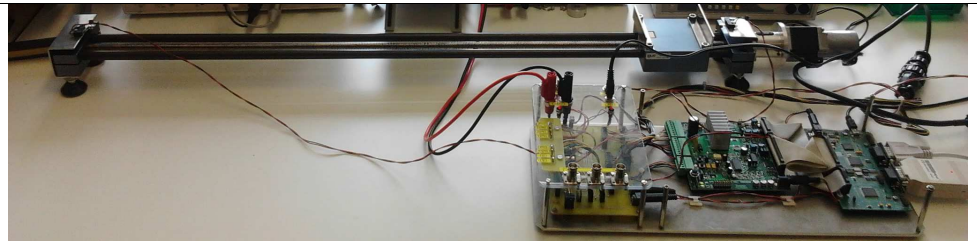
IPS – Commande Vectorielle d'un Moteur Synchrone (PMSM)



IPS - PMSM - Objectifs

*DR = Document Réponse

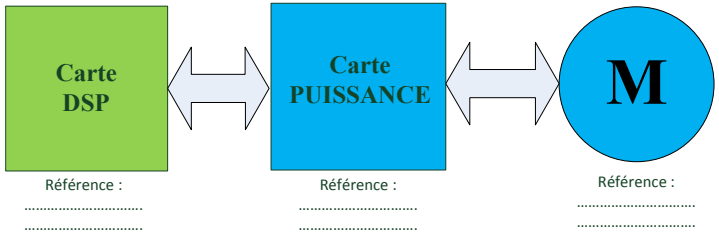
			Validation	sur
PARTIE 1	**Prise en Main**			
	QUEST			
			DR1	**1**
			MODELISATION_MOTEUR	**1**
PARTIE 2	**PWM - ADC**			
	QUEST			
			DR2	**2**
	APPLI			
		2. Simulation	**DR3**	**1**
PARTIE 3	**Asservissement de Vitesse**			
	APPLI			
		1. Simulation	**DR4**	**1**
PARTIE 4	**Asservissement de Vitesse**			
	APPLI		**Démo + Contexte**	**4**
PARTIE 5	**Table Linéaire**			
	APPLI 1		**Démo + Schéma Simulink**	**1**
	APPLI2		**Démo + Schéma Simulink**	**2**
	APPLI 3		**Démo + Schéma Simulink**	**2**
	APPLI 4		**Démo + Schéma Simulink**	**2**
	APPLI 5		**Démo + Schéma Simulink**	**1**
	APPLI 6		**Démo + Schéma Simulink**	**2**
			TOTAL	**20**

PARTIE 1	Prise en Main du Système	Durée : 1.5 UC
	OBJECTIFS	• Faire le tri parmi les docs constructeurs • Revoir les fonctions de Matlab-Simulink

QUESTIONNAIRE	Etude de Documents	1UC

REMARQUE : La plupart des réponses aux questions suivantes se trouvent dans le dossier ARCHITECTURE.

Q1. Compléter le Schéma du système suivant :



Q2. Donner la fréquence F_CPU de fonctionnement du DSP :

REMARQUE : Cette fréquence correspond à la fréquence d'horloge des différents périphériques de la carte.

Q3. Le DSP utilisé possède-t-il une unité de calcul en virgule flottante ?

Q4. Dessiner un Hacheur 4 quadrants

Q5. Dessiner un onduleur pour moteur triphasé

Q6. Peut-on contrôler un moteur à courant continu avec cette carte de puissance (justifier) ?

Q7. Quelle est la plage de vitesse du moteur ?

PMSM

Compléter le fichier Tableur **MODELISATION_MOTEUR**

Grandeur	Notation	Valeur	Unité
Puissance	P		W
Tension phase-neutre	V		V
Grandeurs E.D.C.M. (EQUIVALENT DIRECT CURRENT MOTOR) (EQUIVALENT MCC)			
winding current	Ir	7.4	A
cont stall current (rotor bloqué)	Is	7	A
peak winding current	Ip	21	A
Torque constant	Kt	4.60E-02	N.m/A
Voltage Constant	Ke	4.80E-03	V/rpm
			V/(rad/s)
winding resistance	Ra	0.46	Ω
winding inductance	La	6.40E-04	H
time constant	te		s
	La/Ra		s
Caractéristiques Electriques des phases du moteur			
Rphase	Rϕ	1.53E-01	Ω
Lphase	Lϕ	2.13E-04	H
Caractérisitques Mecaniques			
Rated Torque (Couple Nominal)	Tr		N.m
Peak Torque	Tp	0.95	N.m
Rated Speed	Nn	3000	rpm
			rad/s
	P/Tr		rad/s
Max Speed	Nmax	5000	rpm
			rad/s
Stall Torque	Ts	0.318	N.m
Moment of Inertia	J	30	g.cm^2
			kg.m^2
Friction Torque Max	Tt	2.00E-02	N.m
Friction torque coeff	F=Tt/N		N.m/(rad/s)
mechanical time constant	tm=J/F		s

A COMPLETER

TUTORIEL	Prise en Main de Matlab-Simulink	45 min

REMARQUE : Bien que s'agissant de rappels, il est recommandé de tester les fonctions suivantes.

Tout travail convenable avec Matlab se fait dans un fichier .m (et non directement dans la console)
Sélectionner les lignes à tester puis appuyer sur F9 pour les exécuter dans la console.

1. Prise en Main de MATLAB : Fonctions de Base

TUTO_MATLAB.m

```
%  Tout travail avec MATLAB s'effectue dans un fichier .m
% Sélectionner les lignes à exécuter et taper F9 (evaluate selection)

var1=3.14
myString='hello world'
pi
a = 10
c = 10*-2*a
nom = 'Roger';

% Modifier le format d'affichage des nombres
format long       % Notation 15 Chiffres
format short
%------------------------------------------------------------------------------
%                    REPERTOIRE DE TRAVAIL
%------------------------------------------------------------------------------
% Pour connaitre le répertoire de travail :
pwd
%Pour changer de répertoire, utiliser la commande cd:

%------------------------------------------------------------------------------
%                       OPERATIONS SCALAIRES DE BASE
%------------------------------------------------------------------------------
%Opérations de base (+,?,*,/)
7/45
(1+i)*(2+i)
1 / 0
0 / 0

%Exposants (^)
4^2
(3+4*j)^2

%Priorité des opérations + parenthèses
((2+3)*3)^0.1

%Pas de multiplication implicite!!
%3(1+0.7) % il y aura un message d'erreur!!

%Commande pour réinitialiser la fenêtre de commande
clc

%Fonctions Préprogrammées
sqrt(2)
log(2)          % Logarithme Népérien ln
log10(0.23)     % Logarithme Décimal
cos(1.2)
atan(-0.8)
exp(2+4*i)
round(1.6)      % Arrondi
floor(3.7)      % Partie Entière
ceil(4.23)      % Arrondi Supérieur

angle(i)
abs(1+i)        % Valeur Absolue ou Module d'un Complexe

%Informations sur l'utilisation d'une commande:
help sin

%Pour une documentation plus complète:
doc sin
%------------------------------------------------------------------------------
%                            MATRICES
%------------------------------------------------------------------------------

% Vecteurs
row=[1 2 5.4 -6.6]
row = [1, 2, 5.4, -6.6]
column = [4;2;7;4]

% Matrice
A= [1 2;3 4]

% Transposition
B=transpose(a)
C=A.'

%Sans le point: transposition Hermitienne,
%i.e. transposition + conjugué
%complexe
H = [1+j 2+3*j]
H'
H.'

% opérations élément par élément
A=[1 2 3];B=[4;2;1];
%A.*B        %  erreur!!
%A./B        %  erreur!!
%A.^B        %  erreur!!
A.*B'       %  Ok!!
A./B'       %  Ok!!
A.^(B')     %  Ok!!

%------------------------------------------------------------------------------
%                            POLYNOMES
%------------------------------------------------------------------------------

% P(x) = 3x^2-5x+2, il sera represente par le vecteur P:
P = [3 -5 2]

% Pour Evaluer le polynome, on utilise la fonction polyval.
% Pour calculer P(5) :
polyval(P,5)

% Calcul des Racines d'un Polynome
racines = roots(P)
```


PMSM

```
% La fonction poly permet de créer un polynome
% à partir de ses racines:
P3 = poly([1 -1])

% Tracé Polynome
P=[1, 0, -1]
x=-3:0.1:3
TRACE=polyval(P,x)
plot(x, TRACE)

%-------------------------------------------------------------------------------
%                    FONCTIONS D'INITIALISATION
%-------------------------------------------------------------------------------
o=ones(1,10)      % vecteur ligne avec 10 éléments tous égaux à 1
o=ones(2,10)      % Matrice 2 colonnes *10 lignes avec éléments %tous égaux à 1
z=zeros(23,1)     % vecteur colonne avec 23 éléments
% tous égaux à 0
r=rand(1,45)      % vecteur ligne g avec 10
% éléments aléatoires compris entre 0 et 1
%Pour initialiser une séquence linéaire: linspace
a=linspace(0,10,5) % début à 0, fin à 10 (inclus), 5 valeurs

%On peut aussi utiliser la technique suivante :
b=0:2:10; %?début à 0, par incrément de 2, fin à 10
c=1:5 %?dans ce cas, L'incrément par défaut est 1

%-------------------------------------------------------------------------------
%                       GRAPHIQUES 2D
%-------------------------------------------------------------------------------

x=linspace(0,4*pi,1000);
y=sin(x);

%Valeur de y en fonction des indexes
plot(y);

%Valeur de y en fonction de x
plot(x,y);

%On peut changer la couleur, le type de point,le type de trait
plot(x, y,'b o - '); % x, y, couleur, point, type de trait
plot(x, y, 'o')

%Titres
title('sin(Theta)'); xlabel('Theta'); ylabel('sin(Theta)');

%Pour tracer deux courbes sur le même graphique
y=sin(x);
plot(x,y);
hold on;
y=cos(x);
plot(x,y);

%Pour tracer un nouveau graphique
%(dans l'exemple, 10 est le no. de la figure)
figure(10);
plot(x, y, 'b o - ')
hold on;
plot(x,-y, 'r o - ')
%ou
figure(11);
```



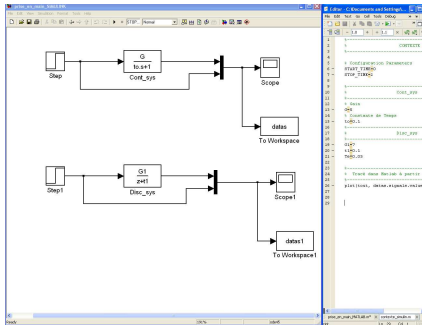
```
plot(x, y, 'b o - ',x, -y, 'r o - ');

%La commande subplot permet
%d'afficher plusieurs graphes dans une même fenêtre.
x=0:0.001:10;
subplot(1,2,1);
plot(x,sin(x));
subplot(1,2,2);
plot(x,cos(x));

%Plusieurs types de graphiques 2D
x=linspace(0,4*pi,1000);
y=sin(x);
plot(x, y);              % graphique 2D en coord. (x, y)
stem(x, y);
loglog(x, y);            % graphique log(y) vs log(x)
semilogx(x, y);          % graphique y vs log(x)
semilogy(x, y);          % graphique log(y) vs x
bar(x, y);              % graphique à barres

% Tracer la fonction
% f(x) = e^(-|x|/(2pi))*cos(x) sur l'intervale x = [-10pi 10pi].
% Utiliser une ligne rouge pleine et un nombre de points adéquat.
x=-10*pi:.01:10*pi;
plot(x, exp(-abs(x)/(2*pi)).*cos(x), 'r');

%-------------------------------------------------------------------------------
%                    ECRITURE DANS UN FICHIER
%-------------------------------------------------------------------------------
B=0:0.001:5
fid=fopen('coeff.txt','w')    %emplacement dans  %C:\MATLAB\WORK
fprintf(fid,'%i,\n',B')
fclose(fid)
%-------------------------------------------------------------------------------
%                    TEST, BOUCLES, ETC..
%-------------------------------------------------------------------------------
doc lang
%-------------------------------------------------------------------------------
%                    FREQUENCY DOMAIN ANALYSIS
%-------------------------------------------------------------------------------
%Définition d'une function de Transfert
g = tf([1 0.1 7.5],[1 0.12 9 0 0])

%Tracé du bode
bode(g)
bode(g,{0.1 , 100})
```

2. Prise en Main de Simulink

Simulink permet de simuler en temporel uniquement un système décrit à partir de blocs.

Il est impératif de définir dans un fichier .m le contexte, à savoir les valeurs numériques des différentes variables du schéma Simulink.

Il est alors nécessaire d'exécuter une fois le contexte pour définir ces variables dans le workspace.



- Entrer dans l'invite de commande Matlab
  >> simulink

- Ouvrir et exécuter le fichier .m de contexte '**CONTEXTE_SIMULINK.m**'.

```
%-------------------------------------------------------
%                    CONTEXTE
%-------------------------------------------------------

% Configuration Parameters
START_TIME=0
STOP_TIME=2

%-------------------------------------------------------
%                   Cont_sys
%-------------------------------------------------------
% Gain
G=5
% Constante de Temps
to=0.1
%-------------------------------------------------------
%                   Disc_sys
%-------------------------------------------------------
G1=7
t1=0.1
Te=0.05

%-------------------------------------------------------
%  Tracé dans Matlab à partir de 'To Workspace'
%-------------------------------------------------------
plot(tout, datas.signals.values)
```

- Ouvrir le fichier '**TUTO_SIMULINK.mdl**'
- Raccourcis Clavier :
  - Zoom in : r
  - Zoom out : v
  - Start Simulation : ctrl + t

PARTIE 2	Etude des Blocs PWM et ADC	Durée : 1UC
	OBJECTIFS	• **Etudier les périphériques du DSP relatifs à la commande** • **Revoir les formats des nombres** • **Comprendre le chemin de données**

/ETUDE_PMSM/2_PWM_ADC

REMARQUE : Lors des Tests, bien regarder le format des différentes données ; Ajouter des oscilloscopes (en simulation) afin d'expliquer les différentes conversions de format.

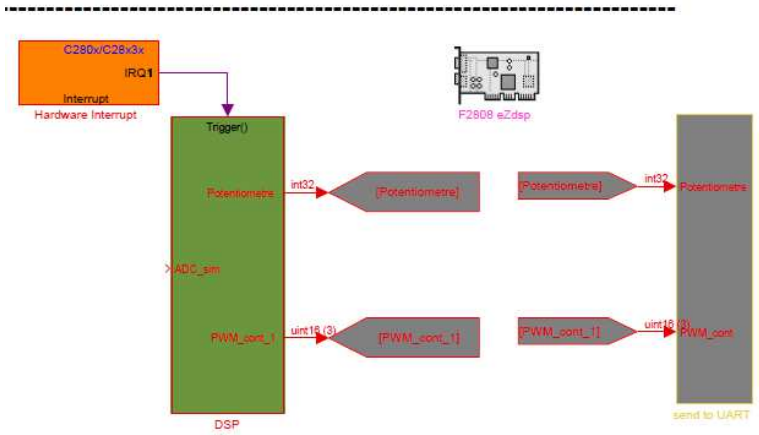
Ne pas oublier d'exécuter le fichier CONTEXTE.m au préalable pour définir les variables du schéma Simulink.

⚠ Compte tenu de l'instabilité de la solution étudiée, le moteur doit être déconnecté de la carte de puissance lors des essais sur cible (Essais à vide).

APPLICATION	Test des Blocs PWM et ADC	

1. Essai sur Cible

PWM_ADC_cible.mdl



Q1. Mesurer la fréquence F_PWM d'un des signaux PWM (U,V ou W) et vérifier l'évolution du rapport cyclique en fonction de la position du potentiomètre.

QUESTIONNAIRE	Analyse du Schéma Simulink	45 min

PWM_ADC_sim.mdl

REMARQUE : Les réponses à ces questions se trouvent dans le bloc Simulink correspondant, ou dans l'aide associée au bloc. Chaque bloc Simulink propose une aide (Help) généralement assez pertinente.

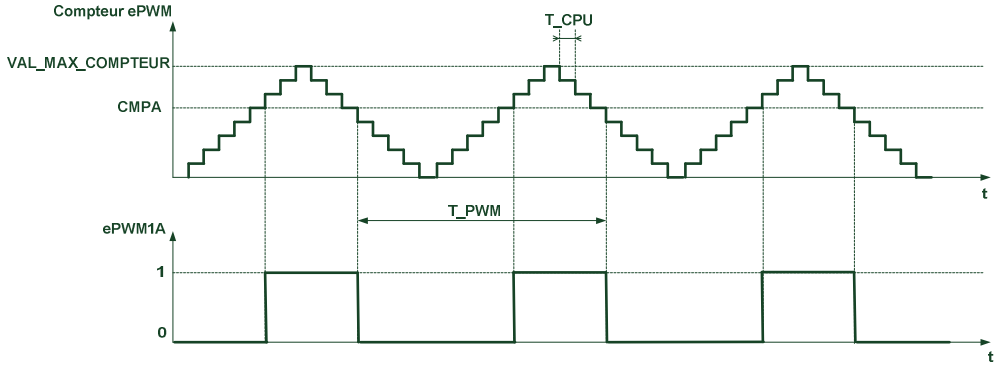
1. Bloc PWM (Pulse Width Modulation / Modulation de Largeur d'Impulsion)

Q2. Dessiner un signal PWM à valeur moyenne constante.

Q3. Donner la valeur de la tension moyenne résultante en fonction du rapport cyclique et de la tension d'alimentation.

Le mode de comptage du bloc PWM est UP/DOWN. La valeur courante de comptage s'incrémente jusqu'à une valeur de fin de comptage puis décrémente.
On peut alors considérer que cette valeur courante de comptage correspond à un signal triangulaire.

Si l'on compare cette valeur courante de comptage au contenu d'un registre CMPA (image du rapport cyclique), et que l'on force une broche du DSP (par exemple ePWM1A) en conséquence, on peut alors voir apparaître notre signal PWM.



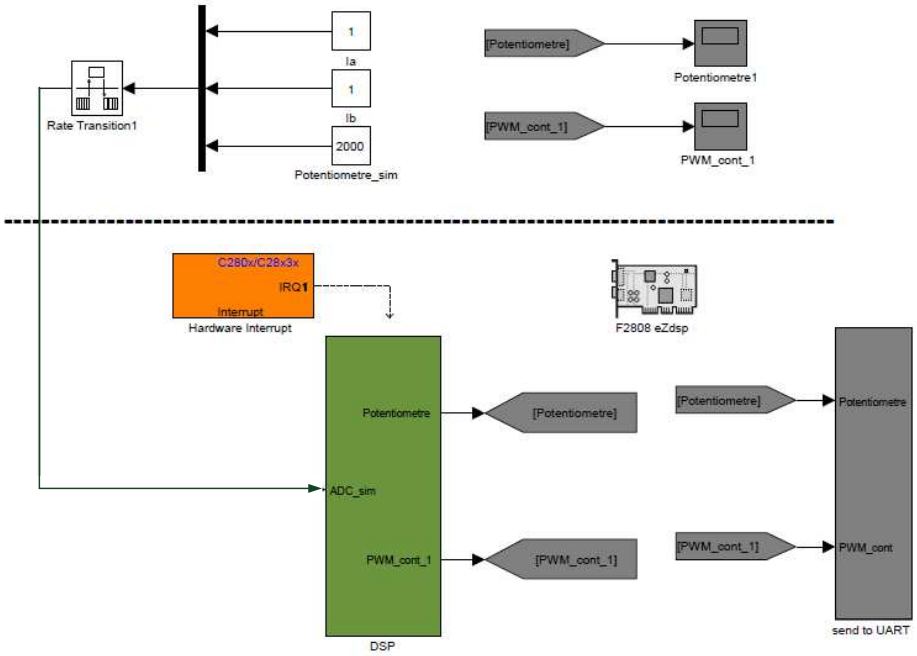
Q4. La valeur courante de comptage est-elle de type signé ou non signé ?
Q5. Exprimer T_PWM en fonction de T_CPU et VAL_MAX_COMPTEUR
Q6. A.N. : Calculer T_PWM pour F_CPU=100MHz et VAL_MAX_COMPTEUR=511

2. Bloc Analog to Digital Converter (ADC)

Q1. Quelle est la taille des données converties ? En déduire la valeur max possible.
Q2. Qu'est ce qui déclenche une conversion Analogique → Numérique ?
Q3. Quelle est alors la fréquence de conversion ?
Q4. Quel est le rôle de la bibliothèque Simulink c28x IQmath ?
Q5. A quoi correspond le format sfix32_En17 ?

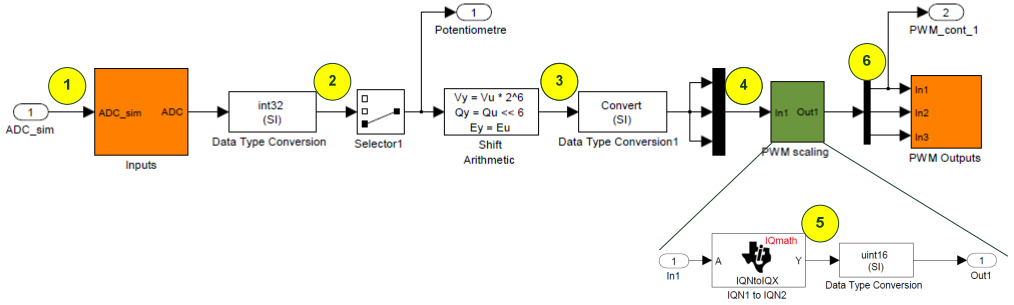
APPLICATION	Test des Blocs PWM et ADC	

PWM_ADC_sim.mdl



1. Simulation

Q1. Ajouter des scopes pour bien comprendre le rôle des différents blocs, puis compléter les tableaux suivant :

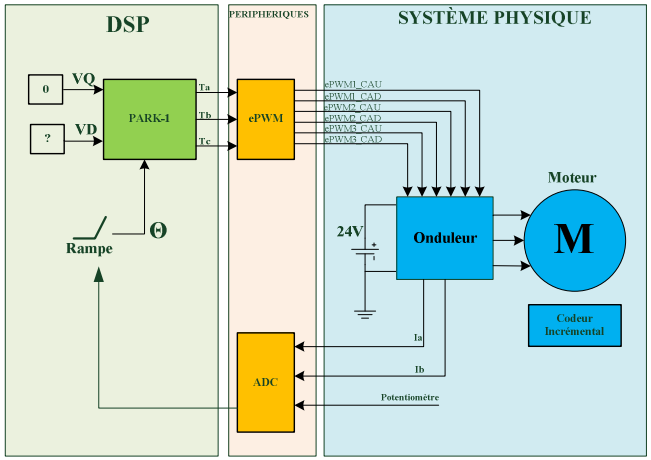


	FORMAT	BASE : Décimal	Hexa	Binaire
1	uint16	4095	0x0FFF	0000 1111 1111 1111
2	int32	4095	0x00000FFF	0000 0000 0000 0000 0000 1111 1111 1111
3				
4				
5				
6				

PARTIE 3	Génération d'un système de Tensions Triphasées	Durée : ½ UC
	OBJECTIFS	• Observer des tensions à valeur moyenne sinusoïdale pour générer le champs tournant

\ETUDE_PMSM\P3_Gene_Sinus

⚠ Les essais sur cible se font à vide (moteur débranché)

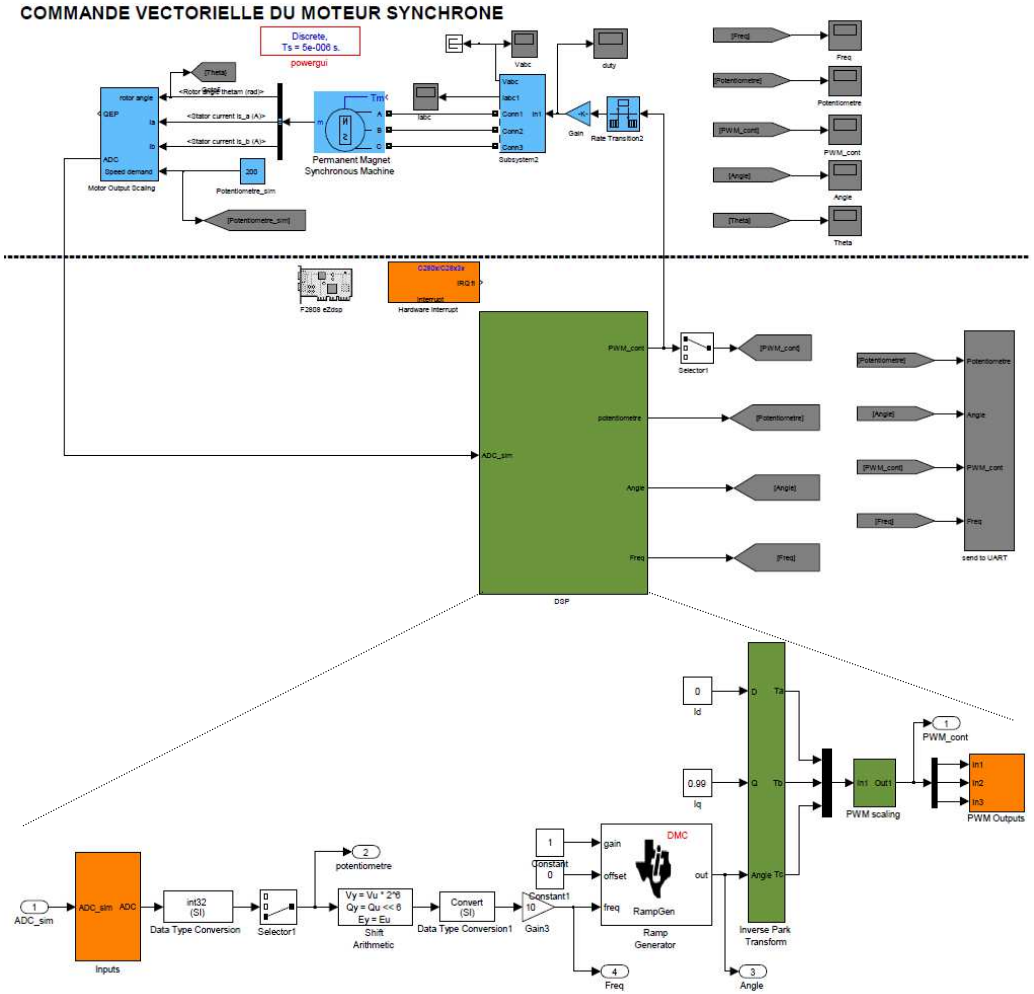


Dans cette partie l'angle des commandes sinusoïdales est généré directement à partir d'une rampe (dont la pente est contrôlable avec le potentiomètre).

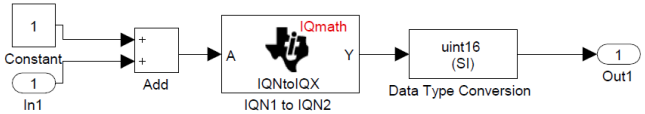
APPLICATION	Test du Contrôle Moteur sans Autopilotage	

GENE_SINUS_sim.mdl

1. Simulation



Q1. Expliquer la modification apportée au bloc « PWM scaling »



Q2. A partir de quelle valeur du potentiomètre observe-t-on un décrochage (arrêt) du moteur (cf vitesse / position du moteur) ?

2. Essai sur Cible (à Vide)

Visualiser, sur la cible, le système de tensions triphasées (utiliser la fonction Acquire → Moyenne) de l'oscilloscope. (REMARQUE : ne pas être trop exigeant sur la qualité des signaux)

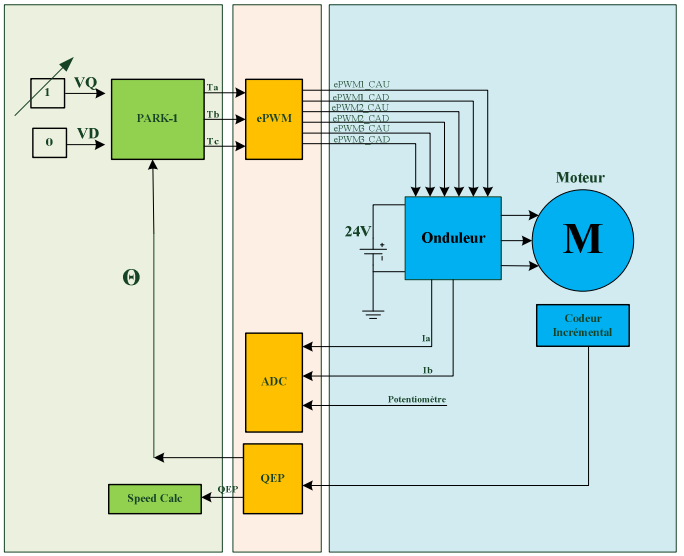
PARTIE 4	Asservissement de Vitesse	Durée : 2 UCs
	OBJECTIFS	• Mettre en œuvre l'autopilotage d'un moteur synchrone • Identifier un système en boucle ouverte • Calculer puis mettre en œuvre un correcteur de Vitesse

\ETUDE_PMSM\P4_Asservissement_Vitesse

L'Autopilotage consiste à générer les commandes en tension sinusoïdales à partir de l'information de position du rotor du moteur.

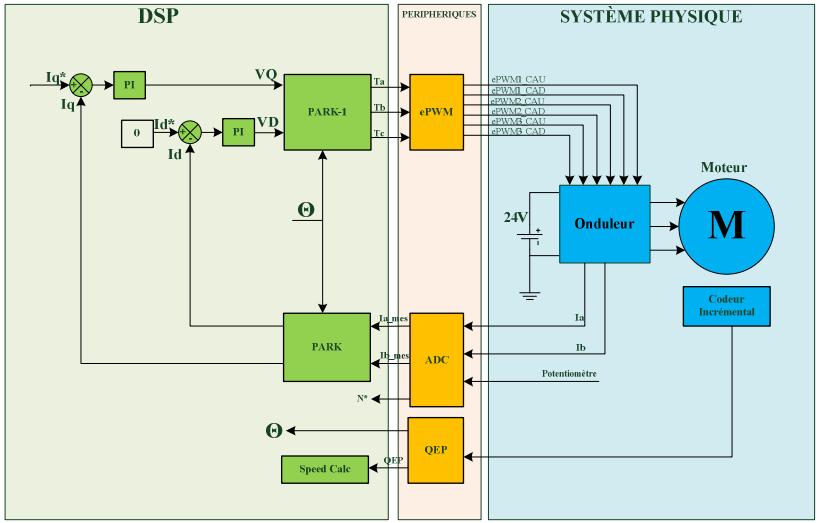
Cela permet d'avoir un parfait synchronisme entre le champs tournant et la rotation du moteur.

La commande se fait alors sur l'amplitude des tensions sinusoïdales.



Quel que soit le type de moteur, le courant est lié au couple. Un à-coup de couple peut donc entrainer un pic de courant destructeur pour le moteur et son variateur de vitesse.

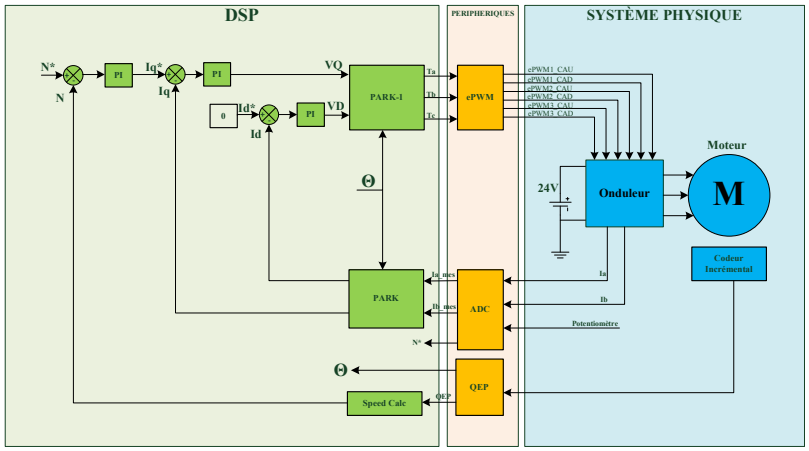
Il est donc impératif de mesurer le courant et de commander le moteur en conséquence. On réalise alors une boucle fermée en courant ; on peut alors contrôler et surtout limiter la consigne de courant et partant le courant dans le moteur.



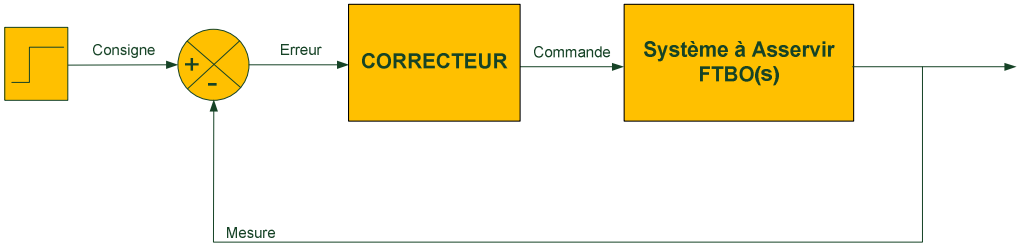
REMARQUE : la position θ est une position absolue. Lors des essais sur cible, une rampe de position est générée pour faire bouger le moteur jusqu'à ce que le signal Index du codeur incrémental passe à 1, déterminant alors la position initiale.
Il faut donc attendre la fin de cette procédure (environ 2s) avant de contrôler le moteur.

TUTORIEL **Réglage (Rapide) d'un Correcteur PI avec la marge de Phase**

Il s'agit de réaliser la structure suivante pour un contrôle de la vitesse :



Le schéma simplifié est le suivant (consigne=N*, Mesure=N).

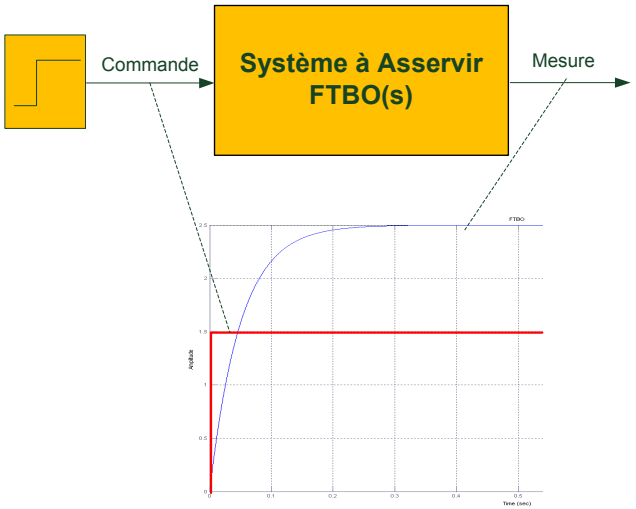


Le système à Asservir regroupe donc Le moteur, l'onduleur, la boucle de courant et les périphériques.

1. Modélisation du Système à Asservir

Avant tout calcul de correcteur, il est nécessaire d'avoir une idée du modèle du système à asservir (le plus simple étant le premier ordre G/(1+τ.s)).

Un simple essai indiciel suffit donc à déterminer le Gain et la Constante de temps du système à asservir :



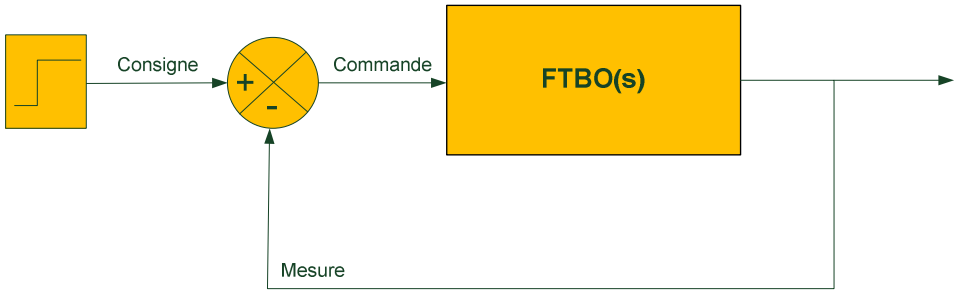
On envoie donc un échelon sur IQ* (consigne de l'image du courant) et on observe N (image de la vitesse).
Le gain G correspond au rapport entre le régime continu de la mesure et l'amplitude de l'échelon de commande.
La constante de temps τ correspond au temps nécessaire pour que la mesure atteigne 63% du régime continu.

A noter que la période d'échantillonnage du correcteur doit être au moins 10 fois plus petite que τ (afin de 'voir' les transitoires précisément).

2. Détermination d'un correcteur PI

De manière générale, asservir (boucler) un système peut entrainer de l'instabilité.

On considère le système suivant, FTBO(s) système à asservir :

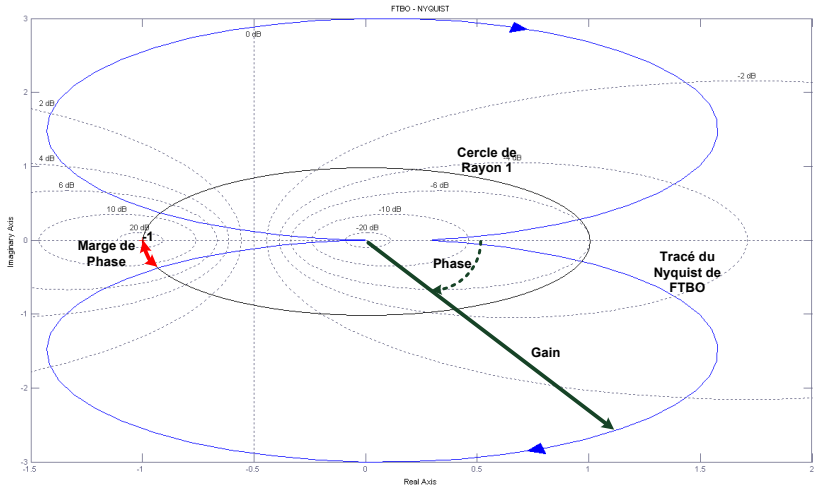


Si le module de FTBO(s) est inférieur à -1, on soustrait à la consigne une valeur négative, ce qui a pour conséquence de rendre l'ensemble instable.

Par ailleurs la consigne étant généralement soit un échelon soit une rampe, constitué d'une somme de sinusoïdes de différentes fréquences (cf. décomposition en séries de fourrier) , il peut exister une fréquence pour laquelle on s'approche dangereusement de ce point -1.

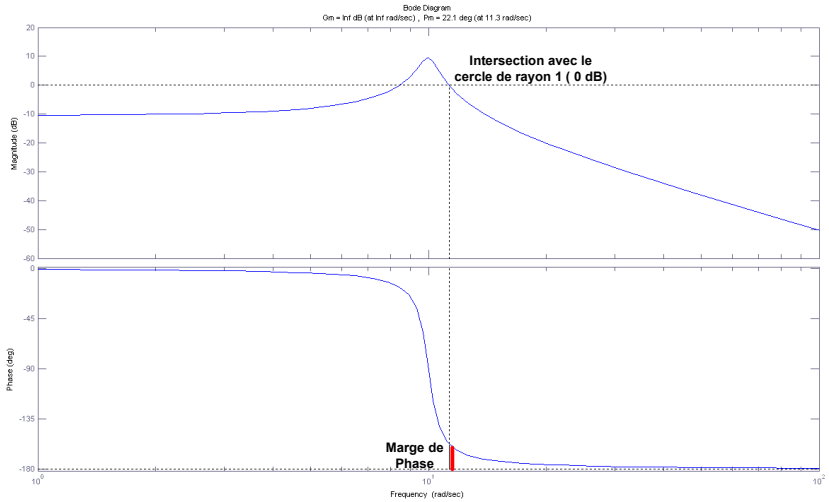
Tout le problème de l'automatique est de veiller à respecter une certaine distance vis-à-vis de ce point.

Le diagramme de Nyquist représente le tracé de FTBO(s) pour l'ensemble des fréquences :

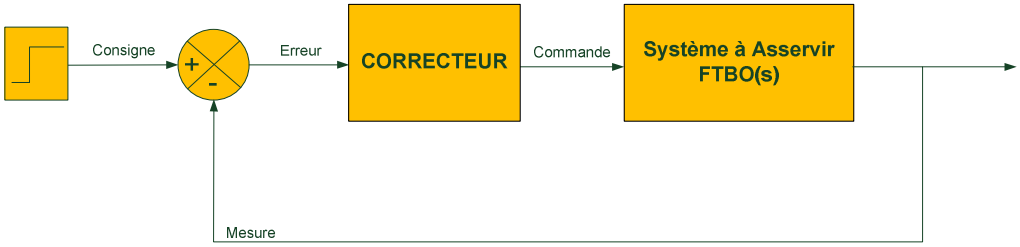


La marge de phase correspond à la distance entre le point -1 et la courbe de FTBO lorsque cette dernière coupe le cercle de rayon 1.

Cette marge de phase est également visible (et plus facilement mesurable) sur le bode de FTBO :



Un critère de réglage d'un correcteur peut donc être l'observation de cette marge de phase.



Rappel des différentes caractéristiques d'une réponse d'un système :

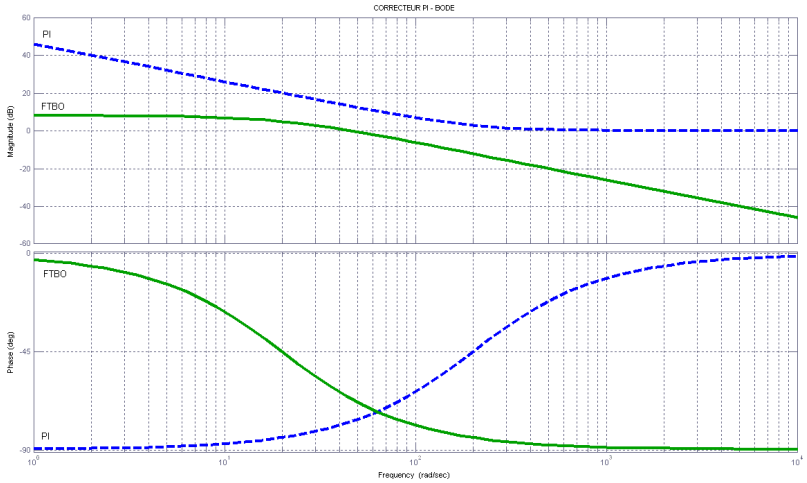
- **Stabilité** : Respect d'une marge de phase d'au moins 30°
- **Précision** : La mesure rejoint-elle la consigne ?
  - Dans le cas d'une réponse à un échelon, nécessité d'une intégration (1/s) dans la boucle ouverte pour respecter une erreur statique nulle
  - Dans le cas d'une réponse à une rampe, nécessité d'une double intégration ($1/s^2$) dans la boucle ouverte pour respecter une erreur de trainage nulle.
- **Rapidité**

Un correcteur PI possède une intégration, ce qui permet de régler le problème de l'erreur statique. En contrepartie il ralentie le système, on veillera donc à respecter une marge de phase suffisante mais pas trop grande non plus (plus la marge de phase est grande, plus la réponse et lente et les risques de saturation augmentent).

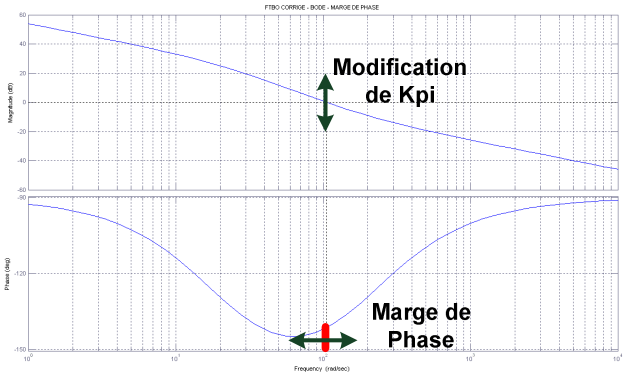
Ci-dessous figurent les bodes du correcteur PI et de FTBO tel que la constante de temps du correcteur PI vaut $0.1*\tau$

$$PI(p) = K_1 + \frac{K_2}{p}$$ On préfèrera la forme : $$PI(p) = Kpi.\frac{\omega_i}{p}(1+\frac{p}{\omega_i})$$

Avec : Kpi= K_1 , $\omega_i = \frac{K_2}{K_1}$ =1/Ti

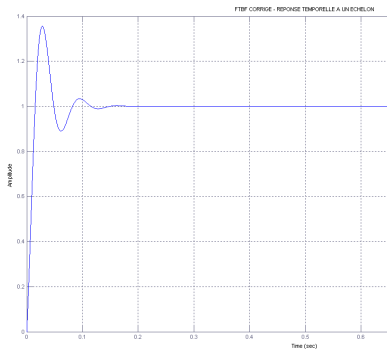


L'association de ces deux blocs donne :



La constante de temps du correcteur étant fixé, le tracé de la phase est figé, on peut alors jouer sur le gain du correcteur pour modifier le module du système corrigé en boucle ouverte et de fait régler la marge de phase.

Pour une marge de phase de 45° on a la réponse temporelle suivante :



Le fichier **CALCUL_PI.m** reprend la méthode ci-dessus.
A noter qu'il s'agit d'un PI ANALOGIQUE.
Le correcteur numérique sous Simulink fonctionne à partir de la discrétisation d'un correcteur analogique pour lequel :
Kp=Kpi
Ki=Te/Ti , Te Période d'Echantillonnage , Ti Constante de temps du correcteur PI.

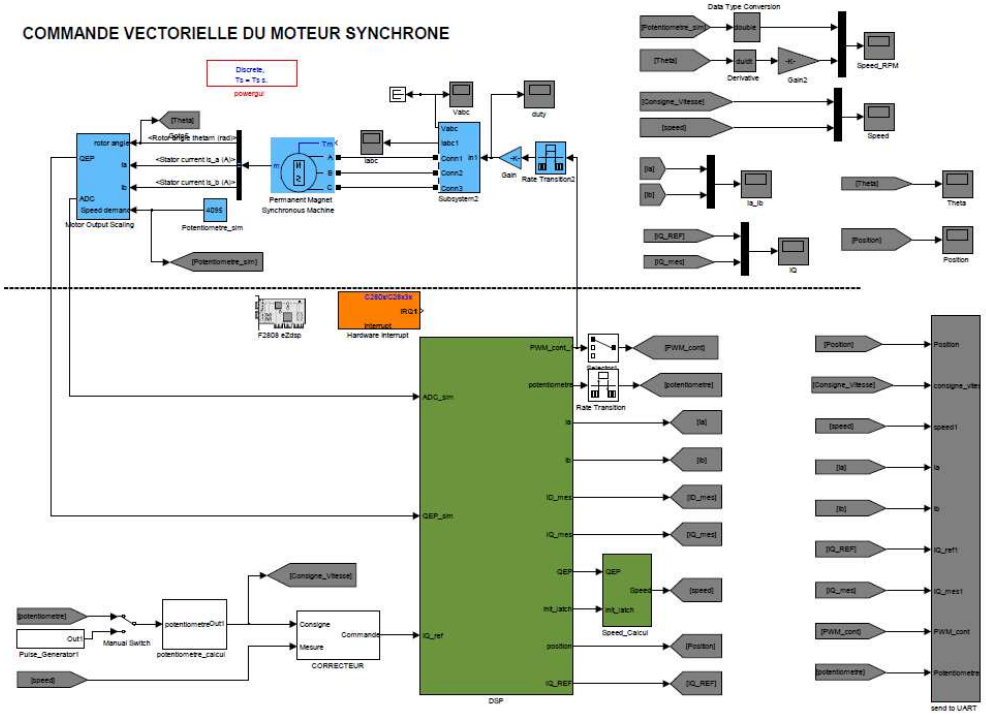
```
%**************************************************************************
%   IPS - CALCUL D'UN PI
%   kerhoas@enib.fr
%**************************************************************************

i=0; % index Figures
%--------------------------------------------------------------------------
%    Définition du Gain et de la Constante de Temps en Boucle Ouverte
%    du système à asservir G/(1+to*s)
%--------------------------------------------------------------------------

G=2.5
to=50e-3

%--------------------------------------------------------------------------
%              Fonction de Transfert en Boucle Ouverte + Tracé
%--------------------------------------------------------------------------

FTBO = tf([G],[to,1])

i=i+1;
figure(i)
t=0:0.0001:1
step(FTBO,t)
grid
title('FTBO - REPONSE A UN ECHELON')

i=i+1;
figure(i)
subplot(1,2,1)
nyquist(FTBO)
grid
title('FTBO - NYQUIST')
subplot(1,2,2)
```



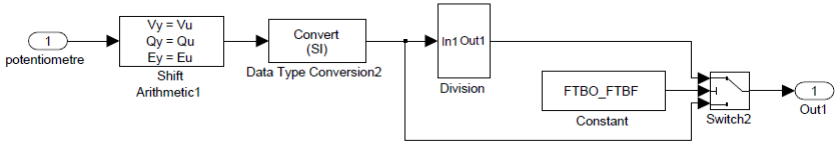
```
bode(FTBO)
grid
title('FTBO - BODE')

%--------------------------------------------------------------------------
%              Définition du Correcteur PI + Tracé
%--------------------------------------------------------------------------

ti = 0.1*to
Kpi = 1

CORR = tf(Kpi*[ti , 1],[ti, 0])

i=i+1;
figure(i)
bode(CORR)
grid
title('CORRECTEUR PI - BODE')
%hold on
%bode(FTBO)

%--------------------------------------------------------------------------
%     Fonction de Transfert EN BO du Système Corrigé + Tracé
%--------------------------------------------------------------------------

FTBO_CORR=CORR*FTBO
[Gm,Pm,Wg,Wp] = margin(FTBO_CORR) % Pm Marge de Phase

i=i+1;
figure(i)
bode(FTBO_CORR)
margin(FTBO_CORR)
grid
title('FTBO CORRIGE - BODE - MARGE DE PHASE')

%--------------------------------------------------------------------------
%     Fonction de Transfert EN BF du Système Corrigé + Tracé
%--------------------------------------------------------------------------

FTBF= (CORR*FTBO)/(1+CORR*FTBO)

i=i+1;
figure(i)
bode(FTBF)
title('FTBF CORRIGE - BODE')
grid
i=i+1;
figure(i)
t=0:0.0001:1
step(FTBF,t)
grid
title('FTBF CORRIGE - REPONSE TEMPORELLE A UN ECHELON')
```

APPLICATION	Asservissement de Vitesse du Moteur Synchrone	2 UC

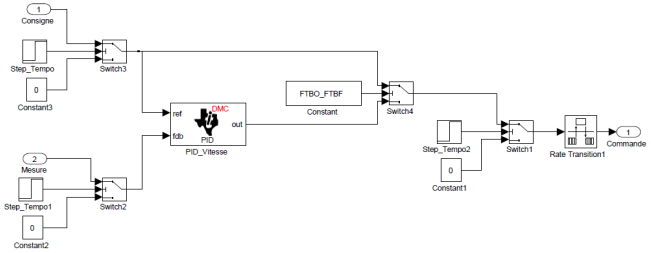
ASSERV_VITESSE_sim.mdl



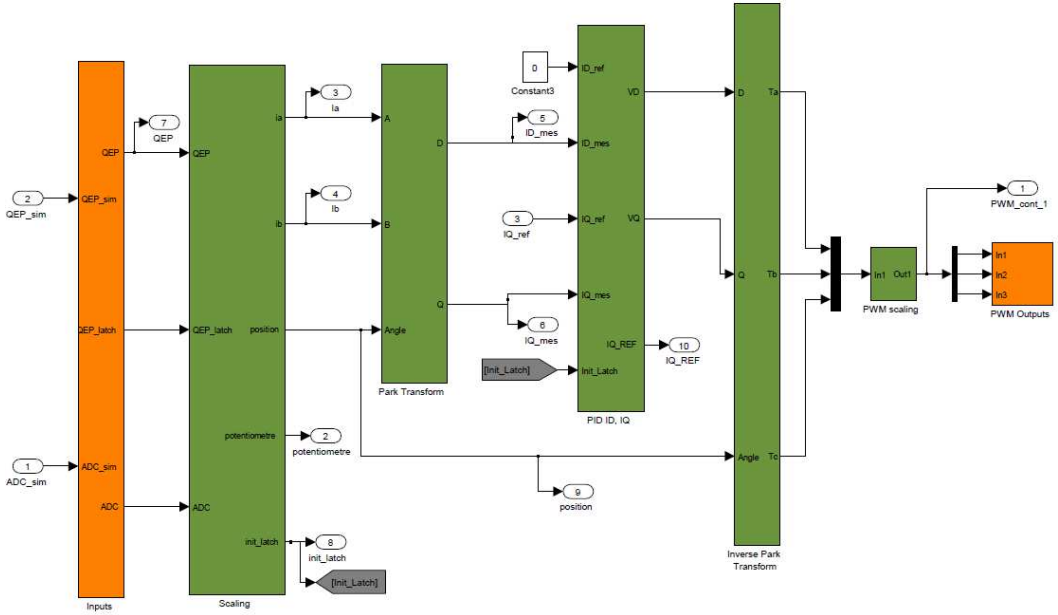
Potentiometre_calcul :



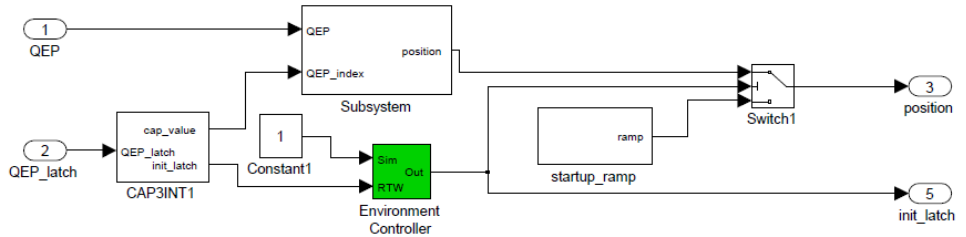
CORRECTEUR :

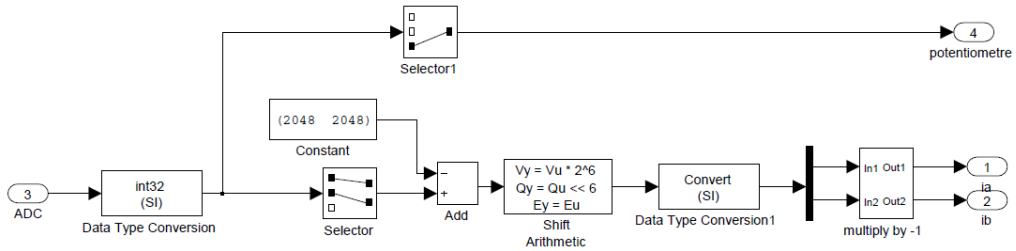


DSP :

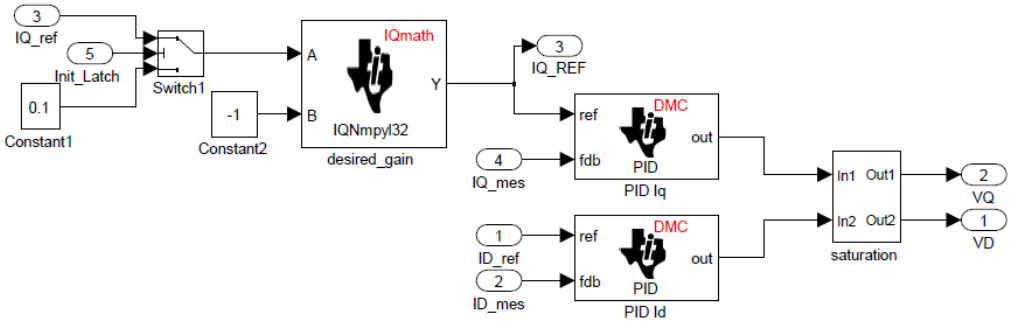


Scalling :

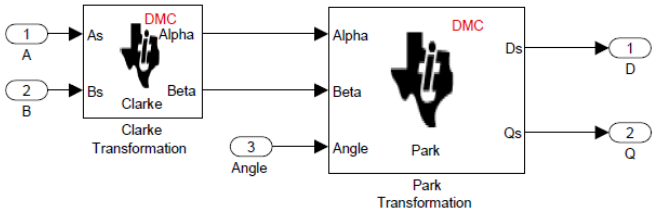




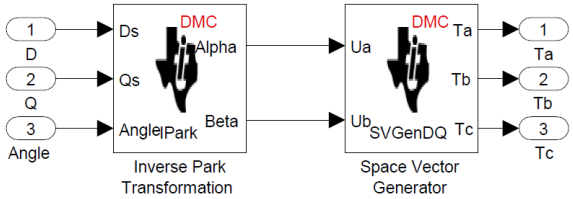
PID ID,IQ :



Park Transform :



Inverse Park Transform



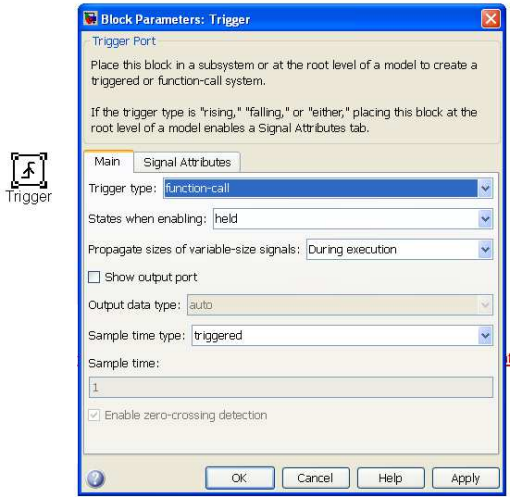
CONTEXTE.m

```
FTBO=1
FTBF=0
SIMU=0
TARGET=1
OUI=1
NON=0
GAUCHE=1
DROITE=-1

%=========================================================
% CHOIX FONCTIONNEMENT BO/BF EN VITESSE
FTBO_FTBF=FTBO              % FTBF

% CHOIX SIMU/FONCTIONNMENT SUR CIBLE (TARGET)
SIMU_TARGET=SIMU      % TARGET

% CHOIX MOTEUR A VIDE / AVEC TABLE_LINEAIRE
TABLE_LINEAIRE=NON

% CHOIX SENS DE ROTATION
SENS=GAUCHE

%=========================================================
% Caractéristiques ePWM
NBBITS_PWM=9
Q_VALUE=NBBITS_PWM-1
VAL_MAX_COMPTEUR=2^NBBITS_PWM-1
F_CPU=100e6     % Fréquence Calcul DSP

% Période Signal PWM (Echantillonage Boucle de Courant)
Ts=round(2*VAL_MAX_COMPTEUR*(1/F_CPU)*1e6)/1e6

% Période d'échantillonage pour la Boucle de Vitesse
Te=1e-3

% Période d'échantillonage UART
T_UART=0.01

% Caractéristiques Système
encoderResolution = 2000 ;
indexOffset = 850 ;
polePairs = 4 ;

% Paramètres Consigne
CONS_COEFF = 0.1

If TABLE_LINEAIRE==OUI
DECAL=6
CONSIGNE_I_START=SENS*0.4
else
DECAL=4
CONSIGNE_I_START=SENS*0.1
end




% PULSE GENERATOR
```

```
% T_pulse Période du Signal Carré désiré
T_pulse=10
T_pulse_corr=T_pulse/2.5
F_pulse=1/T_pulse
FREQ=0.001
VAL_PULSE_MAX=4095
MAX_STEP_ANGLE=1/T_pulse_corr

% TEMPO POUR RAMPE DE STARTUP
if SIMU_TARGET==SIMU
Tempo=0;
else
Tempo=2.5;
end
Tempo_Startup=Tempo/2.5

%-----------------------------------------------------
%                  PID Courant
%-----------------------------------------------------
% Limitation du courant
CURRENT_LIMIT=2.5                  % Limitation en sortie du PID Vitesse
toe=1.4e-3                         % Constante de temps de la BO Courant
OUTPUT_PID_CURRENT_LIMIT=0.99      % Limitation en sortie du PID Courant

Kp_i=0.9; Ki_i=Ts/(0.1*toe); Kd_i=0; Kc_i=0
%-----------------------------------------------------
%                  PID Vitesse
%-----------------------------------------------------
Kp_v=1; Ki_v=0   % A Modifiier
Kc_v=0; Kd_v=0
```


Marche à suivre pour passer d'un fichier de simulation à un fichier de génération de code pour la cible:

1- Dans le contexte, remplacer `SIMU_TARGET=SIMU` par `SIMU_TARGET=TARGET` et réexécuter le contexte
2- Sauvegarder le fichier Simulink sous un autre nom
3- Ajouter le bloc TRIGGER configuré en FUNCTION CALL dans le sous-système DSP



4- Connecter la nouvelle entrée du sous-système DSP au bloc 'hardware interrupt'
5- Effacer tout ce qui figure au dessus des pointillés (ainsi que tous les scopes)
6- Effacer la variable CCS_Obj si cette dernière est présente dans l'environnement MATLAB
7- Stopper CCS si nécessaire
8- Faire CTRL+B

1. Essai Temporel : Détermination du modèle de la FTBO

Sur Cible, utiliser pulse_generator plutôt que le potentiomètre pour générer un échelon de commande ou de consigne.

Effectuer l'essai indiciel en Boucle Ouverte en Simulation et sur Cible afin de déterminer les caractéristiques du modèle du système à asservir.

Observer le comportement du moteur face à une perturbation de couple.

2. Test du d'un correcteur proportionnel en vitesse

REMARQUE : Dans le contexte, remplacer FTBO_FTBF=FTBO par FTBO_FTBF=FTBF pour permettre l'utilisation du correcteur de vitesse.

Tester l'effet d'un correcteur proportionnel (de gain 1 par exemple) et observer le comportement du moteur face à une perturbation de couple.

3. Test d'un correcteur PI en vitesse

Appliquer la méthode du tutoriel ci-dessus afin de déterminer un correcteur PI de vitesse stable et le plus rapide possible.

Comparer les résultats de simulation et sur la cible.

PARTIE 5	Commande de la Table linéaire	Durée : 3UCs
	OBJECTIFS	• Mise en Contexte du Moteur dans un système • Contrôle du Moteur Via Bus de Terrain

Les maquettes de la table linéaire n'étant pas encore disponible, il sera simplement question dans cette partie de simuler la commande de ce système
Il n'est pas nécessaire de respecter l'ordre ex

APPLICATION 1	Pilotage du Moteur dans les deux Sens	45 min

Modifier la structure de commande de la partie 4 pour pouvoir faire tourner le moteur dans les deux sens avec le potentiomètre

APPLICATION 2	Prise en compte des capteurs de fin de course	45 min

Deux boutons poussoirs sont disponibles sur la carte DMC550 permettant de simuler des capteurs de fin de course.
Modifier la structure de commande afin de modifier le comportement du moteur lorsqu'un des boutons poussoirs est actionné.
Deux approches :

1. **Arrêt du moteur lors de l'appui sur un bouton**
2. **Changement du sens de rotation (vitesse constante)**

Il est nécessaire d'envisager une machine d'état via un bloc « Embedded Matlab Function ».
Exemple d'une syntaxe partielle pour une machine d'état :

```
function y= fcn(e1,e2)

persistent etat

if isempty(etat)
etat = 0;
y=0;
end

switch etat
   case 0
      y=.....                    % Affectaction de la sortie
      if (fc1==0) etat=.....     % Détermination de l'état suivant
      end

   case 1
       y=.....
      if (fc2==0) etat=....
      end

   otherwise
      y=0

end
```

APPLICATION 3	Déplacement de la Table Linéaire	30 min

⚠ **Appeler l'enseignant pour raccorder le moteur à la table linéaire**

Proposer un schéma simulink permettant de déplacer la table linéaire dans les deux sens, en boucle ouverte en vitesse puis en boucle fermée, en tenant compte des interrupteurs fin de course.

APPLICATION 4	Utilisation du Bus CAN pour l'échange d'informations avec le DSP	45 min

Une interface CAN est disponible sur la carte ezDSPf2808. Ajouter les blocs nécessaires afin de transmettre la consigne de position ou de vitesse via le bus CAN. (On utilisera le logiciel PCANView côté PC)
REMARQUE : la taille des message doit être nécessairement de 8 octets.

APPLICATION 6	Asservissement de Position	45 min

Proposer une structure de commande permettant l'asservissement en position du moteur (la boucle de vitesse étant imbriquée dans la boucle de position)

APPLICATION 5	Simulation de la Table Linéaire	30 min

Proposer un schéma de simulation représentant l'ensemble du système avec la table linaire de telle sorte que les mesures sur cible correspondent aux grandeurs simulées.