Identification des paramètres du MX28

L'actionneur MX28 dispose de champs non-documentés mais identifiés dans le code source du robot Darwinop[1][2]. Le format de ces champs a été déterminé d'après les valeurs retournées dans les expérimentations présentées par la suite. Ces grandeurs permettent d'accéder aux variables internes du correcteur PID parallèle (Tableau 1 et Figure 1).

Adresse (taille)	Format	Fonction
54 (2 octets)	Int10 + 11ème bit = signe	PWM = rapport cyclique de la commande du moteur (entre -1023 et +1023).
56 (2 octets)	Int16	Entrée du gain proportionnel
58 (2 octets)	Int16	Entrée du gain intégral
60 (2 octets)	Int16	Entrée du gain dérivé
62 (2 octets)	Int16	Sortie du correcteur proportionnel
64 (2 octets)	Int16	Sortie du correcteur intégral
66 (2 octets)	Int16	Sortie du correcteur dérivé

Tableau 1 Champs du correcteur PID parallèle

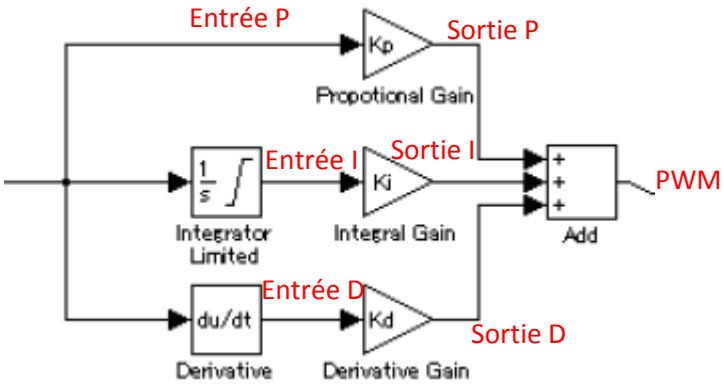


Figure 1 Instrumentation du correcteur PID parallèle

Protocole de tests

Les tests suivants ont été réalisés à l'aide de la bibliothèque darwinoplib[3] facilitant l'écriture de programmes embarqués sur le robot Darwin à l'aide de l'interface Simulink.

Cette instrumentation était effectivement exécutée sur le robot afin d'éliminer les latences du réseau. Les mesures étaient transférées automatiquement par blocs vers Simulink pour l'affichage. Cette configuration permet d'obtenir des périodes d'échantillonnage de 8 ms (identique au fonctionnement normal du robot) tout en obtenant un affichage des données quasiment en temps réel.

Pour les caractérisations nécessitant une période d'échantillonnage plus faible (4 ms par exemple), il a été nécessaire d'exporter localement les données sous la forme « .mat », de les transférer sur le PC afin de les analyser.

[1] http://sourceforge.net/p/darwinop/code/HEAD/tree/trunk/darwin/Framework/include/MX28.h#l151
[2] https://github.com/darwinop-ens/darwin-op/blob/master/Framework/include/MX28.h#L106
[3] https://github.com/darwinop-ens/simulink

Le modèle Simulink de caractérisation est présenté en Figure 2. Il est constitué d'un ensemble de stimuli permettant de régler les gains du correcteur ainsi que les consignes en vitesse et en position.

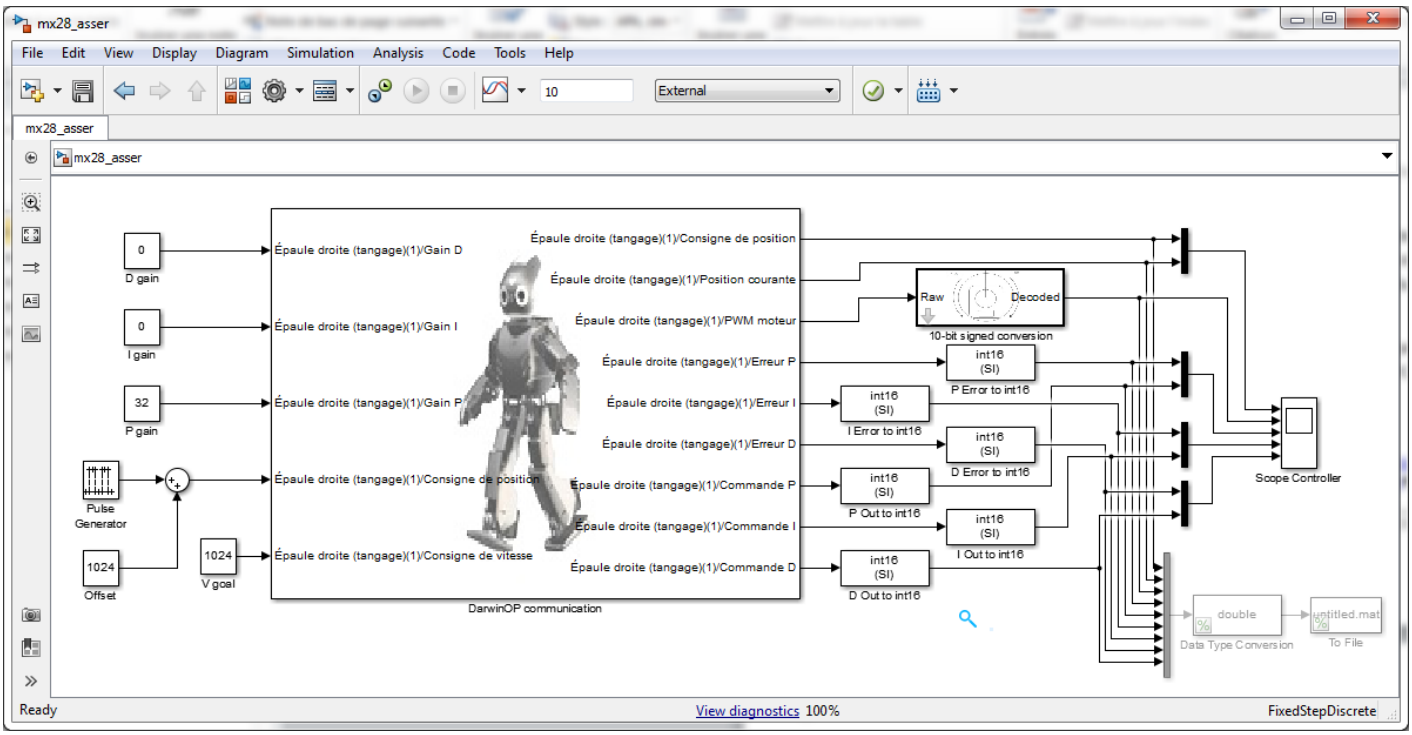


Figure 2 Modèle de caractérisation (MX28_identification_p.slx)

La partie droite de cette figure représente les blocs de remise en forme des données lues (conversions en entiers signés sur 16-bit des entrées/sorties des correcteurs) ainsi que la représentation graphique des résultats dans un scope.

Les résultats obtenus sur le graphe (Figure 3) représentent les grandeurs suivantes :

- Consigne en position sur le graphe 1, courbe jaune,
- Position mesurée sur le graphe 1, courbe violette,
- PWM du moteur sur le graphe 2, courbe jaune,
- Entrée du correcteur proportionnel sur le graphe 3, courbe jaune,
- Sortie du correcteur proportionnel sur le graphe 3, courbe violette,
- Entrée du correcteur intégral sur le graphe 4, courbe jaune,
- Sortie du correcteur intégral sur le graphe 4, courbe violette,
- Entrée du correcteur dérivé sur le graphe 5, courbe jaune,
- Sortie du correcteur dérivé sur le graphe 5, courbe violette.

Ce scope est configuré pour exporter ses données dans l'environnement Matlab dans une variable nommée « ScopeData ».

On peut remarquer que la sortie des correcteurs intégral et dérivé sont toujours à « 0 ». Le stimulus impose des mouvements entre les positions 1024 3072 de l'articulation.

L'actionneur se déplace rapidement après les changements de consignes à t = 0 s, 1 s, 5 s et 9 s. Les lectures de ces grandeurs par le bus dynamixel du robot ne sont pas instantanées, les champs seront effectivement lus à des instants légèrement différents. L'actionneur continuant son mouvement entre deux lectures, on observera dans les caractérisations suivantes un bruit inhérent à ce protocole de mesure.

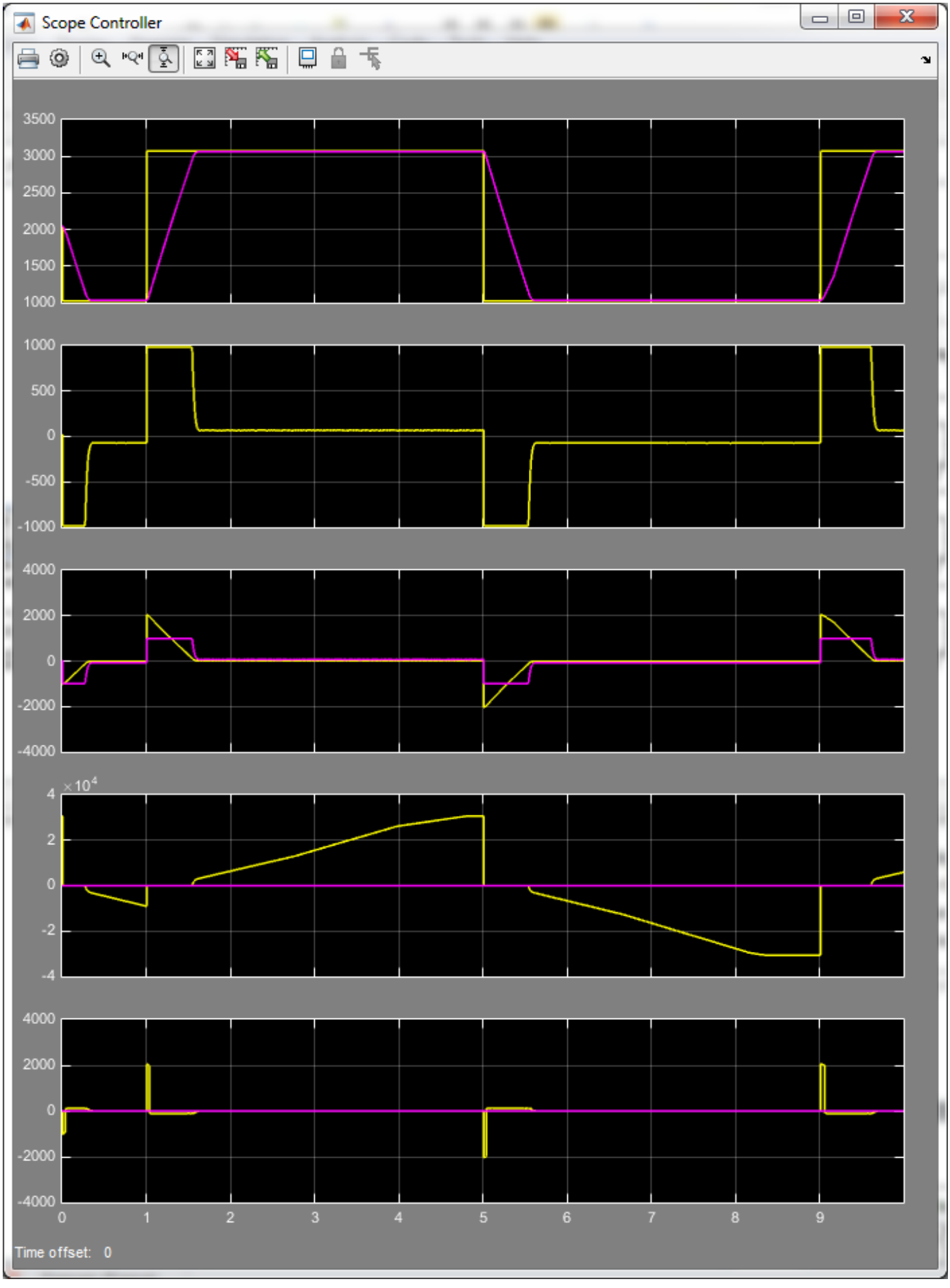


Figure 3 Données mesurées sur le robot

Les différents vecteurs peuvent être extraits à l'aide des commandes Matlab suivantes :

```
>> time = ScopeData.time;
>> pos_cons = ScopeData.signals(1).values(:,1);
>> pos_mes = ScopeData.signals(1).values(:,2);
>> pwm = ScopeData.signals(2).values;
>> p_in = ScopeData.signals(3).values(:,1);
>> p_out = ScopeData.signals(3).values(:,2);
>> i_in = ScopeData.signals(4).values(:,1);
>> i_out = ScopeData.signals(4).values(:,2);
>> d_in = ScopeData.signals(5).values(:,1);
>> d_out = ScopeData.signals(5).values(:,2);
```

Ces vecteurs sont de dimensions 1250 x 1 et contiennent des « double » pour les 4 premiers et des « int16 » pour les 6 derniers.

Caractérisation du correcteur PID

Caractérisation du signal d'erreur

L'erreur, dans le sens « automatique », est la différence entre la consigne et le retour de la grandeur asservie, ici la position de l'actionneur. Dans cette caractérisation, cette erreur peut être calculée à l'aide des commandes suivantes :

```
>> epsilon = pos_cons - pos_mes;
```

Dans le cas de ces déplacements, le signal d'erreur a des valeurs comprises entre -1020 et +1014.

Caractérisation de l'entrée du correcteur proportionnel

On remarque que le signal à l'entrée du correcteur proportionnel est identique au signal d'erreur.

```
>> [min(epsilon - double(p_in)) max(epsilon - double(p_in))]
ans =
   -10     9
```

La différence entre ces deux signaux est comprise entre -10 et +9. Ces extremums sont localisés aux instants de forts déplacements et correspondent au bruit lié à la mesure pendant que l'articulation continue son mouvement. Ces valeurs extrêmes de l'écart correspondent à moins de 1% de la valeur de l'erreur « epsilon ».

Par déduction, l'entrée du correcteur proportionnel est égale au signal d'erreur « epsilon » du système asservi.

Caractérisation de la sortie du correcteur proportionnel

La sortie du correcteur proportionnel semble saturée lors des changements de consignes.

```
>> [min(p_out) max(p_out)]
ans =
   -987    987
```

Ces valeurs minimales et maximales semblent correspondre à une saturation artificielle de la sortie du correcteur proportionnel. Ce correcteur est non-saturé à tous les échantillons suivants :

```
>> p_non_satur = find(abs(p_out) ~= 987);
```

A l'inverse, ce correcteur est saturé à tous les échantillons suivants :

```
>> p_satur = find(abs(p_out) == 987);
```

Caractérisation du gain du correcteur proportionnel

En dehors de ces instants saturés, on remarque que la sortie du correcteur semble proportionnelle à son entrée avec un rapport 8 :

```
>> p_gain = double(p_out(p_non_satur)) ./ double(p_in(p_non_satur));
```

```
>> [min(p_gain) max(p_gain)]
ans =
     8     8
```

Hors le gain spécifié est 32, par conséquent le gain effectif du correcteur proportionnel est ¼ du gain réglé dans les paramètres du MX28. Cette grandeur ne correspond pas aux spécifications du constructeur indiquant un rapport de 1/8[4].

Caractérisation de l'entrée du correcteur intégral

On peut remarquer que l'entrée du correcteur intégral est remise à zéro aux instants où la sortie du correcteur proportionnelle est saturée :

```
>> i_in_zero = i_in(p_satur);
>> [min(i_in_zero) max(i_in_zero)]
ans =
      0      0
```

A l'inverse on peut remarquer peu avant t = 5s et t = 9s des saturations dans l'entrée du correcteur intégral. Cela est dû à une limitation dans l'intégration pour éviter les débordements numériques :

```
>> [min(i_in) max(i_in)]
ans =
 -30690  30690
```

En dehors de ces instants de remise à zéro et de saturation, il est possible de déterminer la relation liant le signal d'erreur « epsilon » à l'entrée du correcteur intégral. On se place par exemple entre les instants t = 1,544 s (élément d'indice 194 du tableau « time ») et 4.800 s (indice 601) correspondant respectivement à la sortie de la saturation du correcteur proportionnel (donc fin de la remise à zéro de l'intégrateur) et au début de la saturation de l'intégrateur. Cette plage correspond au fonctionnement linéaire de l'intégrateur :

```
>> i_in_lin = double(i_in(194:601));
>> epsilon_lin = double(epsilon(194:601));
```

L'intégrateur idéal est déterminé par les commandes suivantes (intégration par la méthode des rectangles de largeur Te = 8 ms) :

```
>> i_ideal_lin = cumsum(epsilon_lin)*0.008;
```

La transformation entre l'entrée réelle du correcteur integral et l'intégrateur ideal peut être obtenu par les commandes suivantes :

```
>> i_gain = i_in_lin ./ i_ideal_lin;
>> mean(i_gain)
ans =
  998.5342
```

On peut en déduire que le gain de l'intégrateur est 1000, cohérent avec la documentation constructeur[5].

Caractérisation de la sortie du correcteur intégral

Cette caractérisation nécessite la spécification d'un gain intégral dans le modèle ainsi qu'un stimulus de type triangle afin d'avoir une erreur de traînage à corriger[6] (Figure 4).

[4] http://support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm#Actuator_Address_1A
[5] http://support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm#Actuator_Address_1A
[6] Un MCC asservi en position (en négligeant les frottements secs) avec un correcteur intégral correspond à un modèle automatique avec un pôle dans la fonction de transfert en boucle ouverte. L'erreur statique d'un tel montage est intrinsèquement nulle. L'ajout d'un correcteur intégral sur cette structure n'apporte un plus que si on s'intéresse à l'erreur de poursuite (traînage).

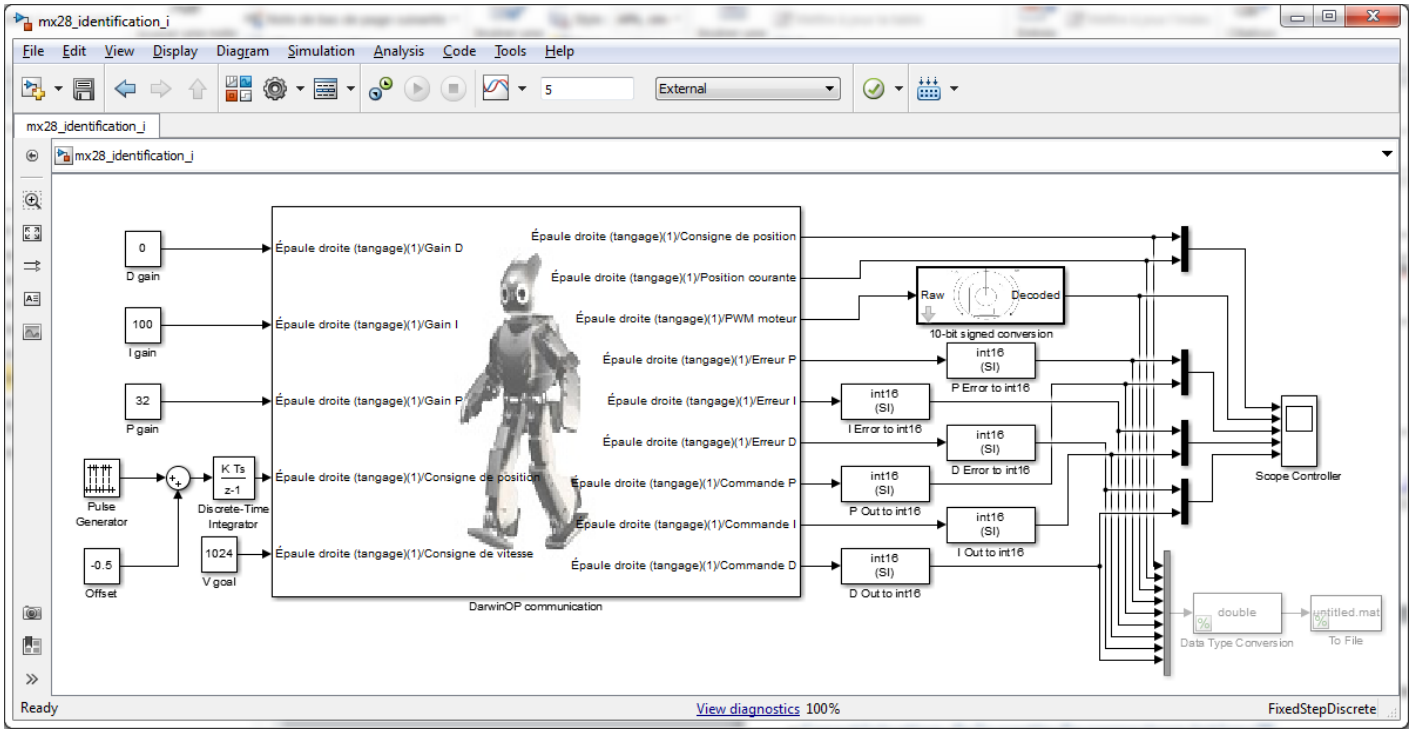


Figure 4 Modèle de stimulation pour caractériser le correcteur intégral (MX28_identification_i.slx)

Les courbes obtenues pour le correcteur intégral (entrée en jaune et sortie en violet) sont présentées dans la Figure 5.

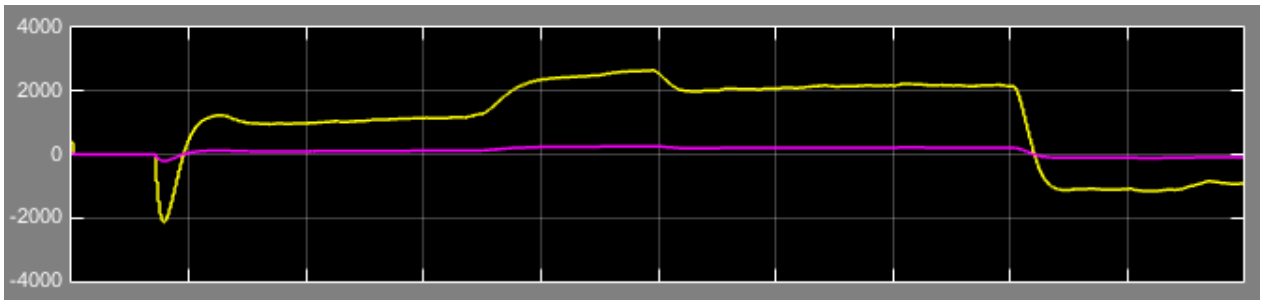


Figure 5 Entrée et sortie du correcteur intégral

La relation de proportionnalité entre ces deux courbes est la suivante :

```
>> i_gain = double(i_out) ./ double(i_in);
>> nanmean(i_gain)
ans =
    0.0978
```

Après plusieurs expérimentations, ce gain serait égal à 100 (gain I demandé) / 1024. Cette valeur ne correspond pas aux spécifications du constructeur[7] qui indique 2048.

Caractérisation de l'entrée du correcteur dérivateur

Cette caractérisation nécessite une fréquence d'échantillonnage très élevée pour être exécutée avec une bonne précision. Le modèle est donc simplifié pour en réduire la complexité et modifié pour générer des fichiers « .mat » sur le robot.

[7] http://support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm#Actuator_Address_1A

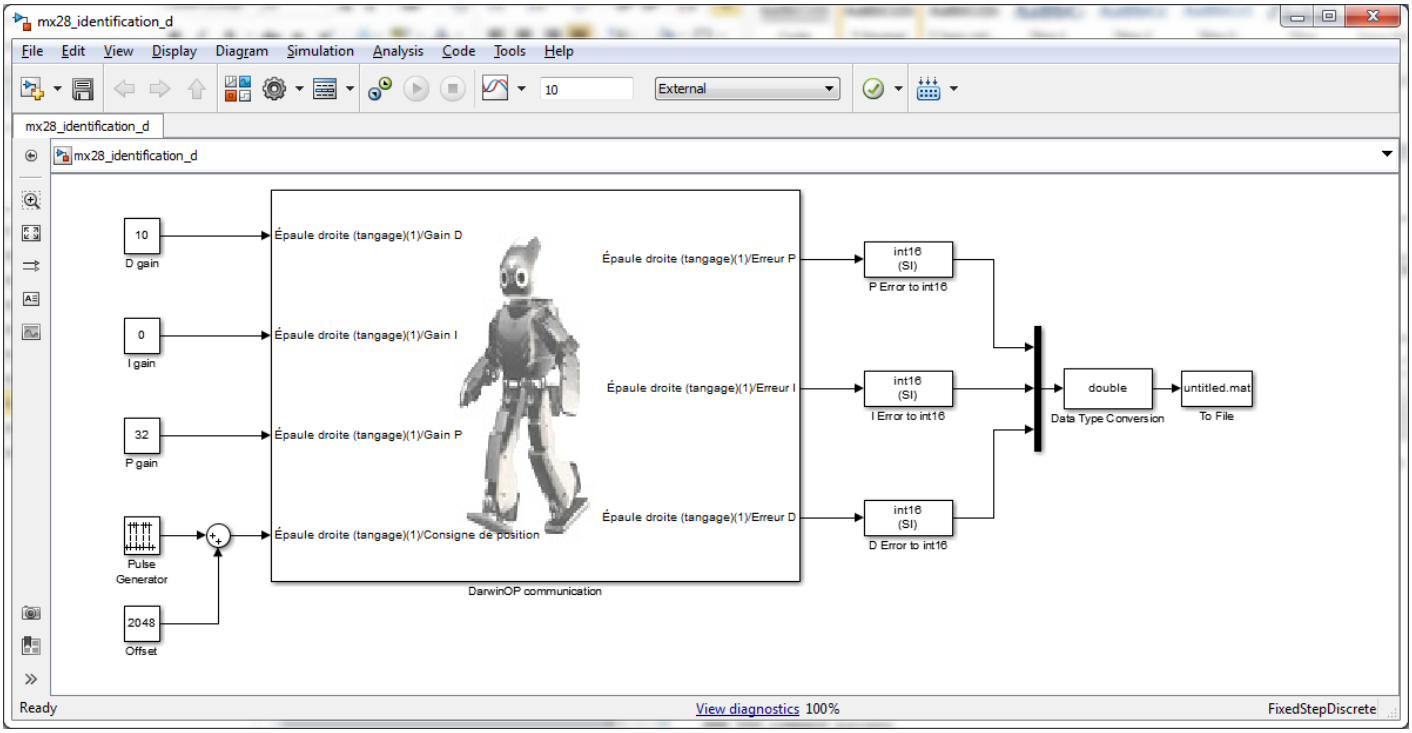


Figure 6 Modèle d'identification de l'entrée du correcteur dérivée

Ces fichiers sont ensuite rapatriés sur le PC et analysés avec Matlab :

```
>> load('untitled.mat')
>> time = Data(1,:);
>> p_in = Data(2,:);
>> i_in = Data(3,:);
>> d_in = Data(4,:);
```

Le but est d'intégrer l'entrée du correcteur différentiel afin de la comparer avec l'entrée du comparateur proportionnel. Cela peut être obtenu à partir des commandes suivantes (la fréquence d'échantillonnage était de 4 ms) :

```
>> d_int = cumsum(d_in)*0.004 ;
```

Graphiquement on remarque une correspondance marquée entre les courbes suivantes :

```
>> plot(d_int);
>> hold on;
>> plot(p_in*32/1000,'r');
```

Ce rapport 32/1000 correspond partiellement aux documents du constructeur 4/1000.

Caractérisation de la sortie du correcteur dérivateur

En réutilisant le modèle de caractérisation du correcteur proportionnel (Figure 2), il est possible de régler le gain dérivé à 10. L'entrée du correcteur dérivée (en jaune) et sa sortie (en violet) sont représentés dans la Figure 7.

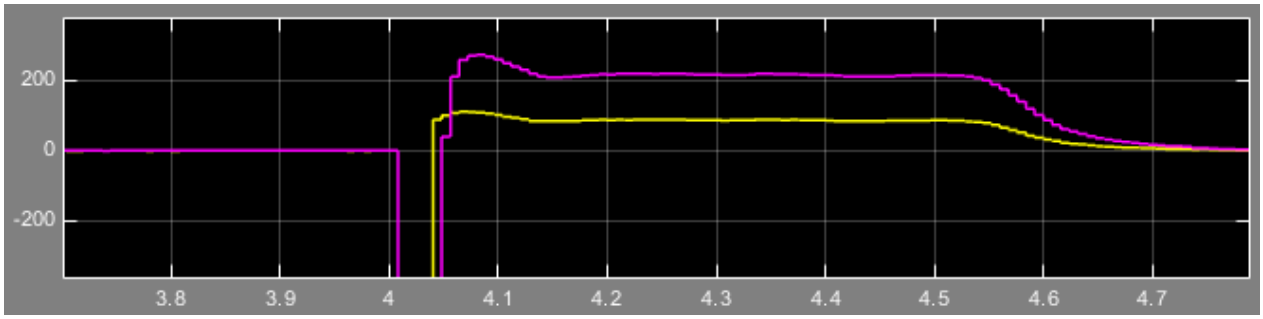


Figure 7 Entrée et sortie du correcteur dérivé

Le gain entre l'entrée et la sortie au cours de ce plateau s'établit à 2,5 pour un gain affiché de 10. Par conséquent le gain réel du correcteur dérivé sera d'un quart du gain réglé dans les paramètres de l'actionneur.

Caractérisation de la commande PWM

Le signal PWM correspond à la commande du hacheur de puissance.

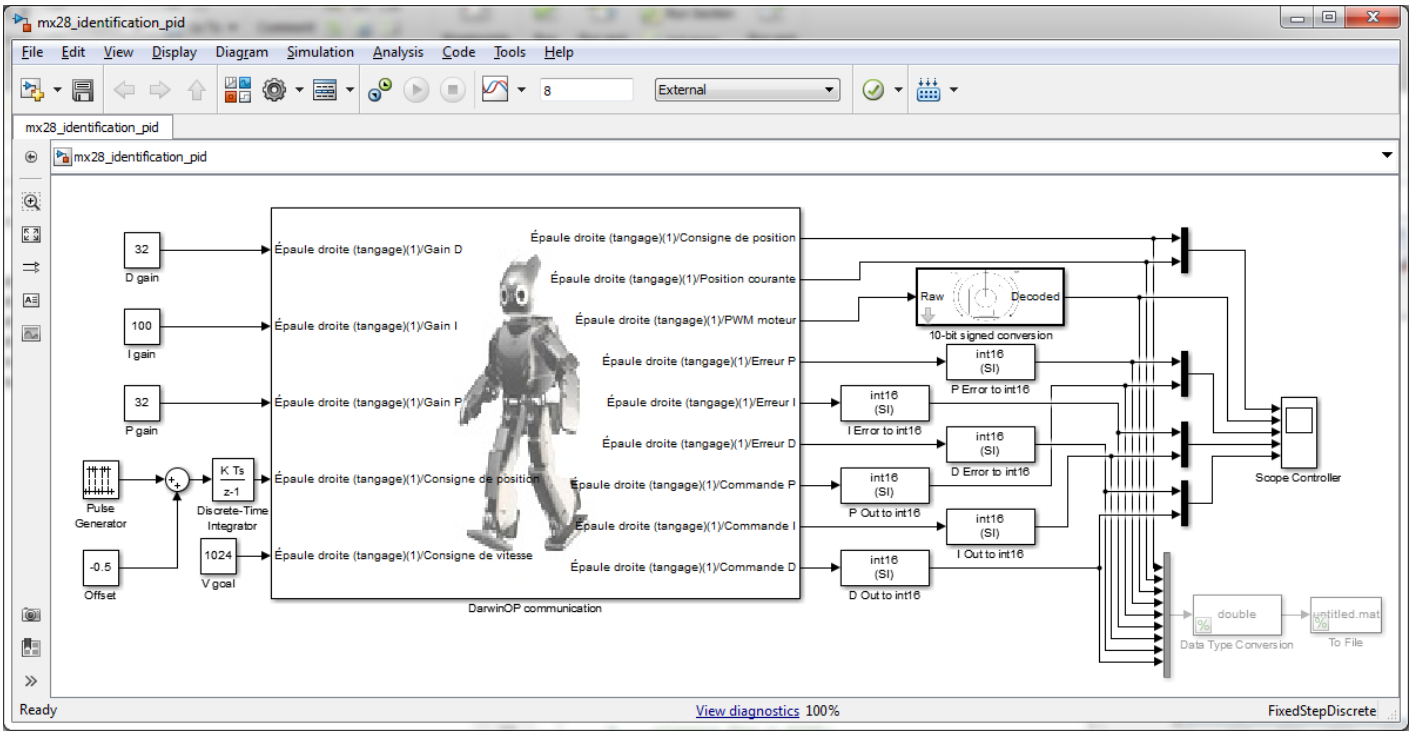


Figure 8 Modèle de caractérisation du PID complet (MX28_identification_PID.slx)

A tout instant, il est possible de reconstruire le signal de la PWM à partir des 3 sorties des correcteurs :

```
>> pwm_rec = p_out+i_out+d_out;
```

La différence entre cette estimation et la PWM mesurée est minimale (par rapport à la valeur de ce signal de l'ordre de plusieurs centaines) :

```
>> diff = pwm-double(my_rec);
>> [min(diff) max(diff)]
ans =
     0     1
```

La Figure 9 représente graphiquement les différents signaux mesurés dans le correcteur (ces graphes représentent les mêmes grandeurs que ceux de la Figure 3).

Figure 9 Caractérisation de la commande d'un correcteur PID

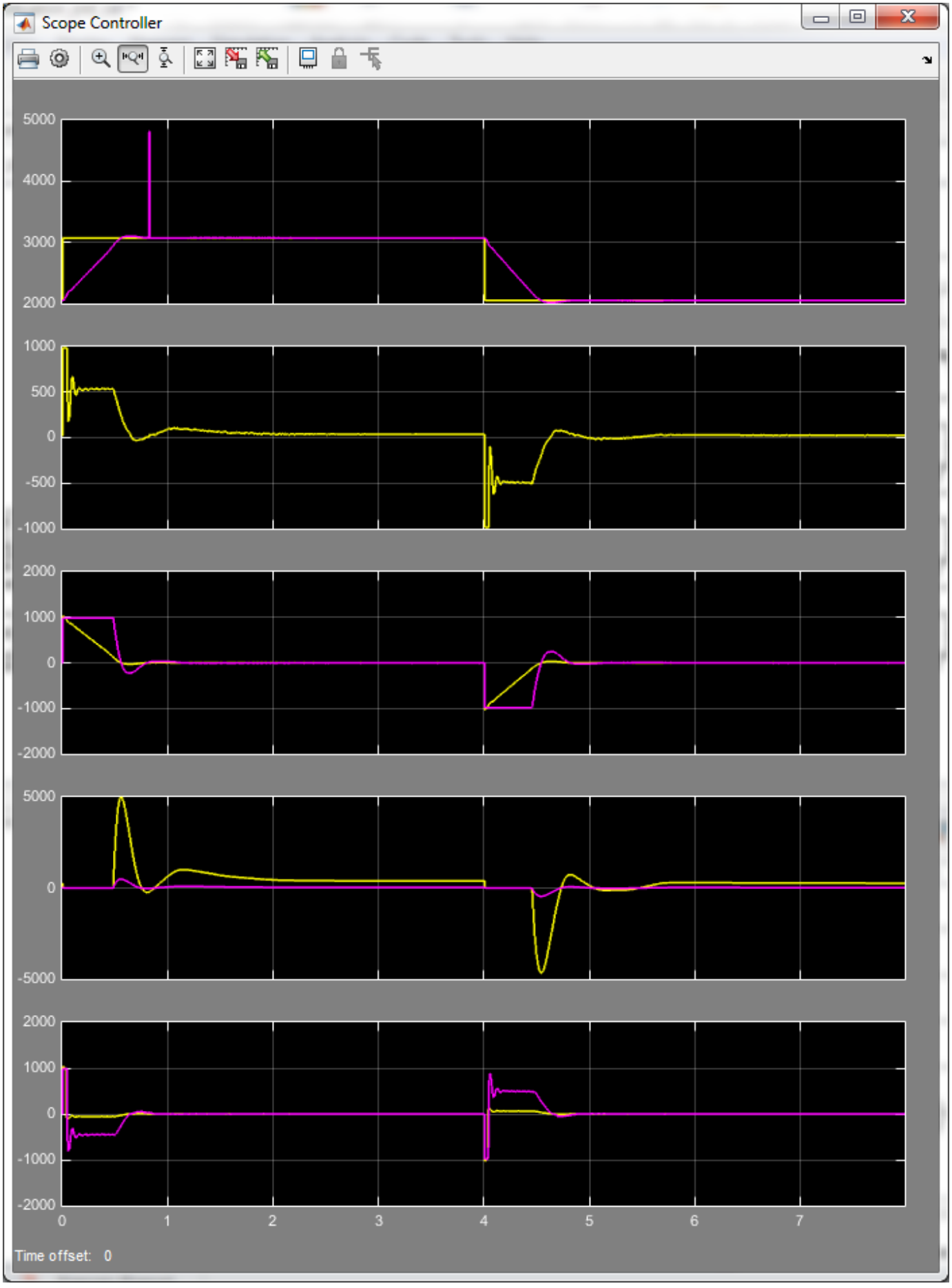
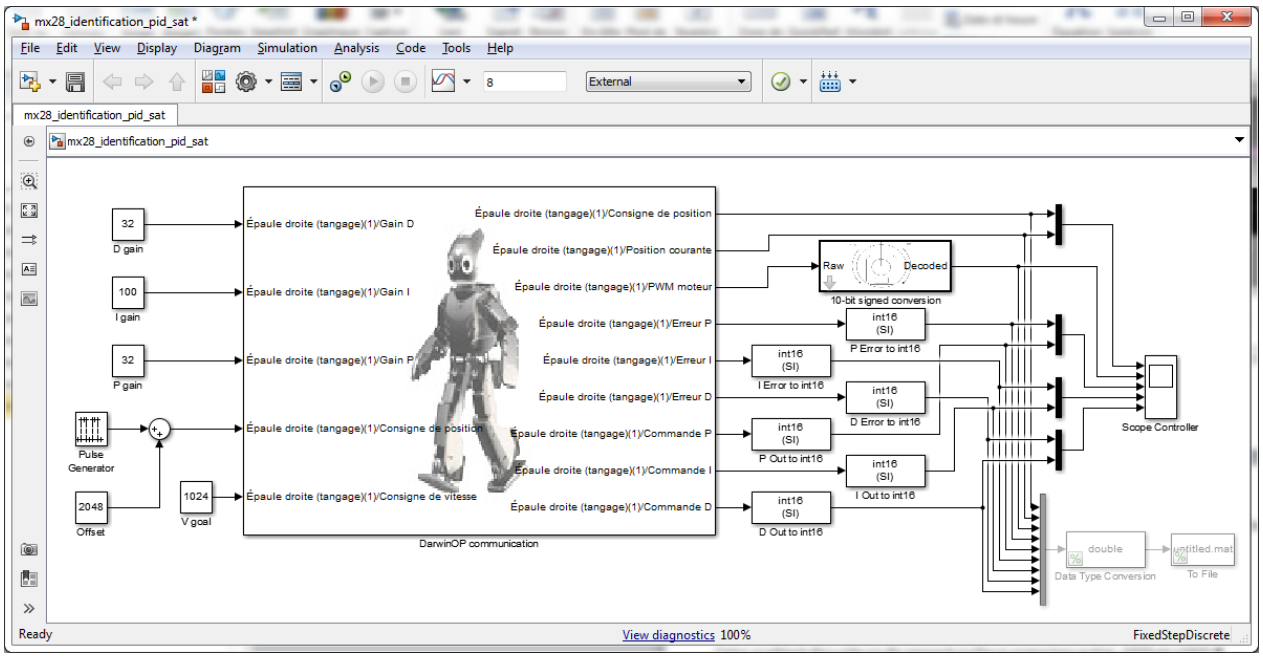


Figure 10 Caractérisation de la commande d'un correcteur PID avec saturations

Cette précédente caractérisation est réalisée sans que l'actionneur atteigne une de ses limites de vitesse ou de couple.



On remarque une saturation dans les transitoires de déplacement postérieurs aux changements de consignes (Figure 10). Cette Figure 10 représente les mêmes grandeurs que la Figure 3. Ces valeurs extrémales sont les suivantes :

```
>> [min(pwm) max(pwm)]
ans =
       -1023        1023
```

Celui-ci admet des valeurs de rapport cyclique comprises entre -1023 et +1023.

Caractérisation des capteurs

Les mesures sont effectuées par l'intermédiaire du seul capteur de position. La vitesse est obtenue par dérivée temporelle de cette position. La charge est estimée d'après les caractéristiques mécaniques et électriques du moteur.

Capteur de position

La mesure de position est la mieux documentée, elle est réalisée par un capteur de position (magnétique – basé sur l'orientation d'un aimant radial sur l'axe de sortie). La position est relevée avec une précision de 4096 points pour un tour.

Capteur de vitesse

La vitesse mécanique de l'axe mécanique de sortie est obtenue à partir de la position (dérivation temporelle). Le protocole expérimental présenté en Figure 11. La consigne de vitesse à la valeur numérique 400 et la période d'échantillonnage est 10 ms. Le facteur d'échelle de la dérivation pour obtenir la vitesse de rotation peut-être obtenu à partir de la dérivation idéale de la position relevée.

```
>> position = ScopeData.signals(1).values(:,2);
>> vitesse = ScopeData.signals(2).values(:,1);
>> vitesse_ideale = [diff(position);0]/0.01;
>> vitesse_gain = vitesse ./ vitesse_ideale;
```

Un tracé de ce gain indique un rapport de $\frac{1024}{1000} \times \frac{1}{8}$ entre la vitesse relevée (champ « Present Speed » de l'actionneur) et la dérivée de la position.

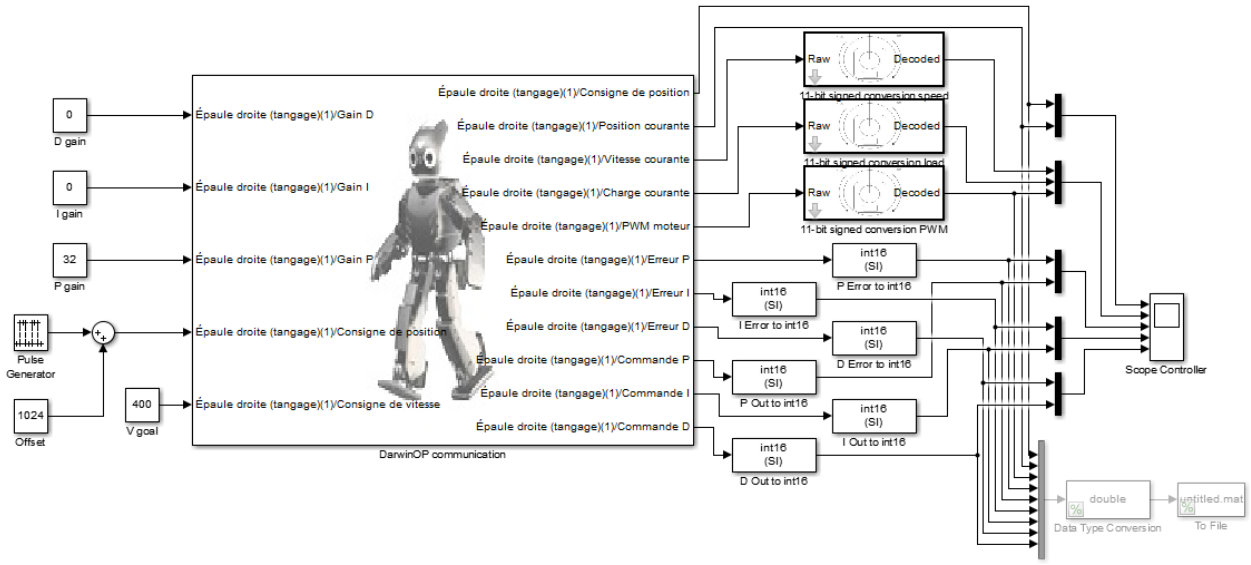


Figure 11 Protocole expérimental de la mesure de vitesse et de charge

Capteur de charge

L'actionneur MX-28 ne dispose pas de capteur de courant consommé par le moteur. L'indication de charge champ « Present Load » est obtenu d'après le modèle de fonctionnement du moteur à courant continu. La première relation lie la tension d'alimentation $U(t)$ à la forme électro-motrice $E(t)$ et au courant consommé par l'induit du moteur noté $I(t)$, supposé constant. Les paramètres R et L désigne habituellement la résistance et l'inductance équivalentes de l'induit.

$$U(t) = E(t) - R \times I(t) - L\frac{dI(t)}{dt} \sim E(t) - R \times I(t)$$

Les grandeurs électriques internes du moteur ($E(t)$ et $I(t)$) sont liées à son comportement mécanique par les relations de conversion électromécanique avec Ω(t) la vitesse de rotation et $C(t)$ le couple délivré par la machine. Les paramètres K_C et K_V désignent respectivement la constante de couple et de flux de la machine :

$$C(t) = K_C I(t)$$

$$E(t) = K_V \Omega(t)$$

Le moteur est connecté à un hacheur commandé en rapport cyclique $PWM(t)$ et alimenté par la tension continue V_{DC} (mesurée à 11.7 V) :

$$U(t) = \frac{PWM(t)}{1024} \times V_{DC}$$

Par combinaison de ces différentes équations :

$$\frac{PWM(t)}{1024} \times V_{DC} = K_V \Omega(t) - R \times \frac{C(t)}{K_C}$$

On peut remarquer que toutes les grandeurs $PWM(t)$ et $\Omega(t)$ ainsi que toutes les constantes V_{DC}, K_V, R et K_C sont connues du concepteur. La mesure de la charge $C(t)$ peut donc être obtenue par combinaison linéaire de la vitesse de rotation $\Omega(t)$ et du rapport cyclique de commande $PWM(t)$.

```
>> vitesse = ScopeData.signals(2).values(:,1);
>> charge = ScopeData.signals(2).values(:,2);
>> pwm = ScopeData.signals(2).values(:,3);
```

Un tracé de ces trois grandeurs est présenté en Figure 12.

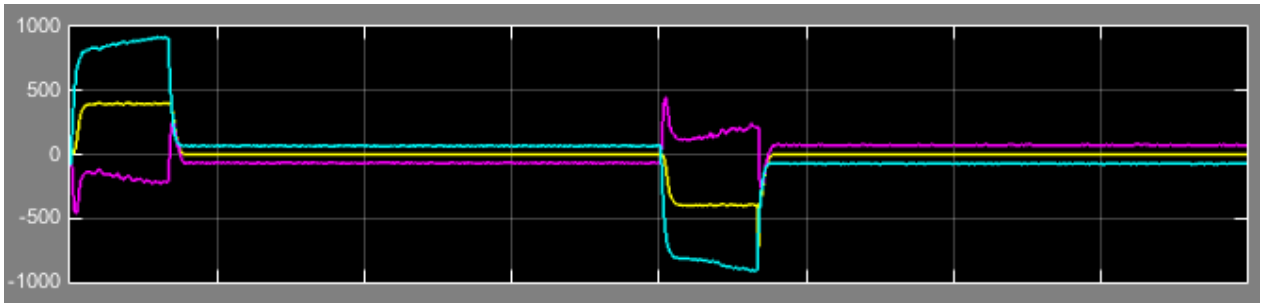


Figure 12 Tracés de la vitesse (jaune), de la charge (violet) et de la PWM (cyan)

La résolution de ce système linéaire à deux inconnues (les coefficients de pondération de $PWM(t)$ et de $\Omega(t)$ permettant d'obtenir la charge) peut être obtenue par déduction. En dehors des déplacements, la charge retournée correspond à l'opposé de la PWM. Le facteur d'échelle sur la vitesse peut-être observé sur les paliers (pendant les déplacements lorsque la vitesse est non-nulle) de la courbe suivante :

```
plot((charge+pwm)./vitesse)
```

On obtient donc la relation $C(t) = 1{,}75 \times \Omega(t) - PWM(t)$

Caractérisation du générateur de trajectoires

Lorsqu'une consigne de vitesse est donnée pendant un déplacement, une trajectoire est utilisée comme consigne de position. Celle-ci correspond à une droite de positions au cours du temps dont le coefficient directeur est issu de la consigne en vitesse.

Les derniers relevés (protocole de la Figure 11) ont été obtenus pour une consigne en vitesse de 400. On rappelle que le signal d'erreur à l'entrée du correcteur PID est égal à la différence entre la consigne de position et la position mesurée. Par conséquent, la consigne de position utilisée peut être retrouvée en ajoutant l'erreur à la position mesurée.

```
>> position = ScopeData.signals(1).values(:,2);
>> erreur = double(ScopeData.signals(3).values(:,1));
>> plot(position+double(erreur))
>> hold on
>> plot(position,'r')
```

Une observation de cette consigne et de cette position mesurée (Figure 13) laisse présager une figure classique d'erreur de trainage finie d'un système d'ordre 1[8] bouclé.

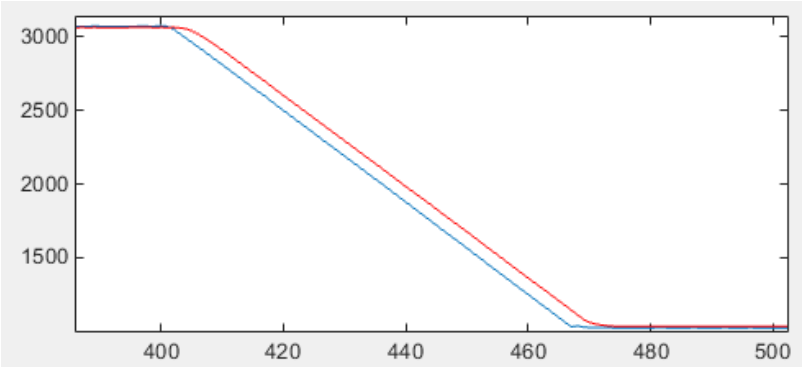


Figure 13 consigne de position (bleu) et position mesurée (rouge)

[8] La fonction de transfert $H(p) = \frac{\theta(p)}{U(p)}$ du moteur à courant continu a un pôle nul.

Cette consigne en position est déterminée à partir de la consigne en vitesse. Il est possible de revenir à la dérivée idéale de cette position et de déterminer sa relation avec la consigne en vitesse.

```
>> vitesse_consigne = 400;
>> vitesse_ideale = [diff(position);0]/0.01;
>> plot(vitesse_ideale/vitesse_consigne);
```

Ce dernier tracé laisse apparaître un gain de $8 \times \frac{1000}{1024}$ entre la vitesse (dérivée de la position) et la consigne de vitesse donnée dans le champ « Moving Speed ». Autrement dit, la consigne de vitesse est déterminée à partir de la position actuelle à partir de laquelle est déterminée une traînée de pente $8 \times \frac{1000}{1024} \times V_{consigne}$.

Caractérisation de la limitation en couple

Le champ « torque limit » agit comme une saturation du rapport cyclique de commande du hacheur à cette limite (Figure 14).

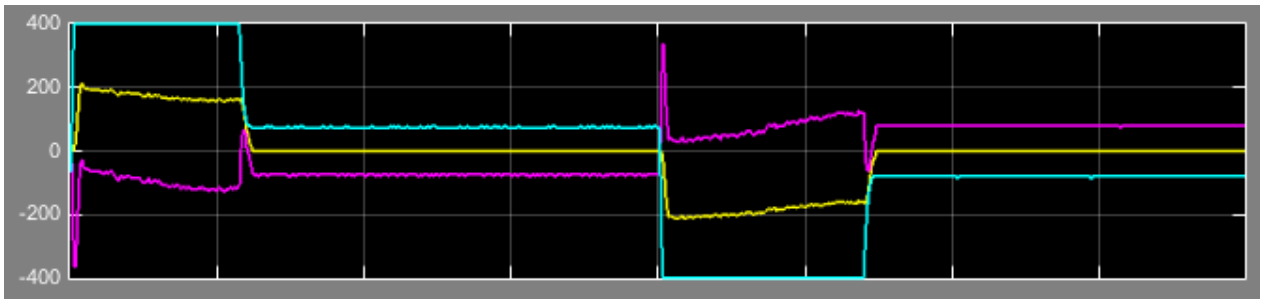


Figure 14 Vitesse de rotation (jaune), indication de charge (violet) et PWM (cyan) pour une limitation du couple à 400

Modèle automatique du MX28

Une fois les caractérisations du correcteur PID effectuées, il est possible d'obtenir une modélisation plus fidèle de l'actionneur. Cette modélisation peut assez facilement être saisie dans Matlab Simulink (modèle MX28_simple.slx représenté en Figure 15).

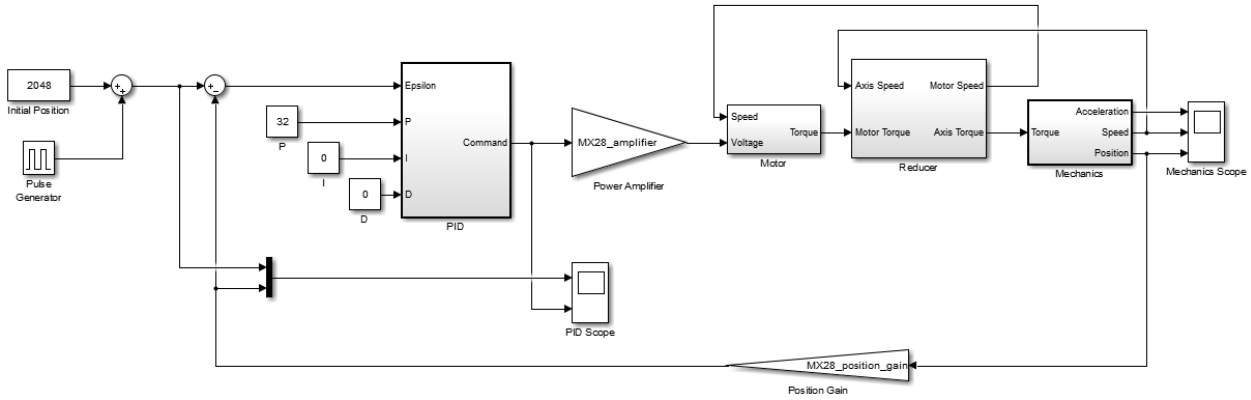


Figure 15 Modèle automatique simple de l'actionneur MX28

Correcteur PID

Le correcteur PID identifié permet d'obtenir un signal de commande (rapport cyclique entre -1023 et +1023) à partir d'un signal de différence « epsilon ». Il est paramétré par trois constantes P, I et D, paramétrables par l'utilisateur spécifiant respectivement le gain proportionnel, le gain intégral et le gain dérivé.

La structure interne du correcteur PID (Figure 16) a été définie à partir de la documentation et des identifications précédentes.

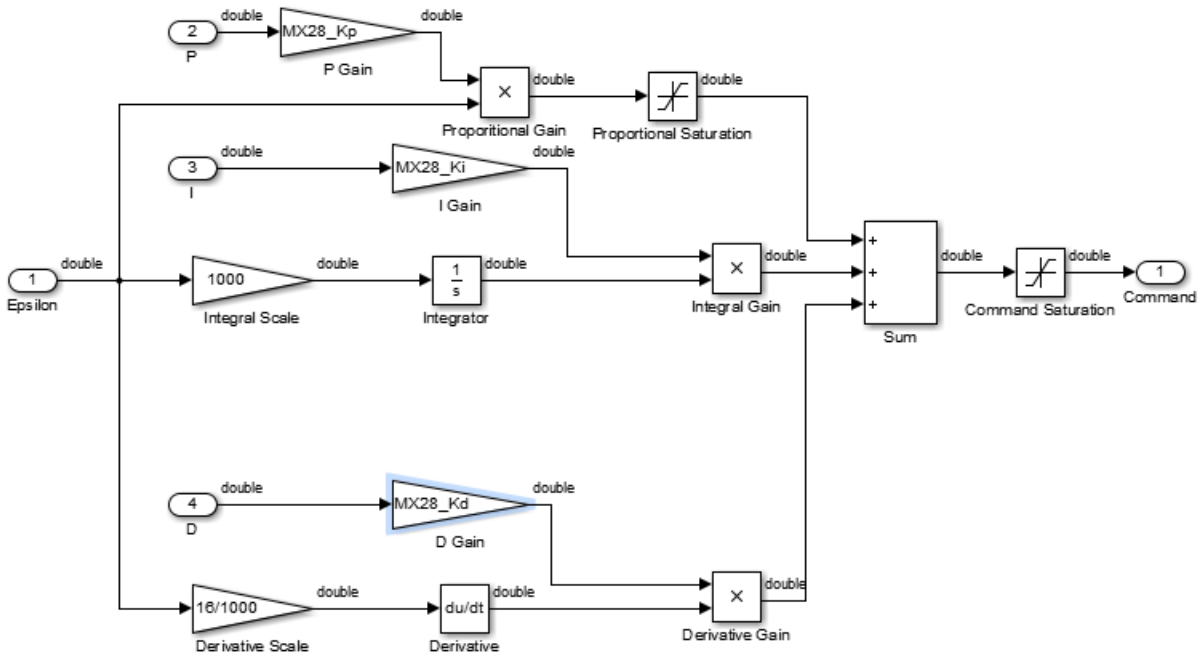


Figure 16 Détails du correcteur PID

Sur la voie « proportionnelle » (en haut), l'erreur « epsilon » est amplifiée par le gain « P » pondéré d'un rapport $MX28\_Kp = {}^{1}/_{4}$ (contre ${}^{1}/_{8}$ annoncé dans la documentation). Le résultat de ce calcul est ensuite saturé à +/-987.

Sur la voie « intégrale », l'erreur est intégrée avec un gain 1000 avant d'être amplifiée par le gain « I » pondéré d'un rapport $MX28\_Ki = {}^{1}/_{1024}$.

Sur la voie « dérivée », l'erreur est dérivée avec un gain 16/1000 avant d'être amplifiée par le gain « D » pondéré d'un rapport $MX28\_Kd = {}^{1}/_{8}$.

Il manque dans ce modèle simplifié la prise en compte des saturations dans la remise à zéro de l'intégrateur. Plusieurs optimisations sont également possibles pour regrouper les gains des branches « intégrale » et « dérivée ».

La pondération complète des coefficients du correcteur selon la branche « intégrale » est donc ${}^{1000}/_{1024}$ (contre ${}^{1000}/_{2048}$ annoncé dans la documentation) et selon la branche « dérivée » ${}^{16}/_{1000} \times {}^{1}/_{8} = {}^{2}/_{1000}$ (contre ${}^{4}/_{1000}$ annoncé dans la documentation).

La saturation en sortie du correcteur PID est paramétrée par le champ « Torque limit ».

Hacheur

Le hacheur 4 quadrants est modélisable comme un convertisseur statique de puissance délivrant pour un rapport cyclique de 100% (=1023) la tension d'alimentation de l'actionneur au moteur. Le moteur est alimenté en inverse pour les rapports cyclique négatif.

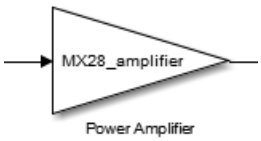


Figure 17 Modélisation du hacheur

Du point de vue modélisation, ce comportement s'apparente à un simple gain de rapport $MX28Amplifier = {}^{\mathrm{Valim}}/_{1023}$ (Figure 17).

Moteur

Le moteur à courant continu est modélisable classiquement en automatique à l'aide des paramètres électriques équivalents (résistance et inductance d'induit) ainsi que par les constantes de conversions électromécaniques issues des caractérisations (Figure 18).

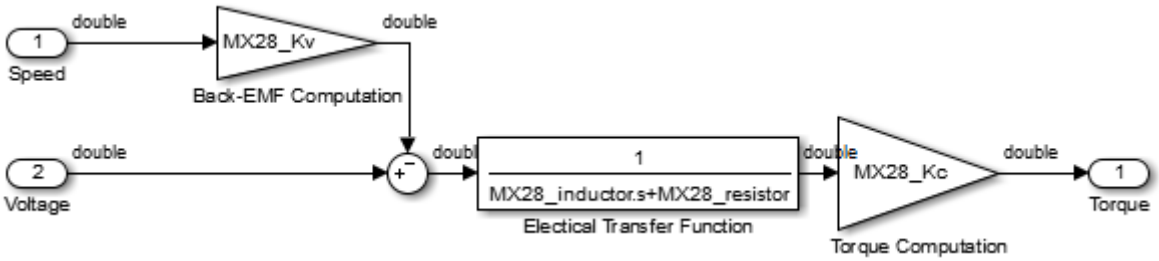


Figure 18 Modélisation du moteur à courant continu

La partie mécanique sera modélisée en sortie du réducteur. On retrouve donc dans ce sous-système uniquement :

- la constante de flux convertissant la vitesse de rotation (en $rad \cdot s^{-1}$) en force électromotrice (en V). Cette constante de conversion est égale à $9{,}926 \times 10^{-3}\ V \cdot rad^{-1} \cdot s$;
- la fonction de transfert électrique convertissant la chute de tension dans les éléments parasites en courant absorbé par la machine. Les constantes électriques sont égales aux valeurs suivantes $MX28\_inductor = 206\ \mu H$ et $MX28\_resistor = 8.6\ \Omega$;
- la constante de couple convertissant le courant absorbé par le moteur (en A) en couple électromécanique (en $N \cdot m$). Cette constante de conversion est égale à $9{,}92 \times 10^{-3} N \cdot m \cdot A^{-1}$.

Réducteur

Le réducteur est modélisé comme une simple transformation de couple et de vitesse entre l'entrée (moteur) et la sortie (axe). Son rendement est supposé unitaire. Les constantes de conversion sont des gains $MX28\_reduction = 192{,}6$ (Figure 19).

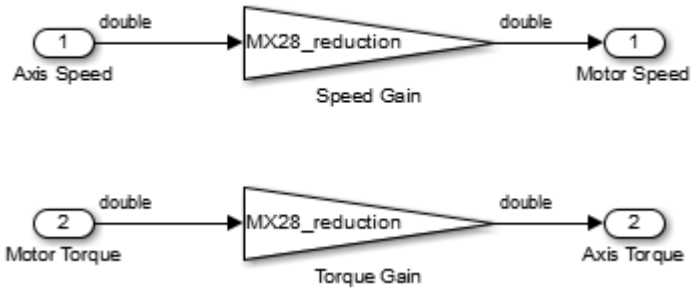


Figure 19 Modélisation du réducteur

Sortie mécanique

La partie mécanique est actionnée en couple par le moteur (Figure 20), les données utiles en sortie sont l'accélération, la vitesse et la position angulaire (respectivement en $rad \cdot s^{-2}$, $rad \cdot s^{-1}$ et rad).

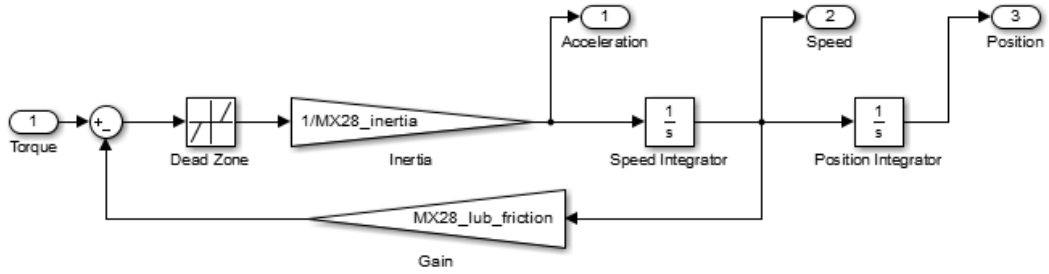


Figure 20 Modélisation de la partie mécanique

L'inertie du moteur et du train d'engrenages ramené à l'axe de sortie vaut $MX28\_inertia = 3{,}3 \times 10^{-3}\ kg \cdot m^2$. Si d'autres charges sont connectées à l'actionneur, il est nécessaire de modifier ce gain en conséquence. Les intégrateurs de vitesse et de position ont des valeurs initiales fixées respectivement à $MX28\_initial\_speed = 0\ rad \cdot s^{-1}$ et $MX28\_initial\_position = \pi\ rad$.

La constante de couple de frottement visqueux est supposée nulle $MX28\_lub\_friction = 0\ N \cdot m \cdot rad^{-1} \cdot s$ ainsi que la constante de frottement sec à l'intérieur du bloc « dead zone » $MX28\_dry\_friction = 0\ N \cdot m$. Les frottements secs et visqueux ne sont actuellement pas pris en compte dans ce modèle.

Capteur de position

Le capteur de position est assimilable à un simple gain convertissant des radians en l'unité d'angle interne de l'articulation (valeur numérique de 0 à 4095). Ce gain (Figure 21) vaut $MX28positiongain = {}^{4096}/_{2 \times \pi}$

Figure 21 Modélisation du capteur de position