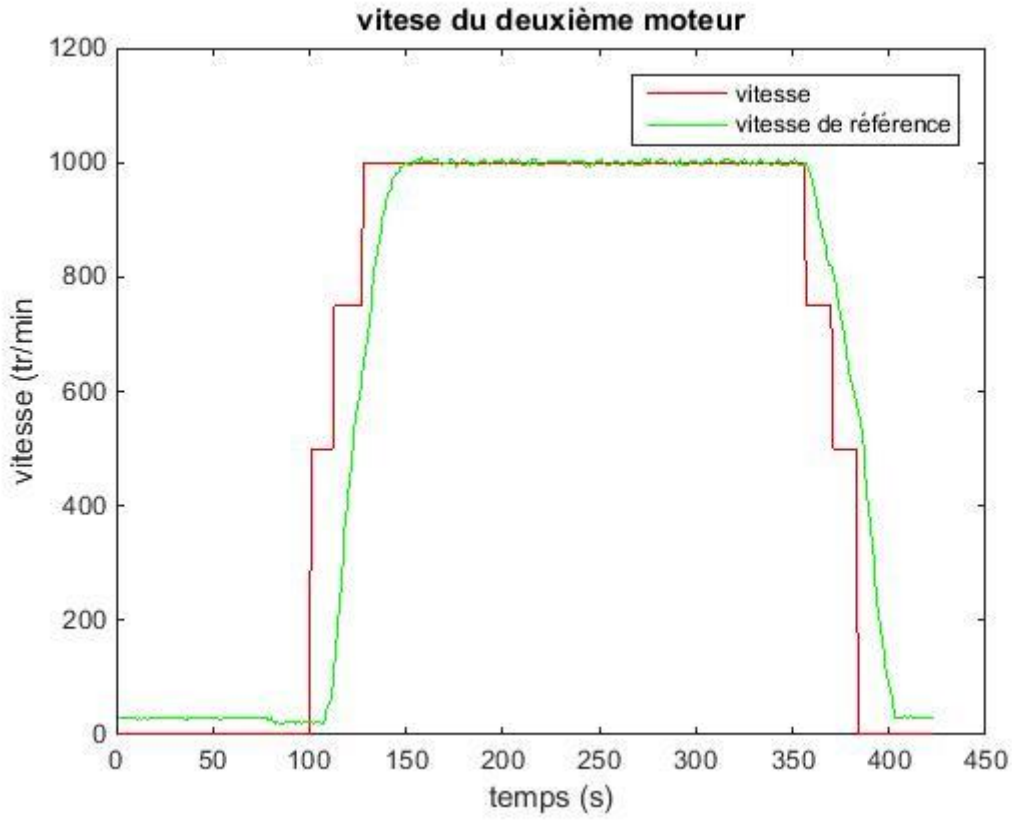


Figure III.15: Les résultats de la commande réelle avec deuxième moteur

- L'entrée du chart Matlab U

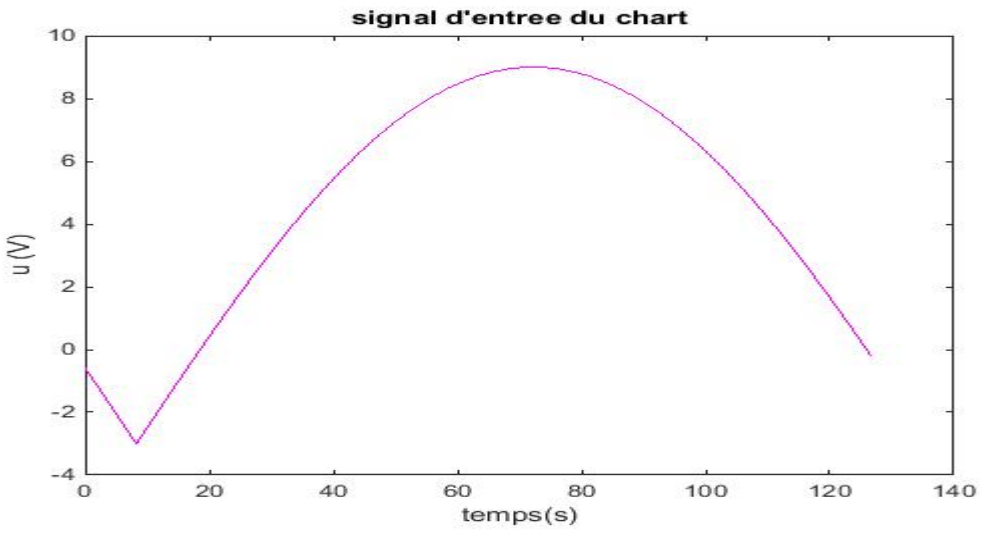


***Figure III.16**:* L'entrée U du chart

III.8. Conclusion

L'implémentation de commande sur la carte dSPACE pour les trois moteurs répartis dans différentes maquettes a nécessité une méthodologie rigoureuse et systématique. À travers l'identification des paramètres spécifiques de chaque moteur et l'établissement des fonctions de transfert correspondantes, nous avons réussi à intégrer efficacement les commandes. Cette démarche nous a permis de garantir la synchronisation et la performance optimale des systèmes impliqués, ouvrant ainsi la voie à des applications futures plus complexes et innovantes dans le domaine de la modélisation et du contrôle des systèmes mécaniques.

Conclusion générale

Conclusion générale

Stateflow est un outil de modélisation et de simulation de logique de décision combinatoire et séquentielle qui permet de concevoir et de développer des systèmes à événements discrets et continus. Voici quelques types de systèmes qui peuvent être modélisés avec Stateflow :

Systèmes à événements discrets : Stateflow est particulièrement adapté pour modéliser les systèmes qui passent d'un état à un autre en réponse à des événements spécifiques. Cela inclut des systèmes de supervision, de planification des tâches et de gestion des pannes.

Systèmes hybrides : Stateflow permet de simuler les systèmes qui combinent des comportements discrets et continus, comme des systèmes de contrôle de mouvement où le déplacement est continu mais les rebonds sur le sol sont des événements discrets.

Systèmes de supervision : Stateflow est utilisé pour concevoir des systèmes de supervision qui surveillent et contrôlent des processus industriels, des réseaux de communication, des systèmes de gestion d'incidents, etc…

Systèmes de planification des tâches : Stateflow permet de modéliser les systèmes qui planifient et exécutent des tâches, comme des systèmes de gestion de production, des systèmes de gestion de stock, etc…

Systèmes de gestion des pannes : Stateflow est utilisé pour concevoir des systèmes de gestion des pannes qui détectent et résolvent les problèmes techniques, comme des systèmes de maintenance, des systèmes de surveillance de la santé des systèmes, etc…

Systèmes de communication : Stateflow permet de modéliser les systèmes de communication qui échangent des informations, comme des systèmes de télécommunications, des systèmes de gestion de données, etc…

Systèmes embarqués : Stateflow peut être utilisé pour générer du code pour des systèmes embarqués, comme des systèmes de contrôle de véhicules, des systèmes de gestion de maison intelligente, etc...

En résumé, Stateflow est un outil versatile qui permet de modéliser et de simuler une grande variété de systèmes, allant des systèmes à événements discrets aux systèmes hybrides, en passant par les systèmes de supervision, de planification des tâches, de gestion des pannes, de communication et de systèmes embarqué

Références

Références

[1] MathWorks Stateflow Documentation, « https://matlabpourtous.com »

[2] MathWorks File Exchange pour Stateflow , « https://eduscol.education.fr »

[3] Steven T. Karris, « Introduction to Stateflow with Applications » , Orchard, 2007

[4] Nassim Khaled, «Virtual Reality andAnimation for MATLAB and Simulink Users »,Springer, 2012.

[5] The MathWorks Inc Simulink, « http ://www.mathworks.com/products/simulink »

[6] Matlab, « Helpe Matllab partie Exemple »

[7] L. Yacoubi, « *L'utilisation du DS1104 de DSPACE* », Ecole de technologie supérieur université du Quebec ,2020.

[8] S.CLAEYS, O.RENIER, « *Développement De Commande à Temps Réel* », Rapport de projet de fin d'étude, Université Des Sciences Et Technologiques De Lile, 2019 .

[9] M.BENDJEDIA, « Synthèse d'algorithmes de commande sans capteurs de moteurs pas à pas et implantation sur architecture programmable », Thèse de Doctorat, Université de France-Comte, 2017.

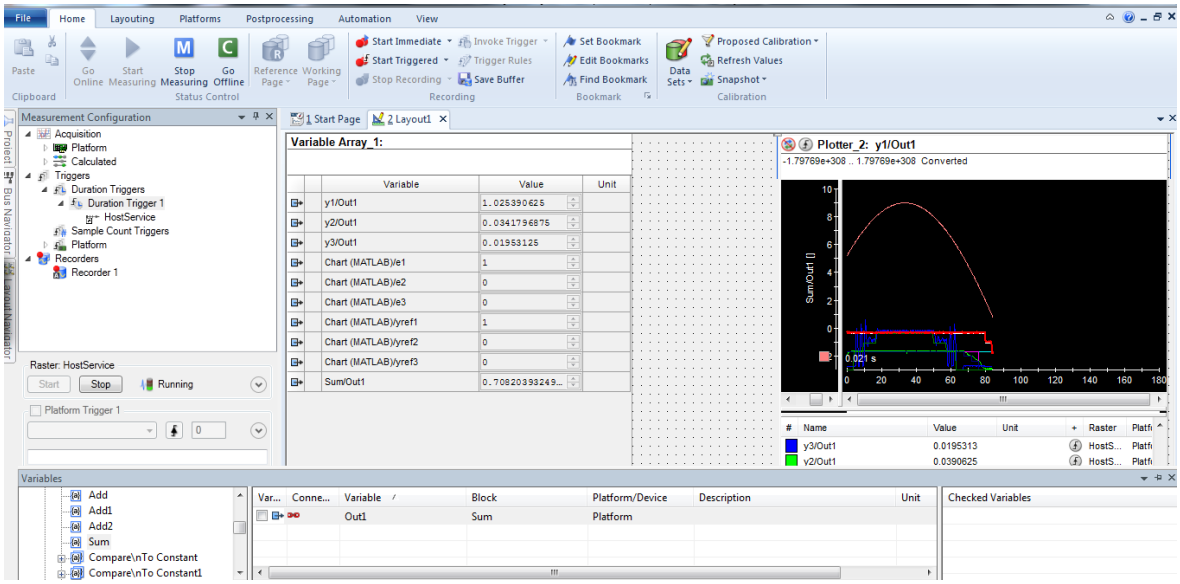
[10] A.Hably, J.Dumon, « Observation et commande par retour d'état d'un procédé de bacs Co

Annexes

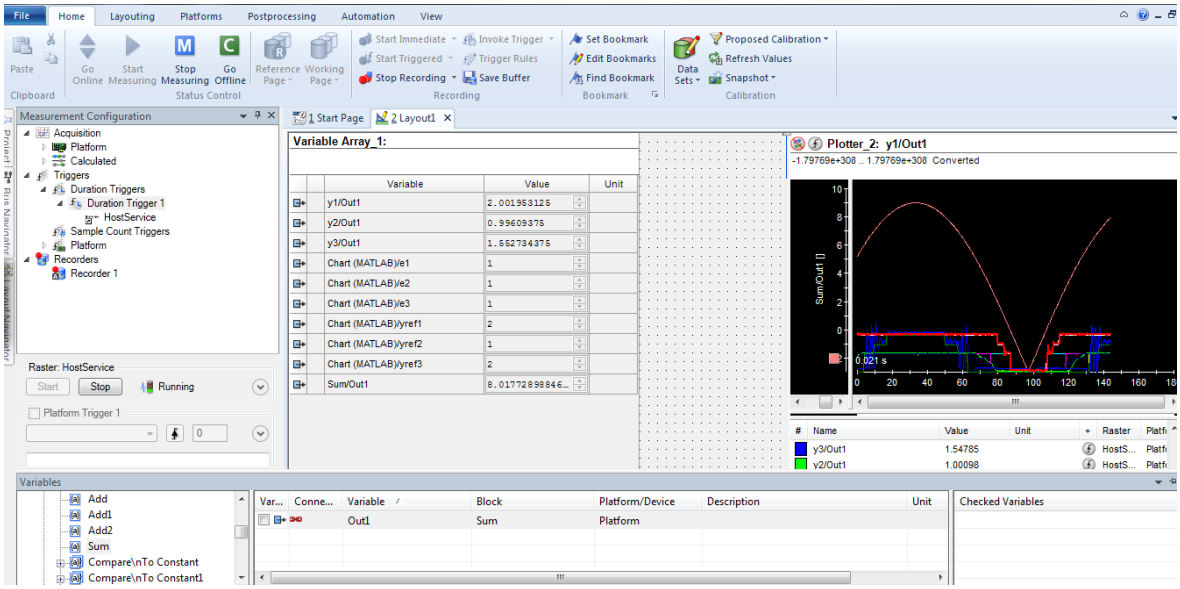
Annexe 01


ADCH1
ADCH2
ADCH3
ADCH4
ADCH5
ADCH6
ADCH7
ADCH8
DACH1
DACH2
DACH3
DACH4
DACH5
DACH6
DACH7
DACH8
DigitalIO
UART RS232
UART RS422 RS485
dSPACE
CLP1104
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
IO10
IO11
IO12
IO13
IO14
IO15
IO16 / INT1
IO17 / INT2
IO18 / INT3
IO19 / INT4
SSTE
SCLK
SSIMO
SSOMI
TXD
RXD
DCD (CTS)
RTS (-RTS)
DTR (TXD)
CTS (-CTS)
DSR (RXD)
VCC
SPWM1
SPWM2
SPWM3
SPWM4
SPWM5
SPWM6
SPWM7
SPWM8
STTPWM
ST2PWM
ST3PWM
SCAP1
SCAP2
SCAP3
SCAP4
Phi0 1
Phi90 1
Index 1
Phi0 2
Phi90 2
Index 2

Annexe 02



Annexe 03



Résumé

Nous devons commander un système de trois moteurs à courant continu, chacun ayant une fonction de transfert discrète appelée G(z) : le premier est d'ordre un et les deux autres sont d'ordre deux. Chaque moteur (i) avec i (1,2,3) possède une entrée yref (i), qui est la sortie du diagramme MATLAB. Le premier moteur s'allume lorsque 0 < U < 3, le deuxième lorsque 3 < U < 6, et le dernier lorsque 6 < U < 9. Ici, U est une entrée du diagramme MATLAB qui est une fonction cosinus. Pour la commande efficace, nous avons utilisé l'environnement Stateflow de MATLAB.

Abstract

We need to control a system of three DC motors, each with a discrete transfer function called G(z): the first is of order one and the other two are of order two. Each motor (i) with i (1,2,3) has an input yref (i), which is the output of the MATLAB diagram. The first motor turns on when 0 < U < 3, the second when 3 < U < 6, and the last when 6 < U < 9. Here, U is an input to the MATLAB diagram, which is a cosine function. For efficient control, we used MATLAB's Stateflow environment.