

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE  
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE  
SCIENTIFIQUE

*Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj*

**Faculté des Sciences et de la technologie**

**Département d'Electronique**

# **Mémoire**

*Présenté pour obtenir*

LE DIPLOME DE MASTER

FILIERE : ELECTRONIQUE INDUSTRIELLE

Spécialité : INDUSTRIES ELECTRONIQUES

Par

- BERBAOUI MOHAMMED ELSAID
- OUNIS NOUR ISLAM

*Intitulé*

*Etude et implémentation d'un PID d'ordre fractionnaire sur un  
microcontrôleur STM32*

*Évalué le:* /09/2021

*Par la commission d'évaluation composée de\* :*

<i>Nom &amp; Prénom</i>	<i>Grade</i>	<i>Qualité</i>	<i>Etablissement</i>
<i>M.</i>	<i>MCB</i>	<i>Président</i>	<i>Univ-BBA</i>
<i>M. Talbi Mohamed Lamine</i>		<i>Encadreur</i>	<i>Univ-BBA</i>
<i>M.</i>		<i>Examineur</i>	<i>Univ-BBA</i>

*Année Universitaire 2020/2021*

# **Remerciements**

Je tiens à exprimer mes chaleureux remerciements à  
Dr Talbi Mohamed Lamine el amine pour avoir  
m'encadrer dans la réalisation de ce travail et pour son  
aide qu'il m'a apporté

Je remercie aussi tous mes amis pour leur soutien  
moral et toutes personnes ayant aidé de près ou de loin  
pour réaliser ce travail.

## **Dédicaces**

Je dédie ce travail humble à mon père et à ma mère, que dieu les sauve, et à tous ceux qui m'ont appris ne serait-ce qu'un mot à toutes les étapes de l'éducation, à tous mes amis et collègues qui ont partagé les difficultés de l'étude et de l'éducation, à tous ceux qui m'ont été crédites de près ou de loi, à tous les ouvriers et employés de l'université BORDJ BOU RIRIDJ.

# Sommaire

Remerciements .....	I
Dédicaces.....	II
Sommaire.....	III
Liste des figures.....	V
Introduction générale.....	1
Chapitre I : Calcul d'ordre fractionnaire et correcteurs PID fractionnaire.....	2
I.1. Introduction .....	3
I.2. Le correcteur PID classique [1].....	3
I.3. Performances des systèmes réglés [2].....	4
I.3.1 Rapidité .....	5
I.3.2 Précision .....	5
I.3.3 Stabilité .....	5
I.4. But de la correction [3].....	5
I.5. Instabilité de l'action Dérivée [5].....	6
I.6. Algorithmes d'ajustement des paramètres de Contrôleur PID.....	6
I.7. Introduction au calcul fractionnaire .....	7
I.8. Opérateurs d'ordre fractionnaire.....	7
I.8.1 Définition de Riemann-Liouville (R-L) .....	7
I.8.2 Définition de Caputo .....	8
I.8.3 Définition de Grundwald-Leitnikov (G-L) .....	8
I.9. Correcteurs D'Ordre Fractionnaire .....	8
I.9.1 Correcteur d'ordre fractionnaire <b><math>PI\alpha D\mu</math></b> .....	8
I.10. Principe de fonctionnement.....	9
I.11. Structure de correcteur PID fractionnaire .....	10
I.12. La fractionalisation des Correcteurs PI et PID.....	11
I.13. Avantage du correcteur fractionnaire.....	12
I.14. Conclusion.....	13
Chapitre II : Programmation par FLOWCODE/STM32 .....	14
II.1. Introduction.....	15

II.2. L'architecture du microcontrôleur STM STM32 F103RB [15] .....	15
II.2.1 Définition .....	15
II.2.2 L'intérêt de carte STM32 NUCLEO .....	16
II.2.3 Description d'une carte NUCLEO-F103RB .....	16
II.2.4 L'environnement logiciel de développement « FLOWCODE » : [16] .....	17
II.3. Programmation graphique [17] .....	19
II.3.1 Les avantages .....	19
II.3.2 Les inconvénients.....	19
II.4. Création d'un projet Flowcode .....	20
II.5. Conclusion : .....	23
Chapitre III : Etude du correcteur fractionnaire et implémentation sous la carte STM32 .....	24
III.1. Introduction .....	25
III.2. Modélisation mathématique du moteur à courant continu [20] :.....	26
III.3. Cconception de contrôleur d'ordre fractionnaire [21] : .....	27
III.4. Algorithme de correcteur fractionnaire en utilisant le toolbox Matlab ninteger. ....	28
III.5. Implémentation de l'algorithme de commande sur une carte STM32. ....	29
III.6. Résultats et Simulation .....	30
III.6.1 Modélisation mathématique du moteur à courant continu [25] : .....	30
III.6.2 Conception de contrôleur d'ordre fractionnaire .....	32
III.6.3 Implémentation de l'algorithme de commande sur une carte STM32. ....	33
III.7. Conclusion .....	35
Conclusion générale .....	36
Bibliographique .....	37
Résumé .....	

## Liste des figures

<b>Figure I.1</b> Schéma bloc d'un système avec correcteur PID (boucle fermé).....	3
<b>Figure I.2</b> Performances d'un système de commande.....	4
<b>Figure I.3</b> Stabilité du système.....	5
<b>Figure I.4</b> Structure interne du $PI^\alpha D^\mu$ d'ordre fractionnaire.....	9
<b>Figure I.5</b> $PI^\alpha D^\mu$ (a) Ordre entier, (b) Ordre fractionnaire.....	10
<b>Figure I.6</b> Fractionalisation d'un intégrateur.....	11
<b>Figure I.7</b> Système de commande à retour unitaire classique.....	12
<b>Figure II.1</b> STM32 Nucleo-64 board.....	15
<b>Figure II.2</b> Composants de la carte NUCLEO-F103RB.....	17
<b>Figure II.3</b> L'interface FlowCode.....	18
<b>Figure II.4</b> les icons de l'interface flow code.....	18
<b>Figure II.5</b> creation d'un nouveau projet dans FlowCode.....	20
<b>Figure II.6</b> la choisisation du microcontrôleur.....	20
<b>Figure II.7</b> l'interface du creation ou declaration des variables sur flowcode.....	21
<b>Figure II.8</b> l'interface du creation d'algorithme flowcode.....	20
<b>Figure II.9</b> l'interface de Configuration la vitesse d'horloge.....	20
<b>Figure II.10</b> l'interface pour contrôler la simulation.....	22
<b>Figure III.1</b> Bras manipulateur.....	26
<b>Figure III.2</b> modèle général d'un moteur à courant continu.....	26
<b>Figure III.3</b> modèle mathématique d'un moteur à courant continu.....	27
<b>Figure III.4</b> Boucle ferme de commande.....	27
<b>Figure III.5</b> interface graphique du bloc npid du toolbox ninteger.....	29
<b>Figure III.6</b> Diagramme de Bode de la fonction de transfert du moteur.....	30
<b>Figure III.7</b> Modèle Simulink pour le contrôle de rétroaction du moteur à courant continu.....	31
<b>Figure III.8</b> la réponse échelon unitaire d'une boucle de commande sans saturation.....	31
<b>Figure III.9</b> la réponse échelon unitaire d'une boucle de commande avec actionneur saturation.....	32
<b>Figure III.10</b> l'interface graphique du toolbox npid ninteger.....	33
<b>Figure III.11</b> Le code programme Flowcode du correcteur fractionnaire.....	34

# Introduction générale

Le contrôleur PID reste le cas le plus utilisé dans la commande des systèmes. Récemment une attention particulière a été accordée à la conception de nouvelles commandes en exploitant les nouveaux concepts mathématiques tels que le calcul d'ordre fractionnaire.

Par comparaison aux correcteurs classiques, les correcteurs d'ordre fractionnaire possèdent en plus deux autres paramètres notés  $\gamma$  et  $\mu$  et qui présentent l'ordre d'intégration et de dérivation respectivement.

L'objectif de ce travail est l'étude et l'implémentation d'un correcteur PID fractionnaire qui sert à commander en vitesse un moteur à courant continu.

Le moteur que nous voulons commander est monté sur un bras manipulateur disponible au niveau du laboratoire pédagogique du département d'électronique.

L'algorithme de correcteur fractionnaire sera implémenté sur la carte de prototypage STM32 en utilisant l'environnement de programmation graphique Flowcode.

Pour ce faire nous avons organisé notre travail en le présentant en trois (03) chapitres :

Chapitre 01 : dans ce chapitre nous allons donner les différentes définitions de fonctionnement des PID classique et fractionnaire et leur calcul.

Chapitre 02 est consacré aux logiciels FLOWCODE et à la carte de développement STM32.

Chapitre 03 est destiné à la présentation résultats de simulation.

Nous terminons notre mémoire avec une conclusion générale.

## **Chapitre I :**

# **Calcul d'ordre fractionnaire et correcteurs PID fractionnaire**



## I.1. Introduction

Le calcul de l'ordre fractionnaire constitue la branche de mathématique traitant de la différenciation et l'intégration sous un ordre arbitraire de l'opération, à savoir l'ordre peut être n'importe quel nombre réel ou même complexe. Leur but était de prolonger la dérivation ou l'intégration d'ordre fractionnaire en employant non seulement un ordre entier mais également des ordres non entiers [1].

Dans ce chapitre, nous allons donner les définitions et les concepts de base du correcteur PID classique, du correcteur fractionnaire ainsi que le calcul d'ordre fractionnaire.

## I.2. Le correcteur PID classique [1].

Le correcteur PID agit de trois manières :

- Action **proportionnelle** : l'erreur est multipliée par un gain  $G$ .
- Action **intégrale** : l'erreur est intégrée et divisée par un gain  $T_i$ .
- Action **dérivée** : l'erreur est dérivée et multipliée par un gain  $T_d$ .

Il existe plusieurs architectures possibles pour combiner les trois effets (série, parallèle ou mixte), on présente ici la plus classique : une structure PID parallèle qui agit sur l'Erreur [1].

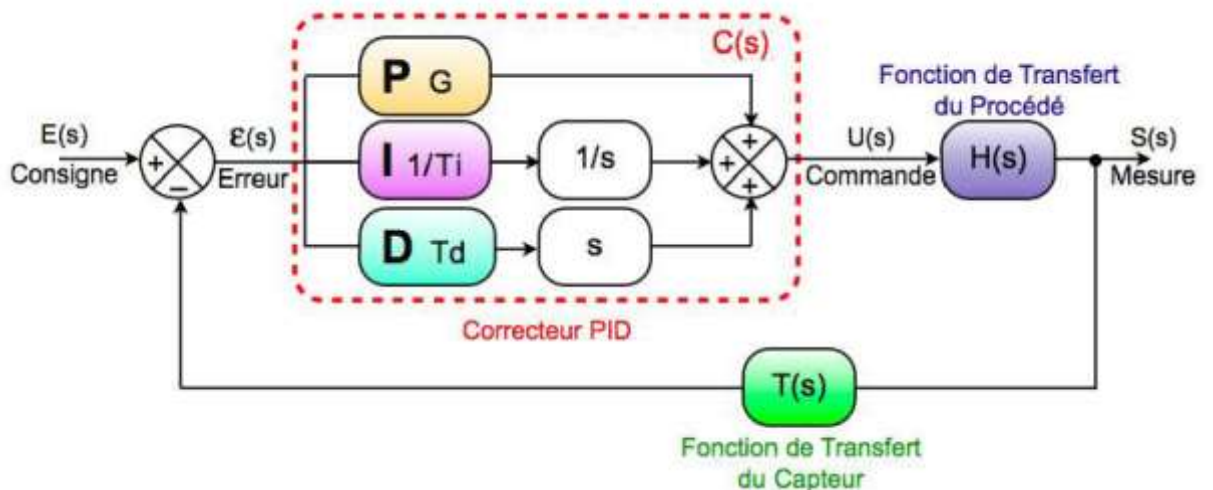


Figure I.1. Schéma bloc d'un système avec correcteur PID (boucle fermée).

La fonction de transfert avec la transformée de Laplace du régulateur PID parallèle est la somme des trois actions :

$$C(s) = K_p + K_i \times \frac{1}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \quad (\text{I.1})$$

Avec :

$\varepsilon(i)$  : Représente l'erreur de suivi, la différence entre la valeur d'entrée désirée ( $v$ ) et la sortie réelle( $s$ );

$e(t)$  : Signal de commande ;

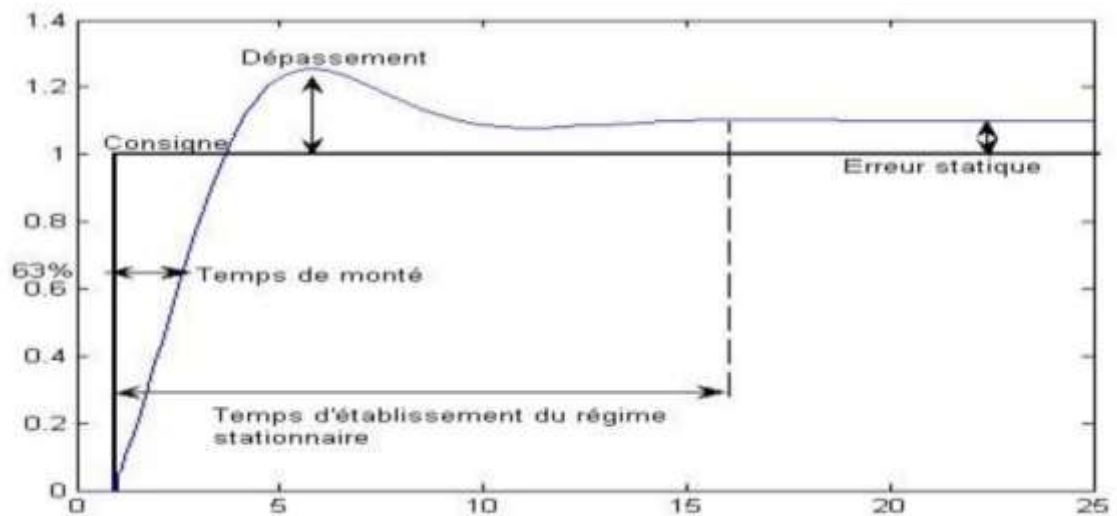
$K_p = G$  : Gain Proportionnel ;

$K_i = 1/T_i$  : Gain Intégral ;

$K_d = T_d$  : Gain Dérivative

### I.3. Performances des systèmes réglés [2]

Les performances des systèmes réglés définies dans un cahier des charges, sont illustrées par la **Figure I.2**:



**Figure I.2.** Performances d'un système de commande.

### I.3.1 Rapidité

La rapidité quantifie le temps de réponse du système. Elle correspond au temps de réaction de la sortie par rapport à la consigne. Le temps mis par la réponse pour ne plus dépasser 5% de la valeur finale. Ce temps est retenu comme critère de rapidité 5%. [2]

### I.3.2 Précision

La précision quantifié l'erreur lorsque l'équilibre est atteint, Avec l'entrée  $e(t)$  et la sortie  $s(t)$  de même nature. Autrement, un système est précis si la sortie suit la consigne en toutes circonstances avec un écart inférieur à la valeur définie dans un cahier des charges. [2]

### I.3.3 Stabilité

On dit qu'un système est stable lorsque celui-ci tend à revenir à son état d'équilibre pour une consigne constante, la sortie doit être constant. [2]

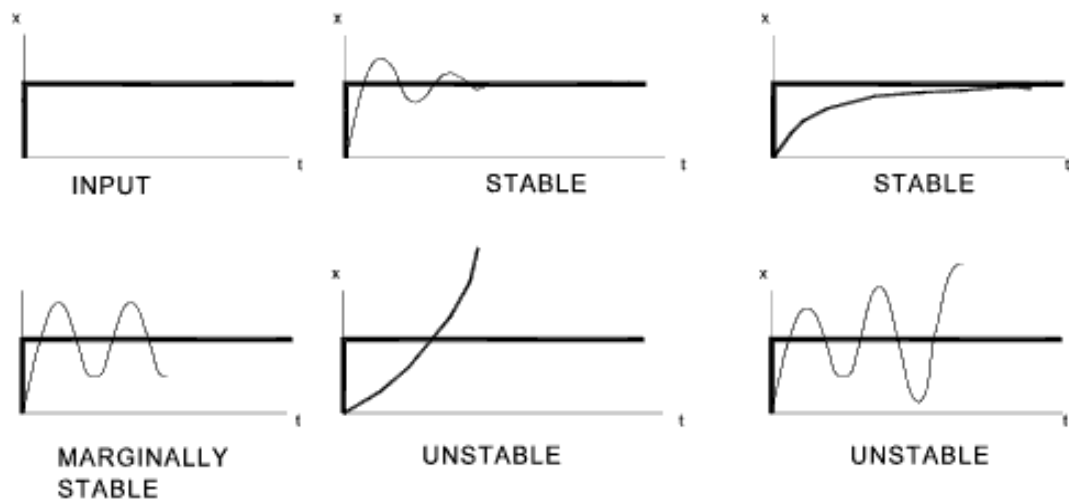


Figure I.3. Stabilité du système.

### I.4. But de la correction [3]

Le but de la correction est de doter l'asservissement des qualités (en termes de spécifications temporelles et fréquentielles) attendues, par le calcul et l'implantation du correcteur nécessaire.

Le concepteur de l'asservissement rencontre deux types de situations, dont il doit faire face :

- Assurer une réponse acceptable pour des signaux de consigne définis en fonction du temps
- Fournir des caractéristiques fréquentielles (gain, déphasage) demandées dans une bande de fréquences.

Le but de la correction est de doter l'asservissement des qualités attendues, par le calcul et l'implantation du correcteur nécessaire. Les opérateurs essentiels du correcteur sont réalisables à partir d'amplificateurs à courant continu et d'éléments résistances/capacités. La réalisation numérique peut se transposer aisément à partir d'un schéma analogique, en conservant la même organisation fonctionnelle et en associant un intégrateur numérique à chaque intégrateur électronique [3],[4].

### **I.5. Instabilité de l'action Dérivée [5]**

Dans un asservissement PID, le terme Dérivation peut parfois poser problème. En effet, prenons le cas d'un système fortement bruité comme un asservissement de la vitesse d'un moteur. Si on dérive la vitesse on obtient l'accélération or celle-ci peut s'avérer très instable si la vitesse mesurée est trop bruitée. L'asservissement est alors fortement altéré et inutile.

Deux solutions sont alors envisageables :

- La première consiste à effectuer un filtrage à l'aide d'un filtre passe-bas afin de limiter le bruit
- La seconde et la plus simple est d'enlever le paramètre Dérivé de l'asservissement, on obtient donc un asservissement PI.

### **I.6. Algorithmes d'ajustement des paramètres de Contrôleur PID**

Régler un régulateur PID consiste à agir sur les 3 paramètres des différentes actions (gain du proportionnel, gain de l'intégral, gain de la dérivée) sur des valeurs optimales pour obtenir la réponse adéquate en précision, rapidité, stabilité et robustesse en sortie du procédé. Pour cela, il existe plusieurs méthodes de calcul des paramètres du régulateur PID. Elles sont basées sur les spécifications temporelles comme

- Méthode Ziegler–Nichols.
- Méthode de Chien-Hrones-Reswick.
- Méthode de Cohen-Coon.
- Méthode de Halman.

## I.7. Introduction au calcul fractionnaire

Le calcul fractionnaire est le champ de l'analyse mathématique, l'investigation et l'application des intégrales et des dérivées d'ordre arbitraire. Le calcul fractionnaire peut être considéré comme un sujet ancien et encore nouveau. Ces dernières années l'intérêt considérable pour le calcul fractionnaire a été stimulé par les applications de ce calcul dans les différents domaines de la physique et de l'ingénierie.

## I.8. Opérateurs d'ordre fractionnaire

Le calcul fractionnaire est une généralisation de l'intégration et de la différenciation à l'opérateur fondamental d'ordre non entier  ${}_{t_0}D_t^m$  où  $t_0$  et  $t_1$  sont des limites de l'opération. L'opérateur intégral-différentiel continu est défini par [6] :

$${}_{t_0}D_t^m = \begin{cases} \frac{d^m}{dt^m} & R(m) > 0, \\ 1 & R(m) = 0, \\ \int_{t_0}^t (d\tau)^{-m} & R(m) < 0, \end{cases} \quad (1.2)$$

Où  $m \in \mathbb{C}$  est l'ordre de l'opération ou  $\mathbb{C}$  anneau des nombres complexes.

Et  $R(\cdot)$  symbolise la partie réelle d'un nombre complexe.

Les trois définitions les plus fréquemment utilisées pour la généralisation de la dérivée et l'intégrale fractionnaire sont la définition de Grunwald Leitnikov, Riemann-Liouville, et la définition de Caputo.

### I.8.1 Définition de Riemann-Liouville (R-L)

La définition de Riemann-Liouville de la dérivée ou de l'intégrale d'ordre fractionnaire  $\alpha$  d'une  $f(t)$  est donnée par [6] :

$${}_a D_t^{-\alpha} f(t) = \{D^m [{}_a D_t^{-\alpha} f(x)], \quad \text{si } R(\alpha) > 0 \quad (1.3)$$

$$f(t), \text{ si } R(\alpha) = 0 \quad \frac{1}{\Gamma(-\alpha)} \int_a^t (t-x)^{-\alpha-1} f(x) dx, \quad \text{si } R(\alpha) < 0 \quad (1.4)$$

Où  $\alpha \in \mathbb{C}$ ,  $m < R(\alpha) < (m+1)$  et  $f(t)$  une fonction localement intégrable définie sur  $[a, \infty]$ .

### I.8.2 Définition de Caputo

Caputo a proposé une nouvelle définition de la dérivée d'ordre fractionnaire qui porte d'ailleurs son nom et qui incorpore les conditions initiales de la fonction à traiter en termes de ses dérivées d'ordre entier. La dérivée d'ordre fractionnaire  $\alpha > 0$  d'une fonction  $f(t)$  définie sur  $[a, \infty]$  donnée comme suit [7] :

$${}_a D_t^\alpha f(t) = \frac{1}{\Gamma(m-\alpha)} \int_a^t (t-\tau)^{m-\alpha-1} f^{(m)}(\tau) d\tau \quad (\text{I.6})$$

Où  $m$  est un entier tel que  $m < \alpha < (m+1)$  et  $f^{(m)}(t)$  ( $m$ ) est la  $m^{ieme}$  dérivée de la fonction  $f(t)$ .

### I.8.3 Définition de Grunwald-Leitnikov (G-L)

D'après la définition de Grünwald-Letnikov on peut définir la dérivée d'ordre fractionnaire par la relation [8][9] :

Supposons que :

$${}_{t_0} GL D_t^\alpha f(t) = \frac{1}{h^\alpha} \sum_{k=0}^{\lfloor \frac{t-t_0}{h} \rfloor} (-1)^k \binom{\alpha}{k} f(t-k.h) \quad (\text{I.7})$$

Où  $m > 0$ ,  $t_0 < t$ ,  $t_0, t \in R$  dénote la partie entière d'un nombre réel,  $h$  est la période d'échantillonnage et les coefficients  $\binom{\alpha}{k}$  sont donnés par :

$$\binom{\alpha}{k} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \quad (\text{I.8})$$

## I.9. Correcteurs D'Ordre Fractionnaire

### I.9.1 Correcteur d'ordre fractionnaire $PI^\alpha D^\mu$

le PID fractionnaire est proposé afin d'améliorer les performances des systèmes asservis linéaires, c'est une généralisation du correcteur PID classique, il a la forme  $PI^\alpha D^\mu$  nommé

où  $\mu$  et  $\alpha$  sont des réels positifs tel que :  $0 < \alpha < 1$ ,  $0 < \mu < 1$ . L'équation de sortie du correcteur  $PI^\alpha D^\mu$  d'ordre fractionnaire dans le domaine de temps est donnée sous la forme :

$$u(t) = k_p(e(t) + \frac{1}{T_I} D^{-\alpha}(e(t)) + T_D D^\mu e(t)) \quad (\text{I.9})$$

L'expression analytique du PID fractionnaire est donnée par l'équation suivante :

$$C(s) = k_p + \frac{k_I}{s^\alpha} + k_D s^\mu \quad (\text{I.10})$$

Avec :

- $k_p$  Représente l'action proportionnelle,
- $T_I$  La constante d'intégration,
- $T_D$  La constante de dérivation
- $k_D s^\mu$  Représente l'action dérivation d'ordre fractionnaire.

$$\text{Et } k_I = \frac{k_p}{T_I}, k_D = T_D k_p \quad (\text{I.11})$$

La Figure I.4 Présente la structure parallèle interne du  $PI^\alpha D^\mu$  d'ordre fractionnaire, définie par des connexions en parallèle entre les parties proportionnelles, intégrale d'ordre fractionnaire et dérivée d'ordre fractionnaire [10] :

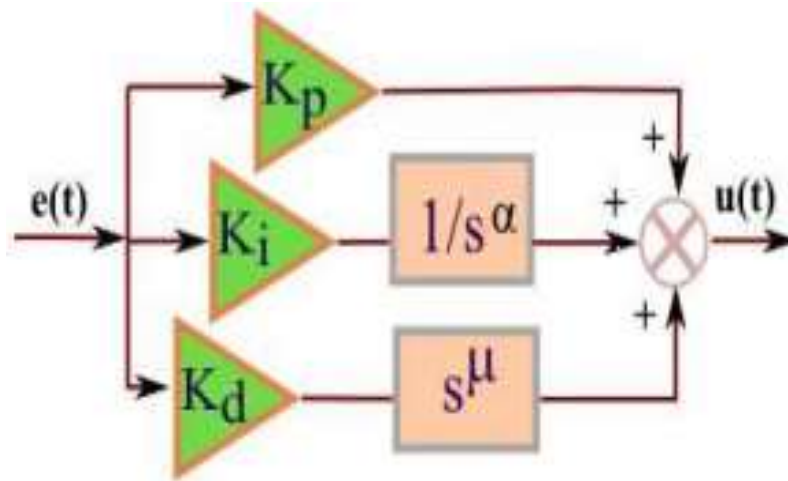


Figure I.4: Structure interne du  $PI^\alpha D^\mu$  d'ordre fractionnaire.

### I.10. Principe de fonctionnement

Par comparaison aux correcteurs classiques, les correcteurs d'ordre fractionnaire possèdent en plus deux autres paramètres notés  $\gamma$  et  $\mu$  et qui présentent l'ordre d'intégration et de

dérivation respectivement. Suivant la variation de ces deux paramètres, on peut distinguer différentes possibilités des correcteurs d'ordre fractionnaire.[11]

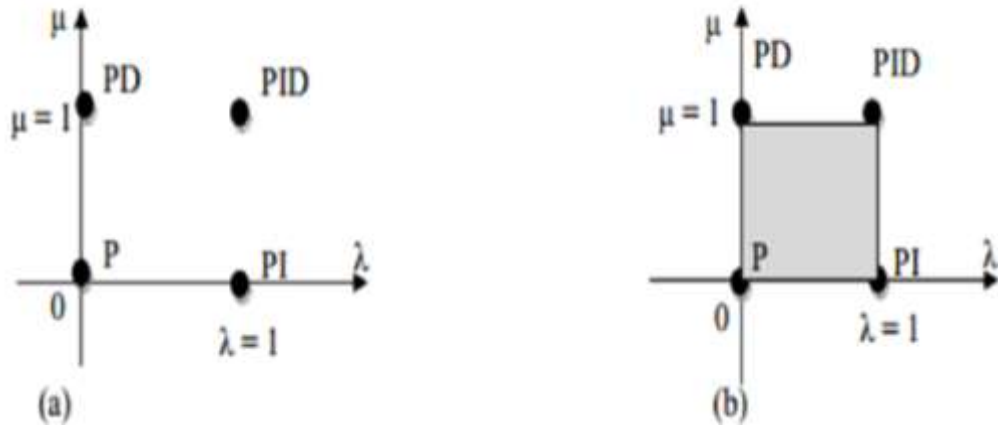


Figure I.5:  $PI^{\alpha}D^{\mu}$  (a) Ordre entier, (b) Ordre fractionnaire

- D'après cette figure, on peut donner les cas suivants :
- Lorsque  $\lambda = 0$  et  $\mu = 0$ , on a un correcteur **P** classique.
- Lorsque  $\lambda = 1$  et  $\mu = 0$ , on a un correcteur **PI** classique.
- Lorsque  $\lambda = 0$  et  $\mu = 1$ , on a un correcteur **PD** classique.
- Lorsque  $\lambda = 1$  et  $\mu = 1$ , on a un correcteur **PID** classique.
- Lorsque  $0 < \lambda < 1$  et  $\mu = 1$ , on a un correcteur **PI fractionnaire**.
- Lorsque  $\lambda = 0$  et  $0 < \mu < 1$ , on a un correcteur **PD fractionnaire**.
- Lorsque  $0 < \lambda < 1$  et  $0 < \mu < 1$ , on a un correcteur **PID fractionnaire**.

D'après ces résultats, on constate que les correcteurs classiques sont des cas particuliers des correcteurs d'ordre fractionnaires.

### I.11. Structure de correcteur PID fractionnaire

Le correcteur PID fractionnaire est implémenté dans des systèmes de commande à retour unitaire classique donné par la figure Fig.IV.1. Où  $u(t)$  désigne le signal de commande et  $e(t)$  l'écart résultant de la différence entre la consigne  $r(t)$  et la grandeur à commander  $y(t)$ ,  $C(s)$  est la fonction de transfert du correcteur fractionnaire,  $Gp(s)$  est la fonction de transfert de système, dans notre cas c'est le moteur à courant continu.



## I.12. La fractionalisation des Correcteurs PI et PID

La fractionalisation de PI classique ainsi que PID classique est obtenue en modifiant le terme intégrateur dans les fonctions de transferts des correcteurs précédents.

L'intégrateur  $1/s$  est fractionalisé comme suit [12, 13] :

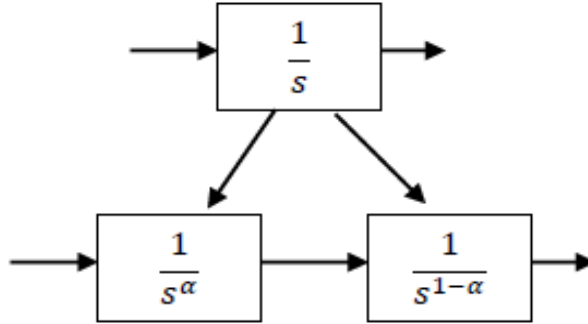


Figure I.6. Fractionalisation d'un intégrateur.

Tel que :

$$\frac{1}{s} = \frac{1}{s^\alpha s^{(1-\alpha)}} \quad (\text{I.12})$$

Avec  $\alpha$  est un nombre réel :  $0 < \alpha < 1$

La fonction de transfert d'un correcteur classique PI est :

$$C_{PI}(p) = K_p + \frac{1}{T_i p} \quad (\text{I.13})$$

La fonction de transfert d'un correcteur PI fractionalisé est donnée comme suite :

$$C_{PIf} = \frac{1}{p^\alpha p^{1-\alpha}} \left( \frac{k_p \tau_i p + 1}{\tau_i} \right) \quad (\text{I.14}) \quad \text{Avec } 0 < \alpha < 1$$

Soit la fonction de transfert d'un correcteur classique PID donné par la fonction suivante:

$$C(s) = K_p \left( 1 + \frac{1}{T_i s} + T_d s \right) \quad (\text{I.15})$$

Le correcteur PID fractionalisé est défini par la fonction suivante [26] :

$$C(s) = \frac{1}{s} \left( \frac{k_p T_d T_i s^2 + k_p T_i s + k_p}{T_i} \right) \quad (\text{I.16})$$

$$= \frac{1}{s} \frac{1}{s^{(1-\alpha)}} \left( \frac{k_p T_d T_i s^2 + k_p T_i s + k_p}{T_i} \right) \quad (\text{I.17})$$

Avec  $0 < \alpha < 1$

### I.13. Avantage du correcteur fractionnaire

Sa possibilité de bien commander la dynamique des systèmes d'ordre fractionnaire.

Il est moins sensible aux changements des paramètres d'un système commandé, ce qui donne une amélioration de la robustesse. Ceci est parce que les correcteurs  $PI^\alpha D^\mu$  d'ordre fractionnaire possèdent deux degrés de liberté supplémentaires pour mieux ajuster les propriétés dynamiques des systèmes de commande.

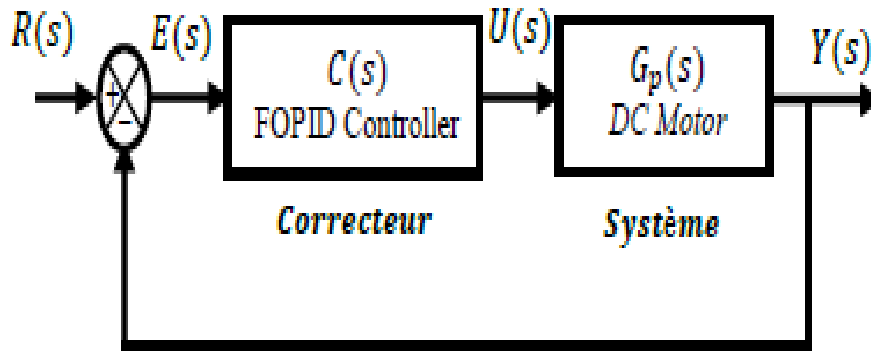


Figure I.7 Système de commande à retour unitaire classique

L'équation de sortie du correcteur P d'ordre fractionnaire dans le domaine de temps est donnée sous la forme :

$$u(t) = K_p [e(t) + \frac{1}{T_i} D^{-\lambda}(e(t)) + T_d D^\mu(e(t))] \quad (\text{I.18})$$

En appliquant la transformée de Laplace à l'équation (IV.3) avec les conditions initiales nulles, la fonction de transfert de ce correcteur peut être exprimé par :

$$C(s) = K_p + \frac{K_i}{s^\lambda} + K_d s^\mu \quad (\text{I.19})$$

Où les d'intégration **Ki** et de dérivation **Kd** sont liés aux paramètres de la forme classique par les relations suivantes :

$$K_i = \frac{K_p}{T_i} \quad (\text{I.20})$$

$$K_d = K_p K_d \quad (\text{I.21})$$

La fonction de transfère  $C(s)$  d'un correcteur est :

$$C(s) = K_p \left( 1 + \frac{1}{T_i s^\lambda} + T_d s^\mu \right) \quad (\text{I.22})$$

En plus de  $K_p$ ,  $K_i$ ,  $K_d$ , le correcteur  $PI^\alpha D^\mu$  possède deux autre paramètre de réglage  $\gamma$  et  $\mu$ . Ceci le rend plus flexible et donc une opportunité pour mieux ajuster les propriétés dynamiques des système de commande d'ordre fractionnaire. S'inspirant de l'idée du correcteur  $PI^\alpha D^\mu$  plusieurs travaux sur les techniques de réglage sont actuellement publiés. Pour plus de détail, se référer à [14]

#### **I.14. Conclusion**

Ce chapitre est une introduction aux éléments de base du calcul d'ordre fractionnaire. Nous avons présenté les concepts de base des opérateurs fractionnaires, les PID classique et les PID fractionnaire.

## **Chapitre II :**

# **Programmation par FLOWCODE/STM32**

## II.1. Introduction

Dans ce chapitre nous allons présenter des généralités sur la carte de développement STM32F103 et le logiciel FlowCode, ses différentes structures et les avantages et les inconvénients de son utilisation. Par la suite, nous exposons l'algorithme de contrôler de la vitesse d'un moteur à courant continu par un PID fractionnaire, dans le cas d'une commande vitesse/position de MCC.

## II.2. L'architecture du microcontrôleur STM STM32 F103RB [15]

### II.2.1 Définition

La carte STM32 Nucleo-64 offre aux utilisateurs un moyen abordable et flexible d'essayer de nouveaux concepts et de construire des prototypes en choisissant parmi les différentes combinaisons de fonctionnalités de performances et de consommation d'énergie, fournies par le microcontrôleur STM32. La carte STM32 Nucleo-64 est livrée avec les bibliothèques logicielles gratuites complètes STM32 et des exemples disponibles avec le package MCU STM32Cube.

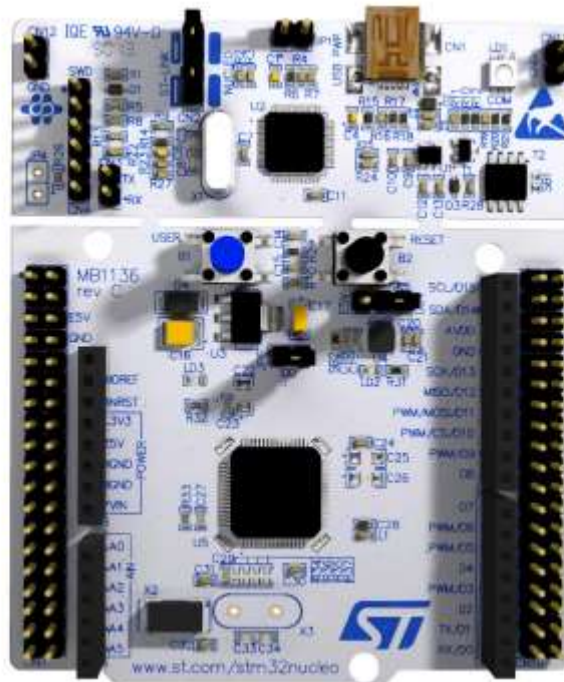


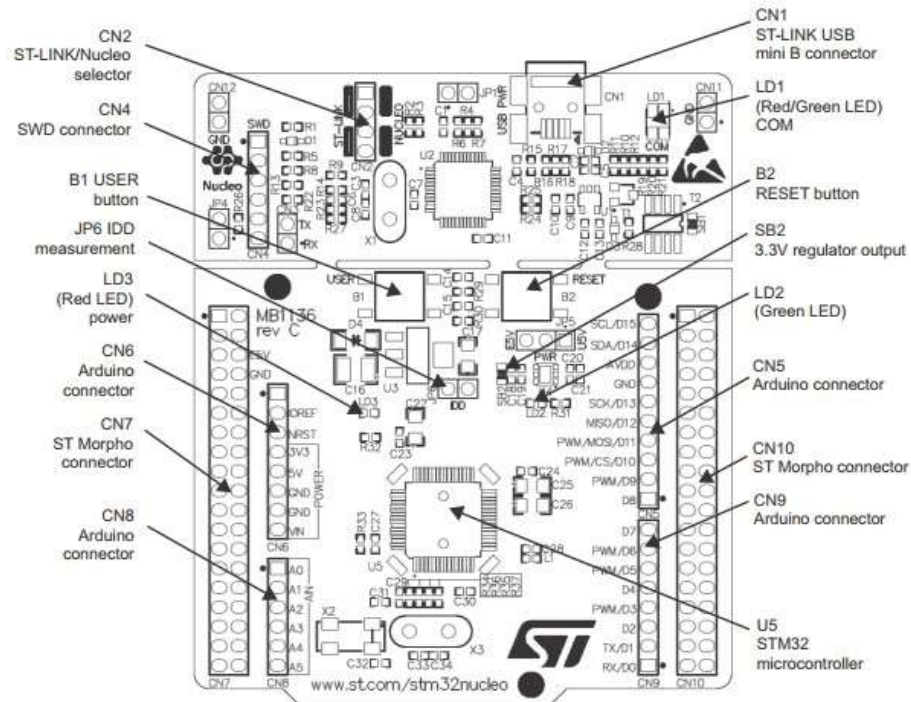
Figure II.1. STM32 Nucleo-64 board

## **II.2.2 L'intérêt de carte STM32 NUCLEO**

- Permet de faire une communication entre un ordinateur et une carte électronique et différents capteurs
- Les cartes STM32 Nucleo sont très abordables permettent à quiconque d'essayer de nouvelles idées et de créer rapidement des prototypes avec n'importe quel MCU STM32.

## **II.2.3 Description d'une carte NUCLEO-F103RB**

- Caractéristiques du processeur
  1. Coré : ARM 32-bit Cortex-M4 CPU
  2. Fréquence jusqu'à 72MHz,
  3. Multiplication à cycle unique et division matérielle
- Mémoire
  1. 64 ou 128 Ko de mémoire Flash
  2. 20 Kbytes de SRAM
- Oscillateur 4-à -16 MHz
- Tension d'alimentation de fonctionnement : 3.3 V, 5 V, 7 V to 12 V
- 2 convertisseurs A/N 12 bits, 1  $\mu$ s (jusqu'à 16 chaînes)
- Connecteurs de la carte :
  1. Connecteur d'extension ARDUINO® Uno V3
- Options d'alimentation flexibles : ST-LINK, USB VBUS ou sources externes
- Dimensions du NUCLEO-F103RB : 7cm  $\times$  8,25cm .



**Figure II.2.** Composants de la carte NUCLEO-F103RB

## II.2.4 L'environnement logiciel de développement « FLOWCODE » : [16]

Flowcode est un logiciel de programmation graphique permettant, à partir de la saisie d'algorigrammes, de créer des programmes pour les microcontrôleurs de la famille des PICmicro de Microchip et ARM Cortex-M et ARM Cortex-R ...ect. Une fois l'logigramme élaboré, Flowcode permet de simuler et visualiser le comportement du programme en déroulant.

Il est capable d'interpréter directement un algorithme et de générer un programme exécutable tout en gardant la possibilité de programmer des blocs en langage C ou directement en assembleur ce qui lui octroie une certaine puissance de traitement.



Figure II.3. L'interface FlowCode















-  Propriétés de l'icône Entrée
-  Propriétés de l'icône Sortie
-  Propriétés de l'icône Pause
-  Propriétés de l'icône Décision (alternative simple ou com)
-  Propriétés de l'icône Multi-décision (alternative généralis)
-  Propriétés de l'icône Point de jonction
-  Propriétés de l'icône Boucle (itération)
-  Propriétés de l'icône Macro (sous-programme)
-  Propriétés de l'icône Routine composant
-  Propriétés de l'icône Calcul
-  Propriétés de l'icône Manipulation de caractères
-  Propriétés de l'icône Interruption
-  Propriétés de l'icône Code C
-  Propriétés de l'icône Commentaire

Figure II.4 les icons de l'interface flow code



### **II.3. Programmation graphique [17]**

Flowcode permet le développement rapide et facile des systèmes électroniques et électromécaniques complexes. L'outil de programmation graphique permet à ceux qui ont peu d'expérience de développer des systèmes en quelques minutes.

Les icônes graphiques utilisées pour développer des systèmes dans Flowcode sont faciles à utiliser. Il permet aux développeurs débutants de prendre les fondamentaux et de fonctionner avec leurs conceptions. L'utilisation d'icônes graphiques permet aux utilisateurs de visualiser et d'apprendre le code côte à côte pour faciliter l'apprentissage.

#### **II.3.1 Les avantages**

- Programmation claire et simple
- Mise en œuvre simple et facile
- Vaste choix de modules d'E/S (Shields)

#### **II.3.2 Les inconvénients**

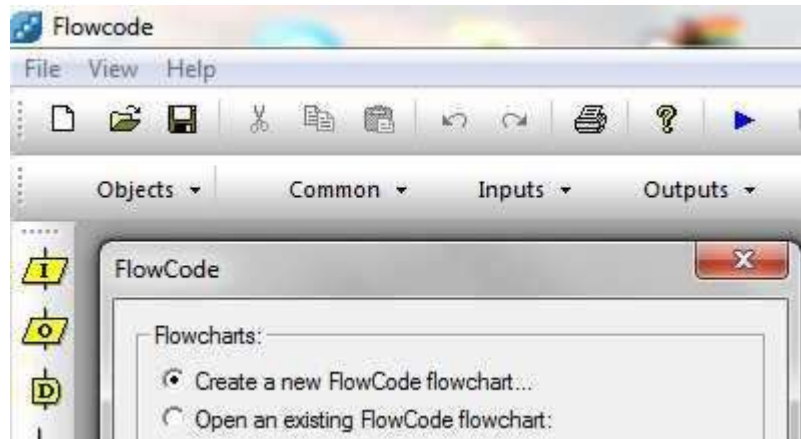
- Encombrement assemblage mécaniques peu commodes (pour les systèmes portables)
- Coût assez élevé.

L'élaboration d'un programme à l'aide de Flowcode passe par plusieurs étapes :

- La création du projet qui consiste à choisir et configurer le  $\mu$ contrôleur utilisé et les composants qui lui sont associés.
- L'édition du programme sous la forme d'un organigramme.
- La validation du programme par son exécution simulée, avec éventuellement une mise au point si le résultat n'est pas satisfaisant.
- La production du code machine destiné à être « gravé » dans la mémoire morte du circuit réel.

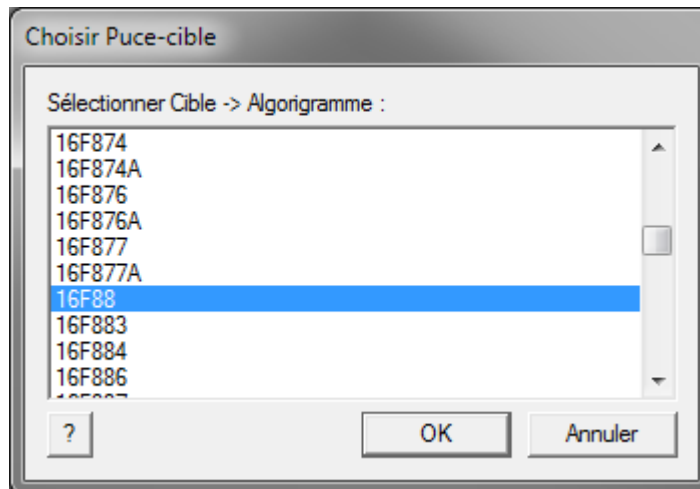
## II.4. Création d'un projet Flowcode

1. Ouvrir Flowcode, une fenêtre apparaît dans la fenêtre principale, permettant d'ouvrir un fichier existant ou d'en créer un nouveau.



**Figure II.5 :** creation d'un nouveau projet dans FlowCode

2. Choisir le microcontrôleur envisagé (par exemple STM32).



**Figure II.6 :** la choisisation du microcontrôleur

3. Configurer les propriétés du microcontrôleur utilisé :

- La fréquence de l'horloge du circuit.
- La vitesse de simulation à 1s par étape qui laisse le temps d'observer le déroulement. Pour une simulation plus réaliste, on choisit aussi vite que possible.

4. Création de variables : Les variables utilisées par flowcode sont des octets positifs uniquement.

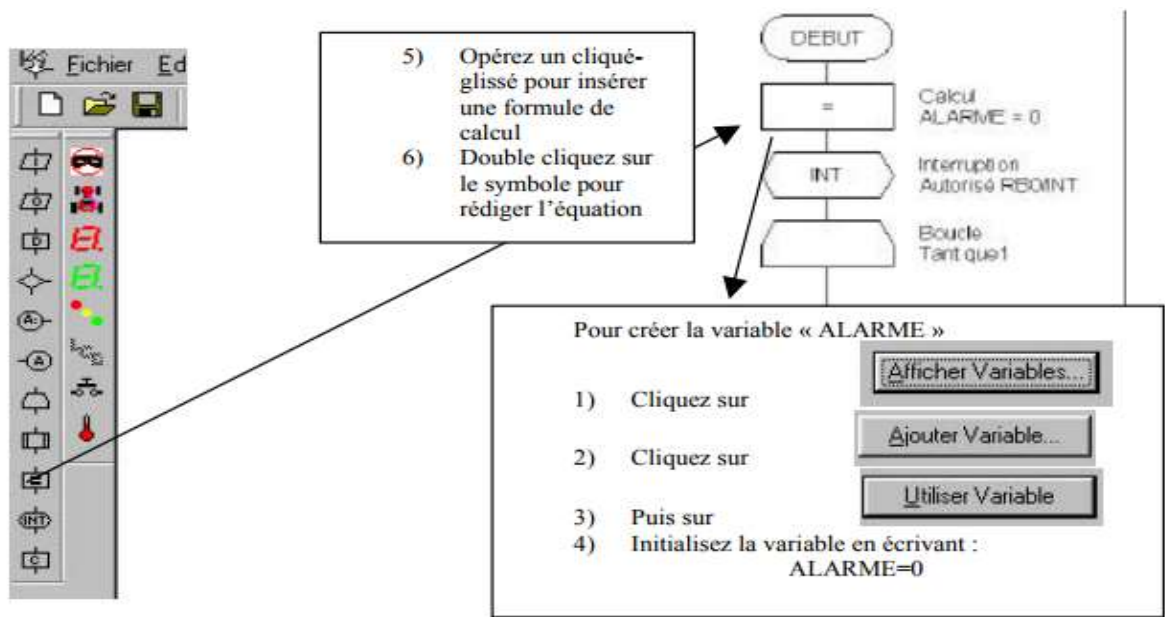


Figure II.7 : l'interface du creation ou declaration des variables sur flowcode

5. Création de l'algorithme : utilisation des entrées/sorties :

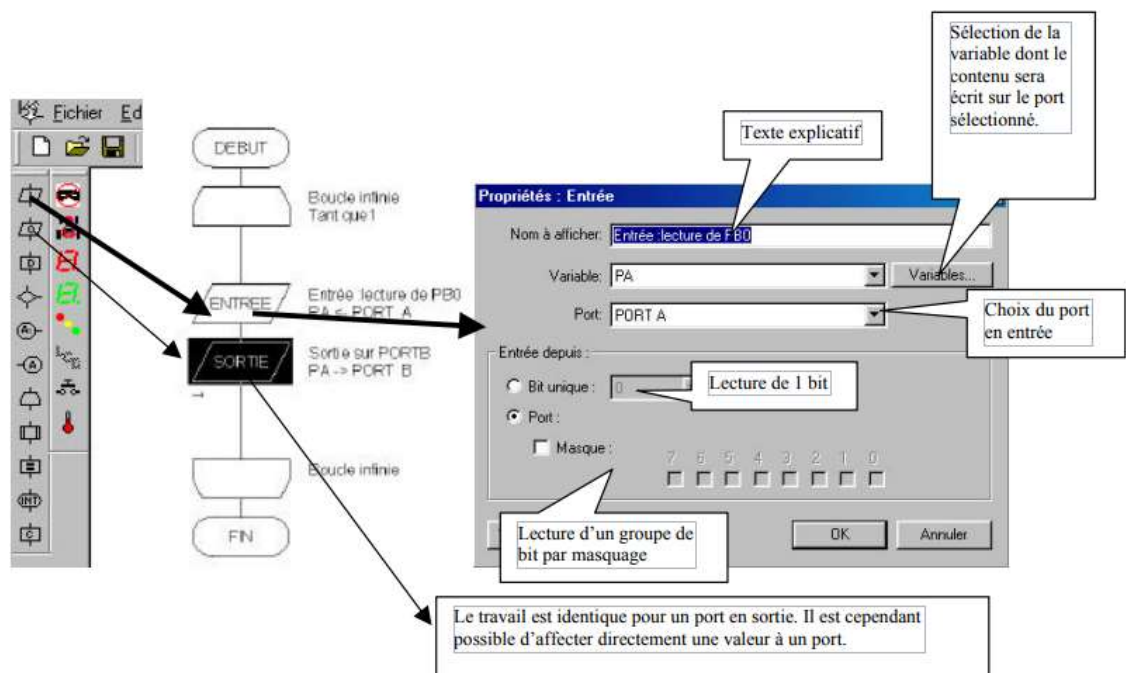


Figure II.8: l'interface du creation d'algorithme flowcode

## 6. Simulation de l'algorithme

a) Configuration de la vitesse de simulation Il faut d'abord configurer la vitesse de simulation dans le menu Exécuter / Vitesse d'horloge. Sélectionnez la vitesse de simulation « aussi vite que possible ».



Figure II.9 : l'interface de Configuration la vitesse d'horloge

b) Lancement/ suspension/ arrêt de la simulation

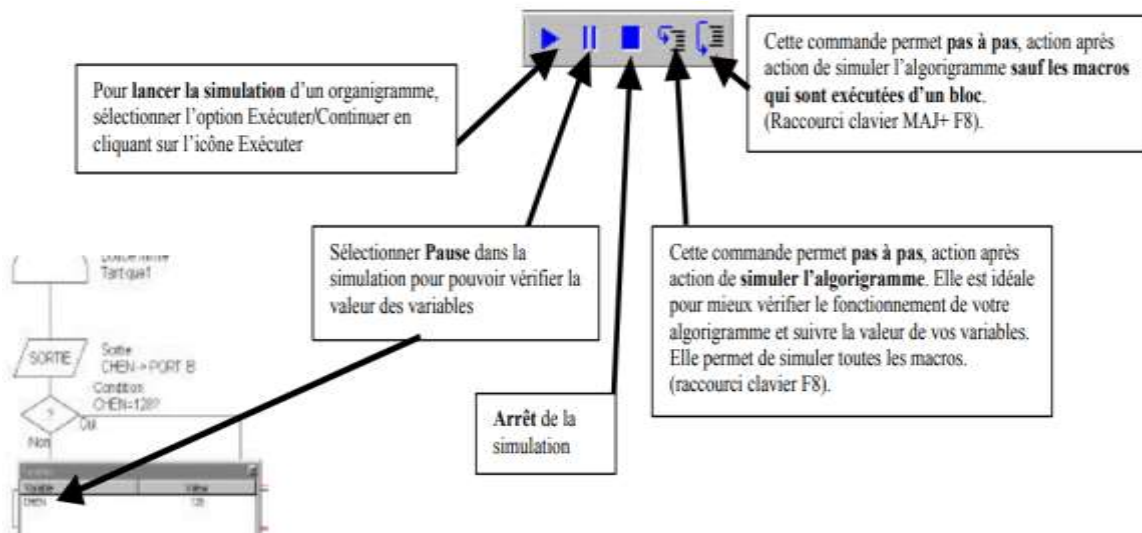


Figure II.10 : l'interface pour contrôler la simulation

## **II.5. Conclusion :**

Dans ce chapitre nous avons présenté les différentes caractéristiques de la carte de prototypage STM32 F103RB. Une partie de ce chapitre est consacré à la création d'un nouveau projet dans le logiciel FlowCode.

## **Chapitre III :**

# **Etude du correcteur fractionnaire et implémentation sous la carte STM32**

### III.1. Introduction

L'objectif de ce travail est l'étude et l'implémentation d'un correcteur PID fractionnaire qui sert à commander en vitesse un moteur à courant continu.

L'algorithme de correcteur fractionnaire sera implémenté sur la carte de prototypage STM32 en utilisant l'environnement de programmation graphique Flowcode.

Pour ce faire, nous allons suivre les étapes suivantes :

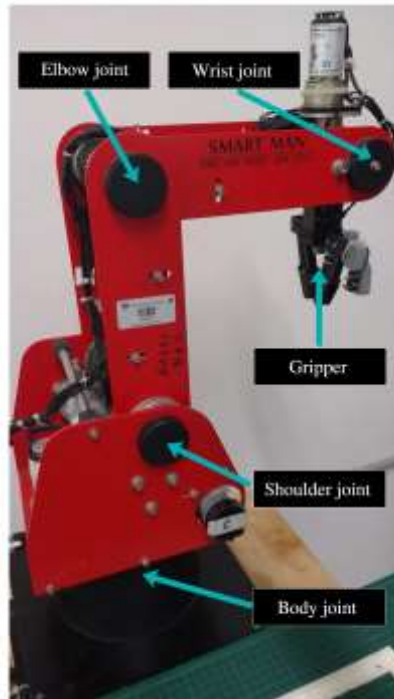
- 1) Modélisation mathématique du moteur à courant continu.
- 2) Conception de contrôleur d'ordre fractionnaire.
- 3) Élaboration de l'algorithme de correcteur fractionnaire en utilisant le toolbox ninteger de Matlab.
- 4) Implémentation de l'algorithme de commande sur une carte STM32.
- 5) Test et évaluations.

Le problème majeur dans l'implémentation et la réalisation d'une chaîne d'asservissement et la partie identification des paramètres du moteur. Pour régler ce problème, nous avons choisi un moteur à courant continu muni d'un encodeur optique dont on connaît ces paramètres [18]. Il est monté sur un bras manipulateur robotisé disponible au niveau des laboratoires pédagogiques du département d'électronique. La figure 1, montre le bras manipulateur.

Ce bras manipulateur est équipé de 6 moteurs à courant continu dont l'identification des paramètres a été faite par les auteurs de [19].

**Tableau III.1** montre les paramètres de chaque moteur

Actuated joint	Estimate $\hat{a}_i = [s^{-1}]$	Estimate $\hat{b}_i = [V^{-1}s^{-2}]$
Waist	$\hat{a}_1 = 10.2$	$\hat{b}_1 = 1.53$
Shoulder	$\hat{a}_2 = 13.2$	$\hat{b}_2 = 1.10$
Elbow	$\hat{a}_3 = 05.5$	$\hat{b}_3 = 0.88$
Wrist	$\hat{a}_4 = 17.$	$\hat{b}_4 = 0.88$

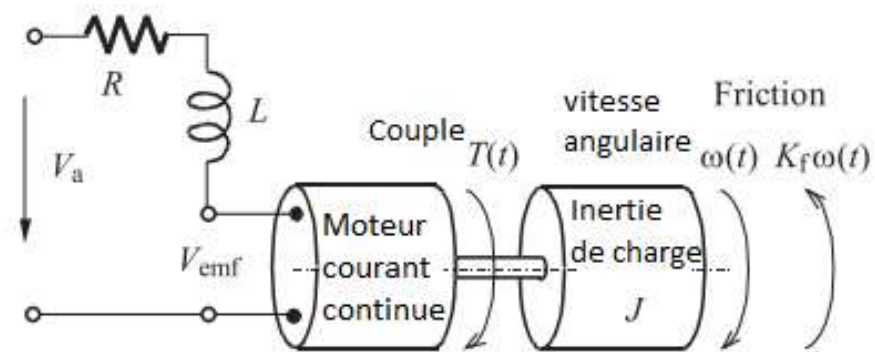


**Figure III.1** Bras manipulateur

### III.2. Modélisation mathématique du moteur à courant continu [20] :

Le moteur à courant continu est un actionneur de puissance, qui convertit l'énergie électrique en rotation mécanique. Les moteurs à courant continu sont souvent utilisés dans l'industrie dans de nombreuses applications de contrôle, les bras manipulateurs robotisés et dans les applications commerciales telles que lecteur de disque.

Soit le modèle général du Moteur à Courant continu (MCC) représenté sur la Figure III.1.

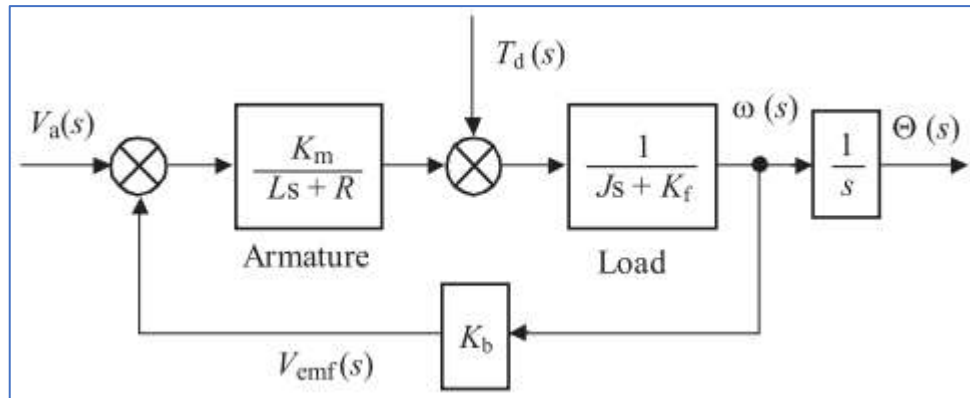


**Figure III.2** modèle général d'un moteur à courant continu



Dans cette situation, la tension  $V_\alpha$  Appliquer à l'entrée commande la vitesse angulaire  $\omega(t)$ .

Le schéma bloc de la figure III.2 représente les relations entre les différents éléments du moteur à courant continu.



**Figure III.3** modèle mathématique d'un moteur à courant continu .

La fonction de transfert de ce système (avec  $T_d(s) = 0$ ) est donnée par l'équation III.1:

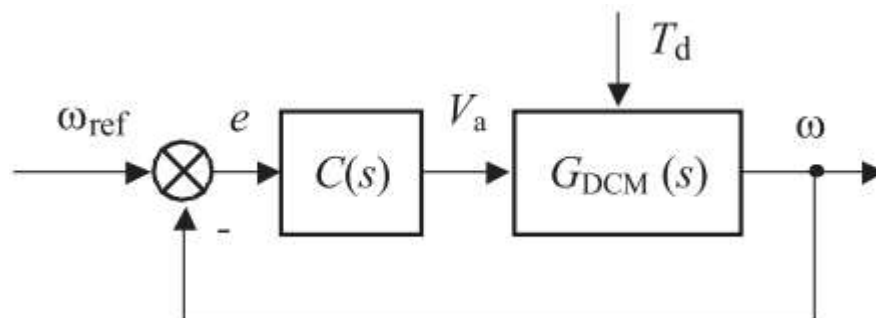
$$G_{DCM}(s) = \frac{\theta(s)}{V_\alpha(s)} = \frac{K_m}{s[R(Js+K_f)+K_bK_m]} = \frac{K_{DCM}}{s(\tau s+1)} \quad (\text{III.1})$$

Avec  $\tau = \frac{RJ}{RK_f+K_bK_m}$  et  $K_{DCM} = K_m/(RK_f + K_bK_m)$  et  $K_b = K_m$ .

Modélisation mathématique des différents moteurs

**III.3. Cconception de contrôleur d'ordre fractionnaire [21] :**

Dans **Figure III.3** représente la boucle d'asservissement à retour unitaire du système, où  $C(s)$  est la fonction de transfert du contrôleur et  $G_{DCM}(s)$  la fonction du transfert du moteur à courant continu.



**Figure III.4** Boucle ferme de commande

Nous voulons concevoir le correcteur, qui nous donnera une réponse échelonnée de la boucle de commande à rétroaction avec un dépassement indépendant des changements de la charge utile (iso-amortissement).

Dans le domaine fréquentiel, cela signifie une marge de phase indépendante des changements de la charge utile.

La marge de phase du système est donnée par [22], [23] :

$$\Phi_m = \arg[C(j\omega_g)G_{DCM}(j\omega_g)] + \pi \quad (\text{III.7})$$

Où  $j\omega_g$  est la fréquence de coupure. Une marge phase indépendante signifie en d'autres termes une phase constante. Ceci peut être accompli par un correcteur de la forme :

$$C(s) = k_1 \frac{k_2 s + 1}{s^\mu}, \quad k_1 = 1/k_{DCM}, \quad k_2 = \tau \quad (\text{III.8})$$

Un tel correcteur donne une marge de phase constante et la marge de phase dans ce cas est donnée par :

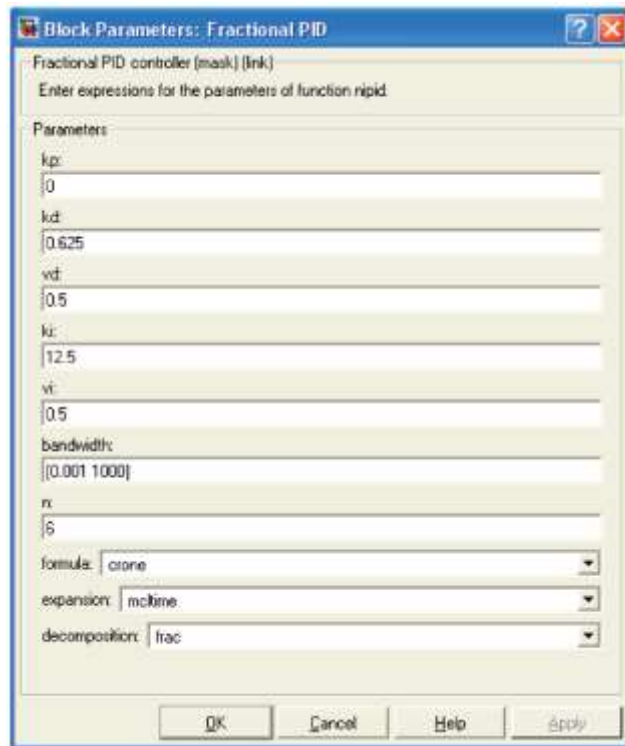
$$\Phi_m = \arg[C(j\omega_g)G_{DCM}(j\omega_g)] + \pi = \arg[C(j\omega)^{-1-\mu}] + \pi = \pi - (1 + \mu) \frac{\pi}{2} \quad (\text{III.9})$$

#### **III.4. Algorithme de correcteur fractionnaire en utilisant le toolbox Matlab ninteger.**

Afin d'implémenter le correcteur fractionnaire, nous allons utiliser le toolbox ninteger, ce toolbox comporte des routines Matlab qui simplifient l'implémentation des algorithmes basés sur le calcul d'ordre fractionnaire.

Plusieurs techniques et méthodes d'approximation sont disponibles telles que les méthodes de Carlson, de Matsuda et de Crone.

Ce toolbox permet aussi de développer des projets sous Simulink grâce à son bloc npid.



**Figure III.5** interface graphique du bloc npid du toolbox ninteger

Comme il est montré sur la figure 1, le bloc npid permet de configurer les paramètres du correcteur fractionnaire tel que, les coefficients  $k_p$ ,  $k_d$  et  $k_i$ , la bande de fréquence, la méthode d'approximation et le nombre de coefficients d'approximation.

### **III.5. Implémentation de l'algorithme de commande sur une carte STM32.**

Une fois les paramètres du contrôleur sont calculés, ils seront utilisés pour implémenter l'algorithme du correcteur fractionnaire. Pour ce faire, nous allons exploiter l'algorithme du correcteur fractionnaire développé par Petras [24], cet algorithme est composé des étapes suivantes :

*Initialisation des paramètres du correcteur fractionnaire*

*Code cyclique*

*Lecture et mise à l'échelle du signal de commande*

*Calcul de la sortie de la chaîne directe*

*Calcul de la sortie de chaîne de retour*

*Mise à jour des coefficients*

*Fin du code cyclique*

*Commander le moteur*

### III.6. Résultats et Simulation

#### III.6.1 Modélisation mathématique du moteur à courant continu [25] :

Nous avons choisi de modéliser le moteur de l'épaule 'Shoulder', La fonction de transfert de moteur est obtenue en remplaçons les valeurs (pour  $K_{DCM} = 1.10$  et  $\tau = 13.1$ ) dans l'équation (3.1).

Nous obtenons la fonction de transfert en boucle ouverte du moteur suivant :

$$G_{DCM}(s) = \frac{1.10}{s(13.1s+1)} \quad (\text{III.2})$$

L'équation de la fonction de transfert discrète du moteur en boucle ouverte pour ( $T=0.001$  et  $a=1/3$ ) a la forme suivante :

$$G_{DCM}(Z^{-1}) = \frac{1.10+0.733*Z^{-1}+0.122*Z^{-2}}{232.89*10^4-465.78*10^4*Z^{-1}+232.88*10^4*Z^{-2}} \quad (\text{III.4})$$

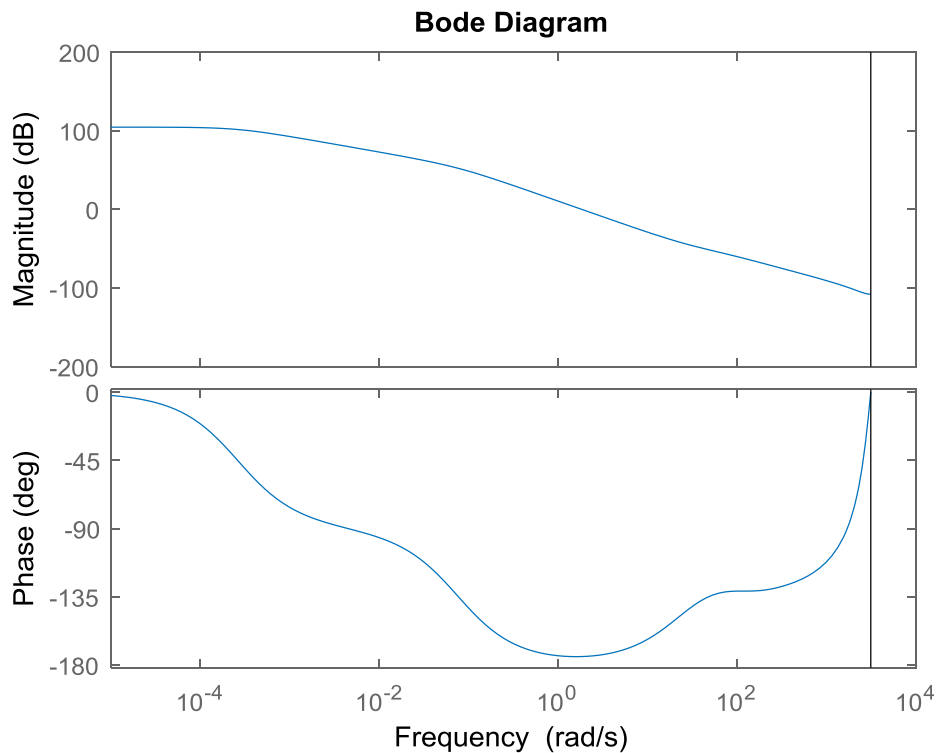


Figure III.6 Diagramme de Bode de la fonction de transfert du moteur

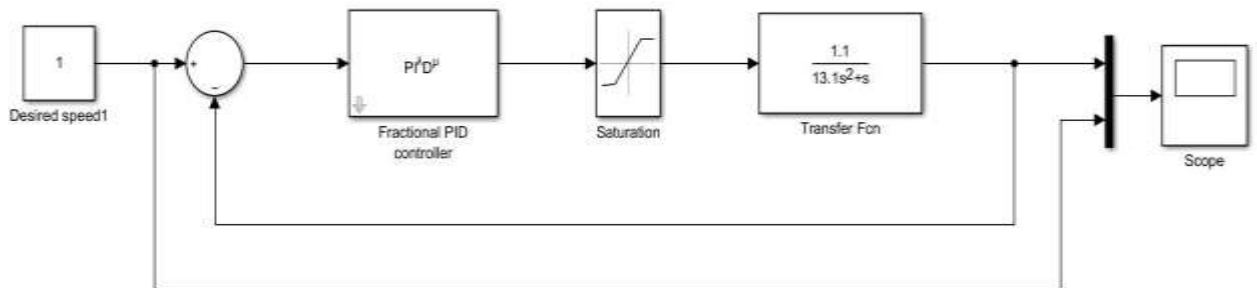
La fonction de transfert du moteur en boucle fermée est donnée par :

$$G_C(s) = \frac{G_0(s)}{1+G_0(s)} = \frac{G_{DCM}(s)C(s)}{1+G_{DCM}(s)C(s)} = \frac{13.1s^1+1}{13.1s^{2.5}+s^{1.5}+13.1s^1+1} \quad (\text{III.11})$$

$$G_0(s) = \frac{13.1s^1+1}{13.1s^{2.5}+s^{1.5}} \quad (\text{III.12})$$

La simulation de la boucle de la chaîne d'asservissement peut également être faite dans l'environnement Matlab/Simulink.

Le modèle général de Simulink est illustré sur la **Figure III.7**.



**Figure III.7** Modèle Simulink pour le contrôle de rétroaction du moteur à courant continu.

Les résultats de simulation dans le domaine temporel sont illustrés à la **Figure III.8 -9**.



**Figure III.8** la réponse échelon unitaire d'une boucle de commande sans saturation .



**Figure III.9** la réponse échelon unitaire d'une boucle de commande avec actionneur saturation .

### III.6.2 Conception de contrôleur d'ordre fractionnaire

On appliquant le toolbox ninteger et en choisissons un contrôleur  $PI^\lambda$  ( un cas particulier de le  $PI^\lambda D^\mu$  ) nous obtenons la fonction de transfert de notre correcteur qui a la forme :

$$C(s) = \frac{\tau}{k_{DCM}} s^{0.5} + \frac{1}{k_{DCM} s^{0.5}} = k_d s^{0.5} + k_i s^{-0.5} = 11.9\sqrt{s} + \frac{0.909}{\sqrt{s}} \quad (\text{III.10})$$

Où  $\mathbf{K_i} = 0.909$ ,  $\mathbf{K_d} = 11.9$  et  $\lambda = 0,5$ .

Les paramètres des correcteurs fractionnaires que nous avons trouvés sont utilisés pour la conception du correcteur fractionnaire fractionnaire , la figure 1 montre l'interface graphique du toolbox npid ainsi que les coefficients utilisés.

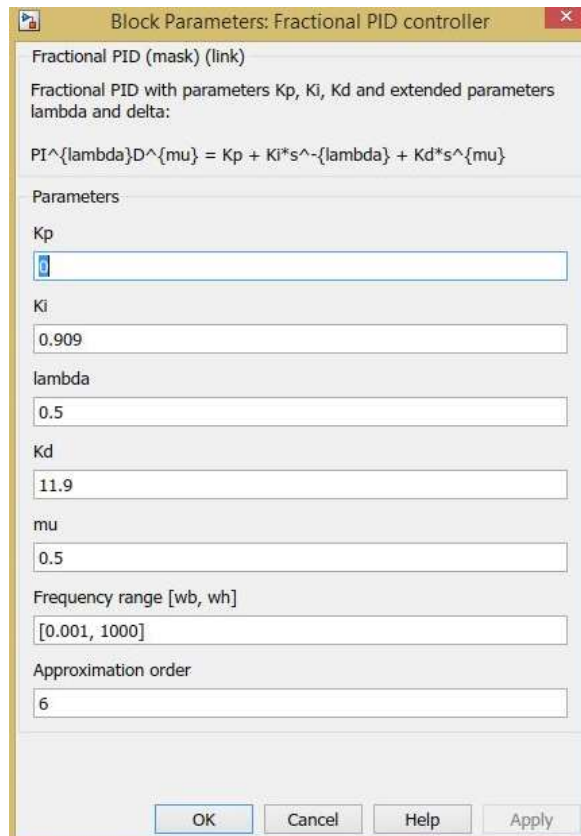


Figure III.10 l'interface graphique du toolbox npid ninteger.

### III.6.3 Implémentation de l'algorithme de commande sur une carte STM32.

Afin d'implémenter l'algorithme du correcteur fractionnaire que nous avons élaboré, nous allons convertir le script développé par Petras [26] par le langage graphique Flowcode.

Le scripte de Petras est le suivant :

```
(* initialization code *)
scale := 32752; % input and output
order := 6; % order of approximation
U FOC := 5; % input and output voltage range: 5[V], 10[V],
...
a[0] := 1.0; a[1] := -2.0; a[2] := 1.11; a[3] := 0.0;
a[4] := -0.111; a[5] := 0.0082; a[6] := 0.0014;
b[0] := 23.17; b[1] := -61.33; b[2] := 55.87; b[3] := -18.52;
b[4] := 0.268; b[5] := 0.560; b[6] := 0.032;
loop i := 0 to order do
s[i] := 0;
endloop
(* cyclic code *)
in := (REAL(input)/scale) * U FOC;
feedback := 0; feedforward := 0;
loop i:=1 to order do
feedback := feedback - a[i] * s[i];
feedforward := feedforward + b[i] * s[i];
endloop
s[0] := in + a[0] * feedback;
```

```

out := b[0] * s[1] + feedforward;
loop i := order downto 1 do
s[i] := s[i-1];
endloop output := INT(out*scale)/U FOC;

```

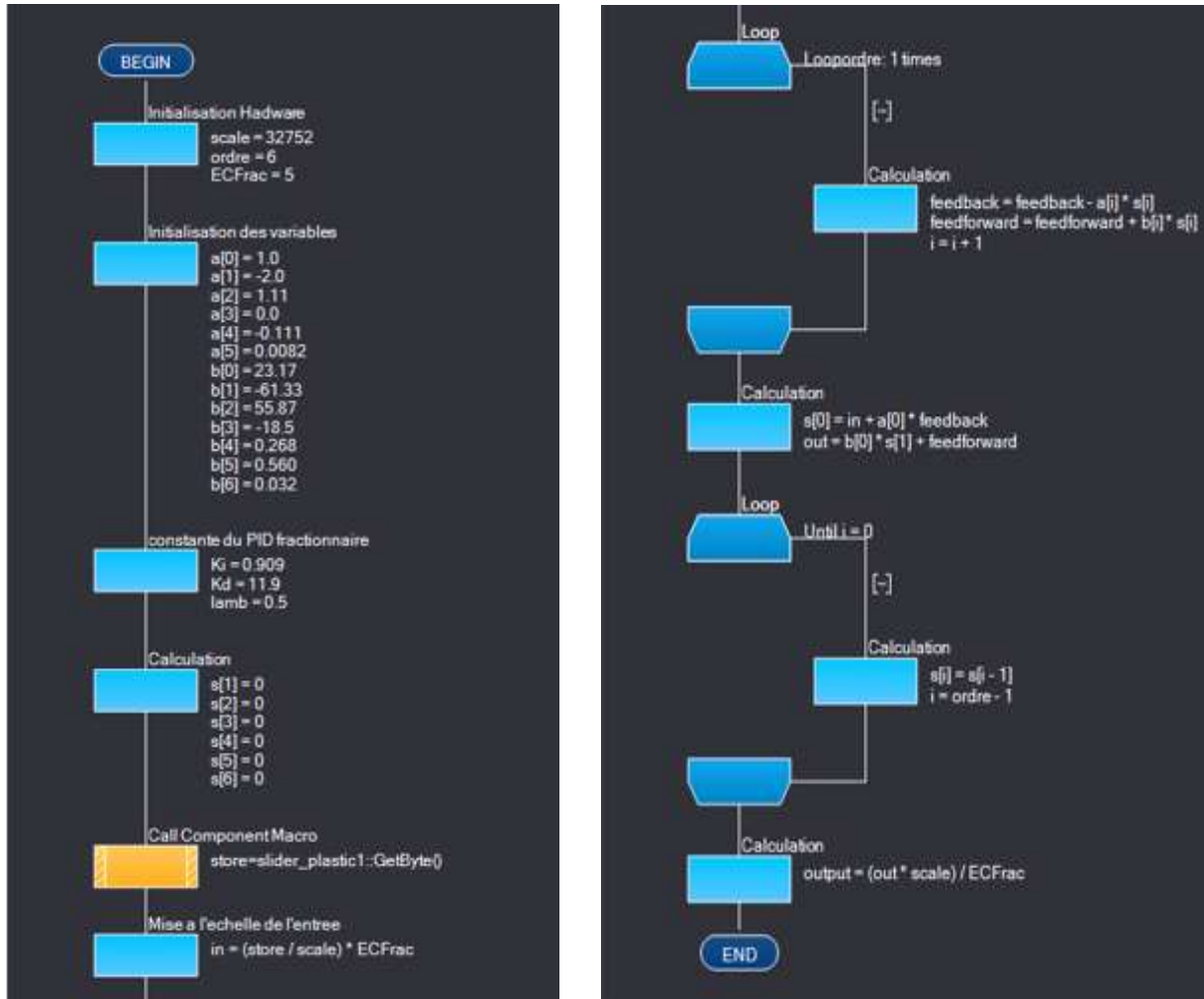


Figure III.11 Le code programme Flowcode du correcteur fractionnaire



### **III.7. Conclusion**

Dans ce chapitre nous avons étudié et implémenté un correcteur fractionnaire sur la carte de prototypage STM32 et cela en utilisant le logiciel de programmation graphique Flowcode.

Le résultat de simulation montre que le correcteur fractionnaire présente des améliorations par rapport à la commande classique.

Différentes courbes et simulations ont été élaborées pour montrer l'utilité et l'efficacité du calcul fractionnaire dans la conception et l'implémentation des correcteurs fractionnaires.

# Conclusion générale

Dans ce mémoire nous avons étudié et implémenté un correcteur d'ordre fractionnaire afin de commander en vitesse un moteur à courant continu monté sur un bras manipulateur robotisé disponible aux laboratoires pédagogiques du département d'électronique.

Pour implémenter cette commande sur une carte de prototypage STM32 en utilisant le logiciel Flowcode, nous avons suivi trois étapes, tout d'abord une étape de modélisation de la fonction de transfert du moteur à courant continu en se basant sur les paramètres d'identification.

La deuxième étape a été l'utilisation du toolbox ninteger pour la conception et le calcul des paramètres du correcteur fractionnaire et enfin une dernière étape qui est l'implémentation du correcteur en utilisant le logiciel Flowcode.

Les résultats de simulation ainsi que la compilation du code montrent que le correcteur PID fractionnaire représente une alternative au PID classique.

La généralisation de la commande sur l'ensemble des moteurs du robot manipulateur et la réalisation d'une commande globale du robot sera la prochaine étape de ce projet.

# Bibliographique

- [1] Yassine BENSALIA, ,,Utilisation des filtres fractionnaires pour la conception de régulateurs adaptatifs robustes", Mémoire de Doctorat en Science Option: Automatique, Université du 20 Août 1955-Skikda , 2016/2017.
- [2] Ghania Boukerche, ,,Etude et Synthèse d'un Contrôleur PI et Application ", Mémoire de Magister Option: Commande des Systèmes industriels, Université de BADJI MOKHTAR 2017
- [3] Bensaoula Mohamed Amine, ,,Conception du Contrôleur PID pour le Moteur à Courant Continu (MCC) à Excitation indépendante Simulation sous Matlab/Simulink", Mémoire de Master Option: Electromécanique, Université de BADJI MOKHTAR 2019
- [4] Jean-Marie Flaus, La régulation industrielle; ,,régulateur PID, Prédictifs et flous" Edition HERMES, Paris 1994.
- [5] Alina BESANCON-VODA & Sylviane GENTIL, ,,Régulateur PID analogiques et numérique"s. Edition Technique d'ingénieur 2012.
- [6] S. Das. Functional fractional calculus. Springer Verlag, 2011.
- [7] Podlubny, I., 'Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications', Academic Press, San Diego, 1999
- [8] Tellab, B. Résolution des équations différentielles fractionnaires. Thèse de doctorat, Université des Frères Mentouri -Constantine -1, Algérie, 2018.
- [9] KORICHI, Z et MEFTAH, T. Etude de systèmes statistiques dans diverses dimensions d'espace basée sur une mécanique quantique fractionnaire. Thèse de doctorat, Université de Kasdi Merbah -Ourgla, Algérie, 2016.
- [10] Petráš, Ivo. Fractional-order nonlinear systems: modeling, analysis and simulation. Springer Science & Business Media, 2011
- [11] Podlubny I. ,,Fractional Order Systems and PI $\lambda$ D $\mu$  Controllers," IEEE Transactions on Automatic Control, Vol. 44, No. 1, pp. 208-214, 1999.
- [12] Bensafia Y., Ladaci S., Khettab K. ,,Performance Analysis of Fractionalized Order PID Controller with the Conventional PID Controller," International Conference on Control, Engineering & Information Technology, CEIT'14 22-25 March 2014, Sousse, Tunisia, 2014.
- [13] Ladaci S., Bensafia Y. ,,Fractionalization: A new tool for robust adaptive control of noisy plants," 6th IFAC Workshop on Fractional Differentiation and Its Applications, FDA'13, Grenoble, France, pp. 379-384, February 4-6, 2013.

- [14] Bagley R.L., Torvik P.J. ,,,, *On the appearance of the fractional derivatives in the behavior of real materials*, "" J. Applied Mechanics, Vol. 41, pp. 294–298, 1984.
- [15] <https://www.st.com/resource/en/datasheet/stm32f103rb.pdf>
- [16] <https://docplayer.fr/storage/75/72286580/72286580.pdf>
- [17] <https://www.festo-didactic.com/ch-fr/systemes-d-apprentissage/flowcode-8-un-environnement-de-programmation-visuel.htm?fbid=Y2guZnIuNTM5LjE2LjE4LjAuMTAyOTQ1>
- [18] Kopacek, B.; Kopacek, P. End of life management of industrial robots. *Elektrotech. Informationstech.* **2013**, 130, 67–71.
- [19] Iqbal, J.; Ullah, M.I.; Khan, A.A.; Irfan, M. Towards sophisticated control of robotic manipulators: An experimental study on a pseudo-industrial arm. *Strojniški Vestnik J. Mech. Eng.* **2015**, 61, 465–470.
- [20] XUE, D.—ZHAO, C.—CHEN, Y. Q. Fractional Order PID Control of a DC-Motor with Elastic Shaft: A Case Study, Proc. of the 2006 American Control Conference, Minneapolis, Minnesota, USA, June 14-16, 2006, pp. 3182–3187.
- [21] [38] PODLUBNY, I. Fractional-Order Systems and PI<sub>Dμ</sub>-Controllers, *IEEE Transactions on Automatic Control* 44 No. 1 (1999), 208–214.
- [22] BIDAN, P. : Commande diffusive d'une machine électrique : une introduction, *ESAIM Proceedings*, vol. 5, France, 1998, pp. 55–68.
- [23] VINAGRE, B. M.—PODLUBNY, I.—DORCÁ K, LFELIU, V.: On Fractional PID Controllers: A Frequency Domain Approach, Proc. of the IFAC Workshop on Digital Control – PID'00, Terrassa, Spain, 2000, pp. 53–55.
- [24] Al-ALAOUI, M. A.: Filling the Gap between the Bilinear and the Backward Difference Transforms: An Interactive Design Approach, *Int. J. Elect. Eng. Edu.* 34 No. 4 (1997), 331-337.
- [25] DORF, R. C.—BISHOP, R. H.: *Modern Control Systems*, Addison-Wesley, New York, 1990.
- [26] PETRAS, I.—GREGA, S. Digital Fractional Order Controllers Realized by PIC Microprocessor: Experimental Results, Proceedings of the ICC2003, High Tatras, Slovak Republic, May 26-29, pp. 873–876.

## Résumé :

Dans ce mémoire nous avons étudié et implémenter un correcteur d'ordre fractionnaire dont le but de commander en vitesse un moteur à courant.

Pour implémenter cette commande sur une carte de prototypage STM32, nous avons suivi trois étapes. Une étape de modélisation de la fonction de transfert du moteur à courant continu. Une étape de conception du correcteur fractionnaire en utilisant toolbox Matlab ninteger et enfin une dernière étape d'implémentation du correcteur en utilisant le logiciel Flowcode.

Les résultats de simulation ainsi que la compilation du code montrent que le correcteur PID fractionnaire représente une alternative au PID classique.

## Abstract:

In this work, we have studied and implemented a fractional order controller in order to control the speed of DC motor.

To implement this controller on an STM32 board, we followed three steps. First, modeling the transfer function of the DC motor. Secondly, design of fractional controller using Matlab ninteger toolbox and finally implementation using Flowcode software.

The simulation results as well as the compilation of the code show that the fractional PID corrector represents an alternative to the classic PID.

## ملخص:

في هذا العمل، قمنا بدراسة وتنفيذ وحدة تحكم الترتيب الجزئي من أجل التحكم في سرعة محرك التيار المستمر. لتنفيذ وحدة التحكم هذه على لوحة STM 32 ، اتبعنا ثلاث خطوات. أولاً، نمذجة وظيفة النقل لمحرك DC . ثانياً ، تصميم وحدة تحكم كسرية باستخدام صندوق أدوات Matlab ninteger وأخيراً التنفيذ باستخدام برنامج Flowcode تظهر نتائج المحاكاة بالإضافة إلى تجميع الكود أن مصحح PID الجزئي يمثل بديلاً لـ PID الكلاسيكي.