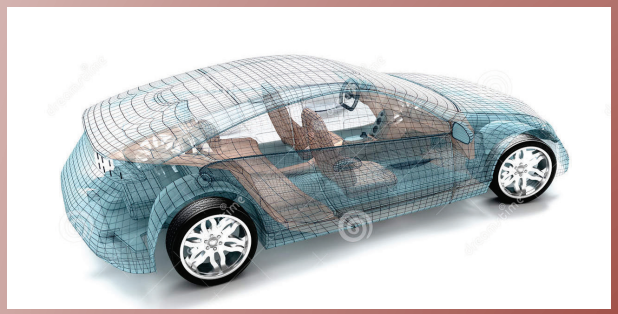
Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB – Simulink (R2015b)

pour l'étudiant et l'ingénieur

Deuxième édition

Introduction au Model Based Design

- *MATLAB*
- *Simulink*
- *Simscape*
- *SimMechanics*
- *SimHydraulics*
- *SimElectronics*
- *Stateflow*



The MathWorks

Ivan LIEBGOTT

Modélisation et Simulation
des Systèmes Multi-Physiques
avec MATLAB/Simulink R2015b
pour l'étudiant et l'ingénieur
Introduction au Model-Based-Design

Auteur : Ivan LIEBGOTT

Professeur de Chaire Supérieure en Classes Préparatoires aux Grandes Ecoles
Lycée des Eucalyptus, Nice

Ingénieur diplômé de l'Institut National Supérieur des Sciences Appliquées (INSA) de Lyon
MASTER en conception de structures Aéronautiques et Spatiales
Agrégé de Sciences de l'Ingénieur
Ancien élève de l'ENS de Cachan

ivan.liebgott@gmail.com Rejoignez-moi sur LinkedIn

Ce livre a été créé pour être librement partagé avec la communauté des utilisateurs de MATLAB.
Vos remarques et vos suggestions seront les bienvenues et me permettront de faire évoluer et d'améliorer cet ouvrage.

Ivan LIEBGOTT 2016

Préface :

Les ingénieurs sont au cœur du processus de conception des systèmes complexes et doivent chaque jour relever les défis de la compétitivité, de l'innovation et de la performance. Cela ne peut se faire sans l'intégration de processus industriels structurés, ni sans la maitrise des outils modernes de modélisation et de simulation. A chaque étape du cycle de conception, les méthodes mises en œuvre doivent permettre de baisser les coûts, de réduire le risque d'erreur et d'en minimiser les impacts.
Au cœur de ce processus, la modélisation et la simulation numérique jouent un rôle majeur et permettent aux ingénieurs d'anticiper, de comprendre et de vérifier les analyses qu'ils mènent tout au long du projet.

Les démarches industrielles standards, comme le cycle en V intègrent pleinement la simulation numérique au travers de méthodes associées comme le « Model Based Design » (conception basée sur le modèle). Les outils modernes de simulation permettent de créer des modèles globaux complexes intégrant tous les comportements du système et prenant en compte l'ensemble des interactions, cette démarche est appelée modélisation multi-physique. Le système réel peut avantageusement être remplacé par son modèle numérique pour réaliser des tests qui auparavant mobilisaient des moyens matériels et humains importants. Cette démarche impose de disposer de modèle validés qui reproduisent fidèlement le comportement des systèmes réels.

Cet ouvrage vous présente une approche de la modélisation multi-physique qui exploite les fonctionnalités et les innovations des logiciels de simulation afin de rendre le processus de modélisation plus rapide et plus efficace. La plate-forme de simulation utilisée est le logiciel MATLAB/Simulink version 2015a.

L'ouvrage propose de donner les clés permettant d'aborder la modélisation globale d'un système en créant le lien entre les méthodes industrielles et les méthodes utilisées dans le cycle de formation des ingénieurs. Il est illustrée par de très nombreux exemples dans différents domaines de la technologie (électrique, hydraulique, mécanique…) et met en évidence l'interconnexion des domaines physiques.

Les fondamentaux de tous les outils nécessaires pour mener cette démarche sont présentés :

- MATLAB
- Simulink
- Simscape
- SimHydraulics
- SimMechanics
- SimElectronics
- Statflow

La démarche propose une introduction à leur utilisation et ne vous rendra expert dans aucun d'eux. Vous pourrez par contre en percevoir tout le potentiel et l'exploiter plus en profondeur en fonction des besoins spécifiques que vous rencontrerez dans votre démarche de modélisation.

Bonne lecture,

Ivan LIEBGOTT

Table des matières

Chapitre 1 : Concepts et stratégies en modélisation

Chapitre 2 : Prise en main de Simscape

Chapitre 3 : Prise en main de MATLAB

Chapitre 4 : Prise en main de Simulink

Chapitre 5 : Prise en main de Stateflow

Chapitre 6 : Prise en main de SimMechanics

Chapitre 7 : L'identification d'un modèle

Chapitre 8: Le contrôle commande avec MATLAB/Simulink

Chapitre 1 : Concepts et stratégies en modélisation

I. Introduction

L'évolution rapide des technologies amène l'homme à concevoir des systèmes de plus en plus complexes dont le comportement ne peut plus être modélisé sans l'aide des logiciels de simulation.
Les interactions entre les composants étant nombreuses, il est primordial d'être en mesure de modéliser le système dans sa globalité en prenant en compte sa structure, son dispositif de commande et son environnement. Les chaînes d'énergie et d'information ne peuvent plus être modélisées séparément mais doivent être intégrées dans un modèle unique. A partir de ce modèle il sera possible d'anticiper le comportement du système réel en phase de conception afin de permettre au système réel de remplir sa fonction.
La modélisation des systèmes complexes est au cœur de la formation des ingénieurs. Cette approche, qui s'appuie sur la décomposition de la complexité fait apparaître des sous-ensembles qui mobilisent chacun des technologies différentes. Le décloisonnement des univers technologiques est donc nécessaire à la mise en place d'une démarche de modélisation pertinente. L'évolution très rapide des logiciels de simulation nous permet maintenant de modéliser toutes les parties d'un système à l'aide d'un logiciel unique. Cette approche multidisciplinaire, permet de gérer les échanges entre les sous-systèmes et de disposer de résultats qui prennent en compte la totalité des phénomènes physiques qui entrent en compte dans le fonctionnement du système. La pertinence des résultats est accrue et la mise en évidence des interactions devient possible. En contrepartie, la maitrise des outils de simulation mobilise un champ de compétences nouveau, plus large et qui impose un processus d'apprentissage structuré. Cette nouvelle approche requiert un outil de modélisation et de simulation à environnement unique, visuel et interactif, validé industriellement et proposant tous les outils de modélisation de la chaîne d'information (régulateur classique, diagramme d'états…) et de la chaîne d'énergie (bibliothèques de composants avancées, maquette CAO intégrée…).

Afin de rendre efficace la démarche de formation, les méthodes et les concepts mis en œuvre doivent être les mêmes dans le cycle de formation des futurs ingénieurs que dans le monde industriel. Nous allons dans un premier temps voir comment il est possible de relier les standards de conception industrielle avec les concepts en vigueur dans les cursus de formation des ingénieurs.

La démarche mise en œuvre industriellement par les équipes d'ingénieurs pour concevoir et développer un système est un processus complexe qui respecte le plus souvent des standards industriels comme le « Cycle en V » et des méthodes associées comme le « Model Based Design ». Ces démarches imposent de disposer de modèles validés que les équipes d'ingénieurs utiliseront pour remplacer le matériel réel dans les phases de test et de réglage. La compétence « modéliser » est donc au cœur du processus d'apprentissage et reste l'une des compétences de base de l'ingénieur. L'évolution des logiciels de modélisation et de simulation a fortement modifié les méthodes d'acquisition de cette compétence pour les futurs ingénieurs. Le processus de modélisation ne se limite plus à écrire les équations différentielles qui régissent le comportement d'un système et à tenter de les résoudre analytiquement dans les rares cas où cela est possible. L'exploitation pertinente des logiciels de modélisation et de simulation permet de mettre en œuvre différentes approches afin de modéliser et d'obtenir rapidement des résultats valides :

- *La modélisation « white box » :*
  Cette approche consiste à écrire les équations et à les résoudre à l'aide de l'outil numérique. Cela permet d'obtenir une première approximation de la structure du modèle. Dans le cas général, les paramètres qui caractérisent le comportement du système ne sont jamais totalement connus. Il faut alors les négliger ou les estimer pour mener à bien la simulation. Les résultats obtenus doivent alors être confrontés à l'expérimentation afin d'optimiser et de valider le modèle.

- *La modélisation « multi-physique » :*
  Cette approche consiste à assembler des composants dont le comportement est déjà modélisé dans le logiciel. Cette méthode ne nécessite pas l'écriture des équations et permet un gain de temps considérable. Les bibliothèques de composants permettent de modéliser dans tous les domaines physiques (mécanique, électrique, hydraulique, thermique…). Il est alors très simple de relier les différents domaines pour obtenir le modèle global d'un système. Il est également possible d'intégrer la maquette 3D de la structure modélisée ce qui permet de prendre en compte très facilement le comportement dynamique et les non-linéarités liées à la géométrie.

- *La modélisation « grise box » :*
  Cette approche vient compléter le modèle « white box » ou « multi-physique » en estimant les paramètres inconnus par comparaison entre la réponse du modèle et la réponse du système réel. Des algorithmes permettent d'automatiser cette tâche et de donner des valeurs aux paramètres inconnus afin d'annuler les écarts entre les performances du modèle et les performances du système réel.

- *La modélisation « black box » :*
  Cette approche, appelée également identification, consiste à associer un modèle mathématique à un comportement mesuré expérimentalement. L'outil numérique permet d'automatiser cette tâche en proposant des modèles mathématiques d'identification complexes. Cette méthode donne uniquement un modèle du comportement global du système sans se préoccuper de l'influence séparée des différents paramètres sur les performances.

La combinaison de ces approches permet de construire un modèle rapidement et de le valider vis-à-vis de l'expérimentation. Cette partie propose la mise en œuvre sur un cas concret d'une approche de type « Model Based Design ». En partant du cahier des charges et des exigences liées à la conception, les différentes étapes du cycle en V seront parcourues en utilisant les outils modernes de modélisation, de simulation et de validation. Les différentes approches de modélisation sont comparées et les avantages et inconvénients des différentes méthodes mis en évidence. La démarche permet de créer un lien entre le processus de formation et la réalité industrielle. Les étapes importantes de la démarche « Model Based Design » et de la détection d'erreur dans la génération de code C sont illustrées (Model in the loop, Software in the loop, Processor in the loop et Hardware in the loop).

II. Industrialisation et cycle de conception d'un système

La démarche mise en œuvre industriellement par les équipes d'ingénieurs pour concevoir un système est un processus complexe qui respecte le plus souvent des standards industriels comme le « Cycle en V » et des méthodes associées comme le « Model Based Design ». La Figure 1 propose une représentation simplifiée du « Cycle en V », adaptée à la présentation des concepts de ce processus.

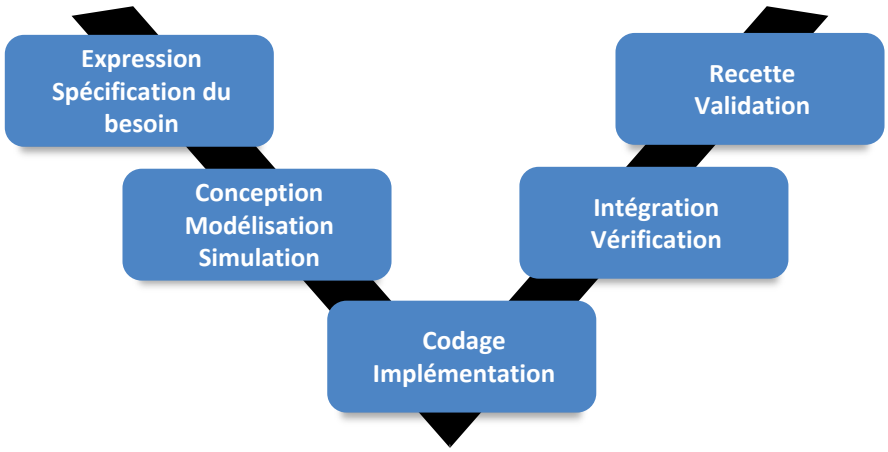


Figure 1 : cycle en V simplifié

A. Les compétences de l'ingénieur

Pour mettre en œuvre cette démarche l'étudiant et l'ingénieur vont mobiliser 6 grandes compétences. Ces compétences sont à la base de la formation des ingénieurs et sont détaillées sur la Figure 2.

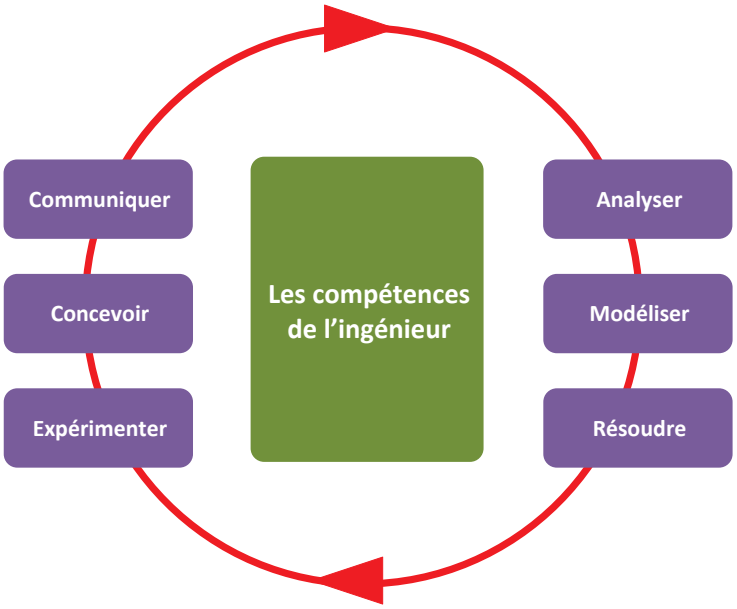


Figure 2 : les compétences de l'ingénieur

B. Le triptyque des performances(*)

La Figure 3 décrit les trois types de performances qui seront considérés et comparées dans la démarche :

- Les performances issues des spécifications du cahier des charges
- Les performances issues de la simulation numérique des modèles
- Les performances issues de la mesure directe sur le système réel à l'aide de capteurs

Trois types d'écart pourront être ainsi formés et analysés :

- L'écart entre les performances spécifiées par le cahier des charges et celles mesurées sur le système réel
- L'écart entre les performances mesurées sur le système réel et les performances issues de la simulation des modèles numériques
- L'écart entre les performances spécifiées par le cahier des charges et les performances issues de la simulation des modèles numériques

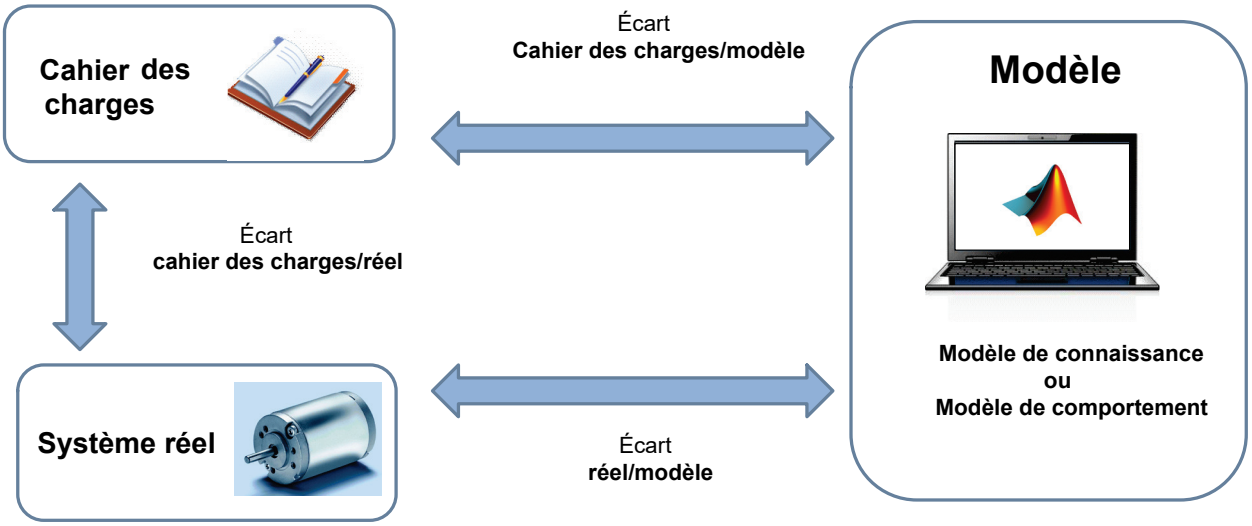


Figure 3 : triptyque Cahier des charges/Système réel/Modèle

(*) Vous pouvez assister au webinar que nous avons réalisé avec Nadia Bedjaoui, qui présente la mise en œuvre de la démarche de l'ingénieur en exploitant les fonctionnalités du logiciel MATLAB/Simulink :

http://fr.mathworks.com/videos/matlab-and-simulink-for-teaching-98093.html

III. Mise en œuvre de la démarche- Introduction au « Model Based Design »

Afin de comprendre les actions à mener dans les différentes phases du cycle en V, nous allons appliquer cette démarche à une problématique simple. Il est conseillé avant d'aborder la partie de l'ouvrage consacrée à la manipulation des outils de simulation et de modélisation, de prendre connaissance de cette partie. Cela permettra au lecteur de prendre du recul sur la démarche de modélisation, de comprendre la place du modèle dans le processus industriel et de percevoir tout l'intérêt de la conception basée sur le modèle appelée plus communément « Model Based Design »

L'exemple choisi propose de reproduire la démarche de conception de la logique de commande de l'asservissement de position angulaire d'un moteur à courant continu. L'objectif est de positionner angulairement l'axe du moteur conformément à une consigne de position imposée en respectant des critères de performances courants comme la précision, la rapidité ou la stabilité.

A. Architecture matérielle du projet

Afin de mettre en œuvre la démarche, la Figure 4 présente l'environnement matériel du projet. Il n'est pas nécessaire de disposer de ce matériel pour comprendre les concepts abordés dans cette partie.

- *Un ordinateur équipé d'un logiciel de modélisation, de simulation et de pilotage :*

Le rôle de l'ordinateur est de servir de support à la conception des modèles numériques, à leur simulation et à la compilation des résultats. L'ordinateur va également servir à créer des interfaces de mesures et de pilotage afin de commander le moteur via la carte de commande du moteur.

- *La carte de commande du moteur :*

Le rôle de la carte de commande du moteur est de piloter le moteur. Elle possède des entrées, pour récupérer les informations issues des capteurs (position angulaire du moteur…) et prendre en compte la consigne issue de l'ordinateur, et des sorties pour fournir une tension aux bornes du moteur et permettre sa mise en rotation. La tension fournie est modulable en fonction de la logique de commande programmée. Dans notre exemple elle fera également office de carte de puissance.

- *Le moteur à courant continu :*

Il est alimenté par une tension continue et délivre une vitesse de rotation. Un capteur de position intégré renvoie à la carte de commande la position angulaire du moteur.

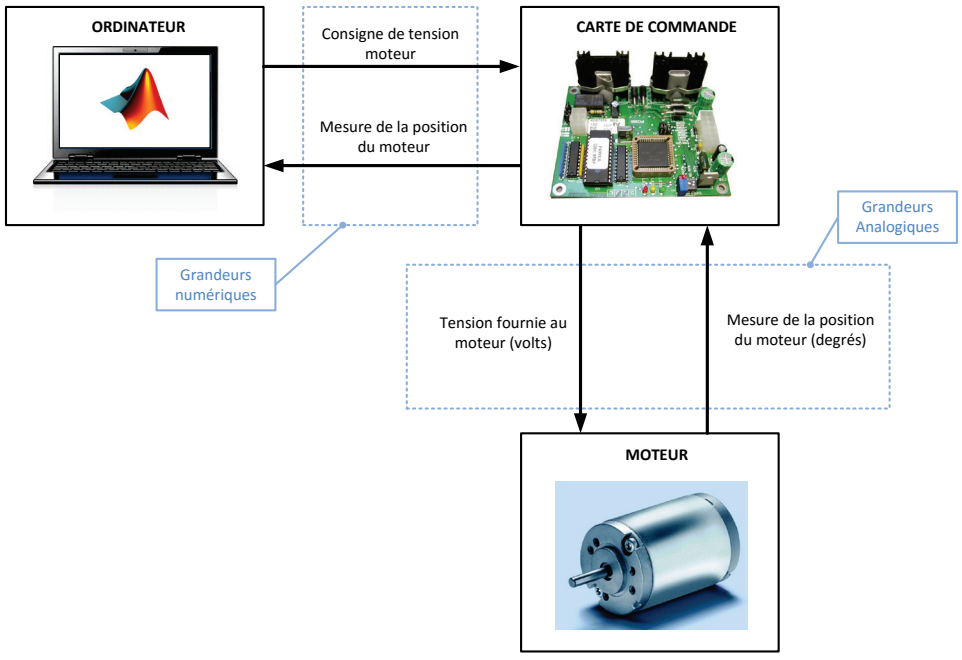


Figure 4 : architecture matérielle nécessaire à la mise en place de la démarche

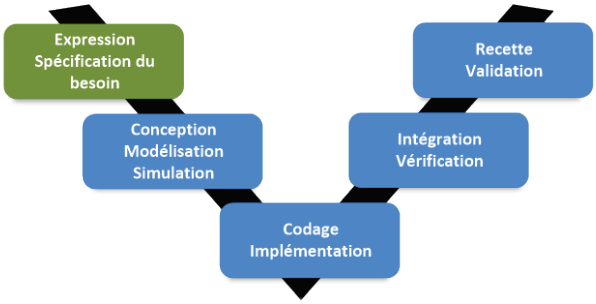


Figure 5 : la phase d'expression et de spécification du besoin

Le cycle démarre par **l'Expression et la Spécification** des besoins du système. Dans cette phase l'ensemble des exigences que doit remplir le système est formalisé. Les critères de performances sont définis et quantifiés. L'ensemble de ces informations est regroupé dans le cahier des charges fonctionnel. Il est analysé par les équipes de concepteur qui devront produire les solutions technologiques adéquates. Le diagramme des exigences de la Figure 6 regroupe les exigences de l'asservissement de position à concevoir.

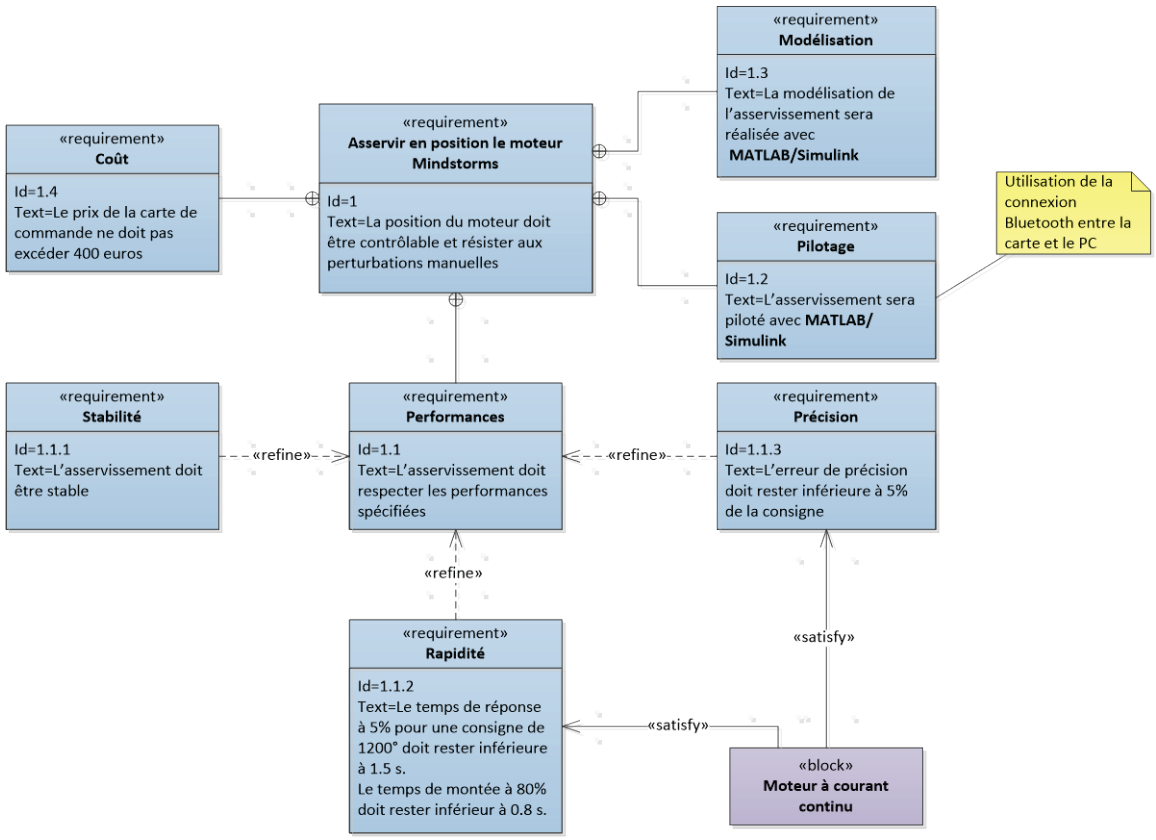


Figure 6 : diagramme des exigences de l'asservissement de position

C. La phase de Conception-Modélisation-Simulation

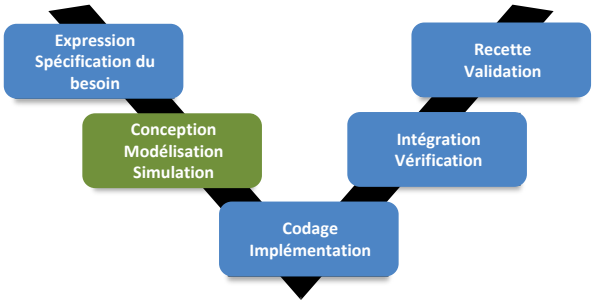


Figure 7 : la phase de Conception, Modélisation, Simulation

La phase de **Conception-Modélisation-Simulation** consiste à inventer, innover et proposer des solutions technologiques pour répondre aux exigences spécifiées. Dans cette phase sont définis, la structure du système, l'interconnexion des domaines physiques (mécanique, électrique, hydraulique, thermique…), les lois de commandes... Cette phase très complexe nécessite de pouvoir anticiper les comportements avant même que le système réel ne soit produit. C'est alors que le travail de modélisation va commencer. Les ingénieurs vont construire des modèles pour tenter de comprendre et d'anticiper les choix technologiques effectués. Cette étape requiert des connaissances théoriques et technologiques de haut niveau dans tous les domaines qui sont abordés. Afin de pouvoir exploiter les modèles et obtenir des résultats, la simulation de ces modèles à l'aide d'outils numériques adaptés doit alors être mise en œuvre.
Industriellement, mener des campagnes d'essais sur du matériel réel est très couteux. Cette activité mobilise des moyens matériels et humains important et impose des contraintes très fortes en termes de délais et de coût. L'un des objectifs majeurs est alors de limiter la durée de ces tests en optimisant la préparation de cette phase. Une solution consiste à remplacer les tests sur le matériel réel par des essais sur des modèles numériques reproduisant le comportement du système réel. Cette méthode appelée « Model Based Design » (conception basée sur le modèle) tend à devenir un standard de conception qui vient enrichir la démarche du cycle en V.
Cette méthode impose de disposer de modèles validés. Les techniques de modélisation et leur maitrise par les équipes d'ingénieurs se retrouvent au cœur de l'activité de conception. L'évolution rapide des fonctionnalités des logiciels de simulation numérique ouvre de nouvelles possibilités de modélisation qui accélèrent le processus de création et de validation des modèles. La résolution numérique est indispensable au processus et le logiciel de simulation devient le partenaire incontournable de cette activité. Sa performance et ses fonctionnalités détermineront souvent la rapidité et l'efficacité du processus de modélisation. Différentes méthodes peuvent être mise en œuvre pour optimiser cette phase.

- La modélisation « white box » :
- La modélisation « multi-physique » :
- La modélisation « grise box » :
- La modélisation « black box » :

Ces différentes approches ne sont pas exclusives dans le processus de modélisation. Elles sont au contraire complémentaires et il est fréquent de les combiner pour mener la démarche avec la plus grande pertinence. La connaissance et la maitrise de ces concepts par les étudiants et les ingénieurs leur permettent d'être compétents dans la démarche de modélisation. Dans tous les cas la confrontation à l'expérimentation et à la mesure permettront de valider le comportement du modèle.

Dans notre exemple, nous aurons besoin de créer le modèle numérique d'un moteur à courant continu afin de réaliser toute la phase de contrôle commande sur un modèle lorsqu'il sera validé.

La modélisation du moteur à courant continu consiste à établir la relation entre la tension aux bornes de l'induit du moteur et sa vitesse de rotation.

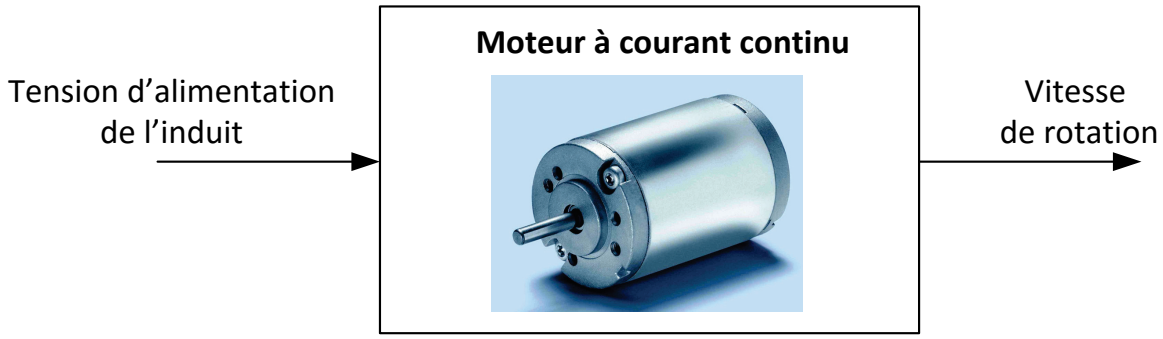


Figure 8 : relation entrée/sortie pour le moteur à courant continu

1. La modélisation « white box »

Les équations électrique, mécanique et de couplage d'un moteur représentent le modèle de connaissance du moteur à courant continu et sont détaillées sur la Figure 9 :

$$\begin{cases} u(t) = e(t) + R\,i(t) + L\dfrac{d\,i(t)}{dt} \\ e(t) = K_e\,\omega_m(t) \\ J\dfrac{d\,\omega_m(t)}{dt} = C_m(t) - C_r(t) - f\,\omega_m(t) \\ C_m(t) = K_m\,i(t) \end{cases}$$

$u(t)$: tension de commande du moteur (V)

$\mathrm{e}(\mathrm{t})$: force contre électromotrice du moteur (V)

$i(t)$: intensité dans le moteur (A)

$C_m(t)$: couple exercé par le moteur $(\mathrm{N.m})$

$C_r(t)$: couple résistant ramené sur l'arbre moteur $(\mathrm{N.m})$

$\omega_m(t)$: vitesse de rotation du moteur $(\mathrm{rad/s})$

R : résistance de l'induit (Ω)

L : inductance de l'induit (H)

J : inertie équivalente ramenée sur l'arbre moteur $(\mathrm{kg.m^2})$

f : paramètre de frottemement visqueux $(\mathrm{N.m.s})$

$\mathrm{K_t}$: constante de couple $(\mathrm{N.m/A})$

$\mathrm{K_e}$: coefficient de force contre électromotrice $(\mathrm{V.s/rad})$

Figure 9 : équations de comportements du moteur à courant continu

Parmi les paramètres qui caractérisent le comportement du moteur, certains sont parfaitement connus et d'autres reste indéterminés. L'inductance L et la résistance R de l'induit sont données dans la documentation technique du moteur, il en va de même pour les coefficients Ke et Kt. Par contre le paramètre de frottement visqueux f n'est pas forcément connu, en particulier si le moteur est ensuite couplé à un mécanisme. Il faudra soit négliger ce paramètre f ou alors en faire une estimation pour espérer obtenir des résultats valides avec le modèle.

Après avoir appliqué la transformée de Laplace à ces équations, il est facile d'obtenir un modèle numérique sous forme de schéma bloc que l'on pourra modéliser et résoudre à l'aide du logiciel de simulation numérique (Figure 10). Ce principe de résolution est basé sur le principe de causalité. Le comportement dynamique d'un système est caractérisé par un bloc contenant, par exemple, la fonction de transfert du système avec une entrée et une sortie. L'information circulant dans les connexions entre deux blocs est un signal numérique orienté. La sortie du bloc est calculée numériquement en déterminant pour chaque pas de calcul la transformation du signal d'entrée imposé par le contenu du bloc. Cette approche largement utilisée industriellement nécessite une parfaite connaissance des lois physiques qui caractérisent le comportement des systèmes.

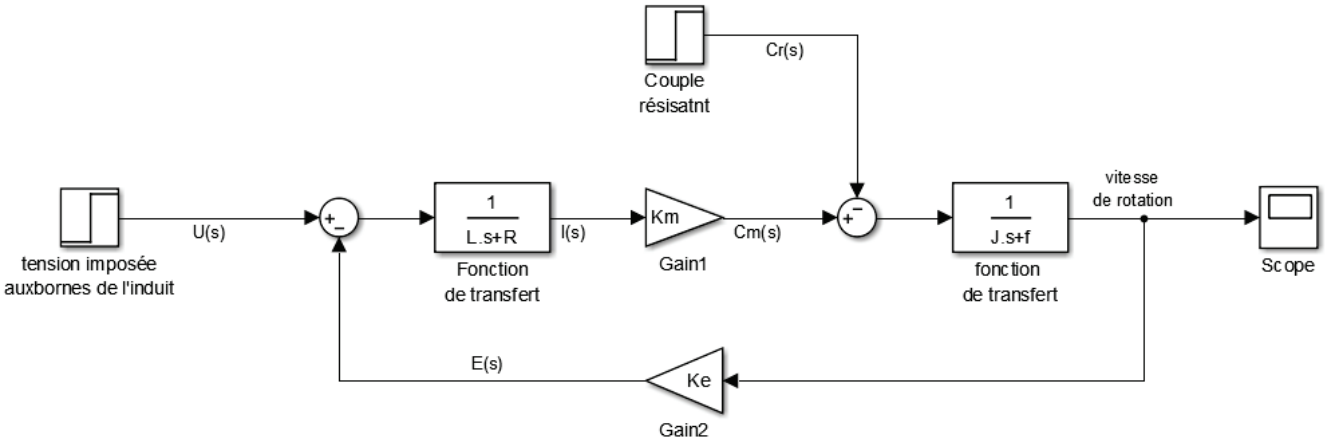


Figure 10 : modélisation sous forme de schéma bloc du moteur à courant continu

Il est maintenant possible de remplacer le système par son modèle comme représenté sur la Figure 11.

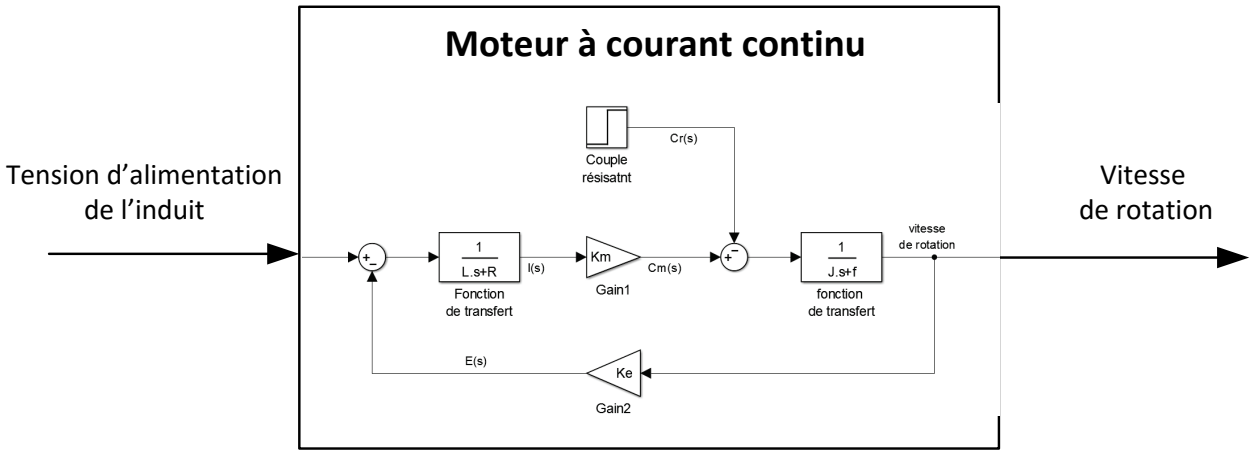


Figure 11 : modélisation « white box » du moteur à courant continu

2. La modélisation « Multi-Physique »

La modélisation multi-physique propose une approche acausale de la modélisation et permet de faire de la modélisation par assemblage de composants issus de bibliothèques prédéfinies. Le comportement physique des composants est pris en compte directement par le logiciel, il est donc possible de modéliser un système sans avoir à écrire les équations différentielles qui caractérise son comportement.
Les connexions entre deux composants ne sont pas orientées, ont une signification physique et transmettent un niveau d'information supérieure aux connexions en modélisation causale. Ces connexions peuvent être un fil électrique (transfert d'information de type courant et tension), un arbre moteur (transfert d'information de type couple et vitesse angulaire), l'extrémité de la tige d'un vérin (transfert d'information de type force et vitesse linéaire)… Le principe de calcul s'appuie sur un bilan de puissance à chaque nœud du modèle et ne repose pas sur le principe de causalité, d'où le nom de modélisation acausale.

La modélisation multi-physique du moteur à courant continu est représentée sur la Figure 12.

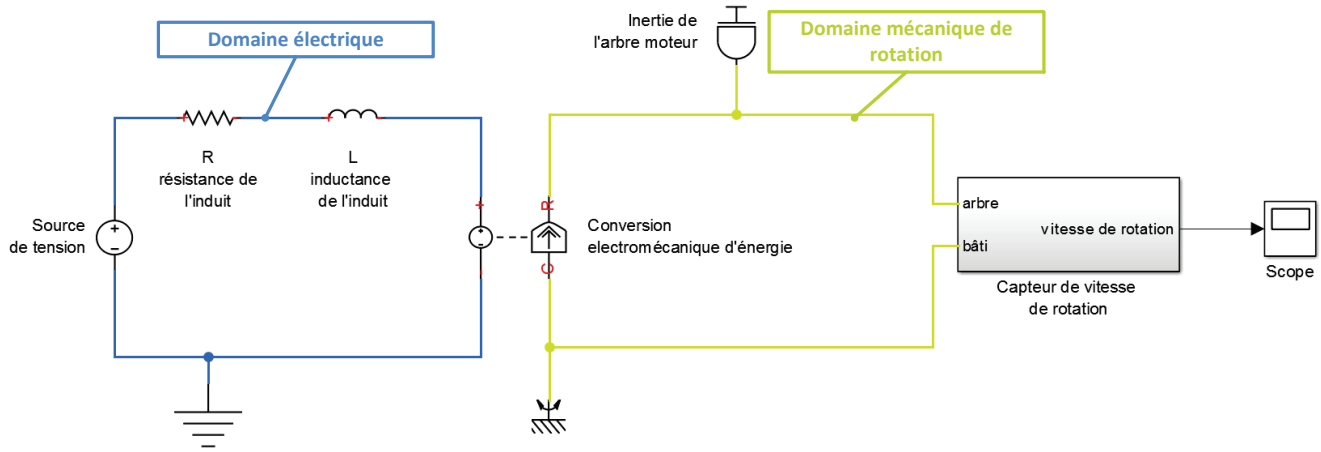


Figure 12 : modélisation par assemblage de composants du moteur à courant continu

Il est maintenant possible de remplacer le système par son modèle comme représenté sur la Figure 13.

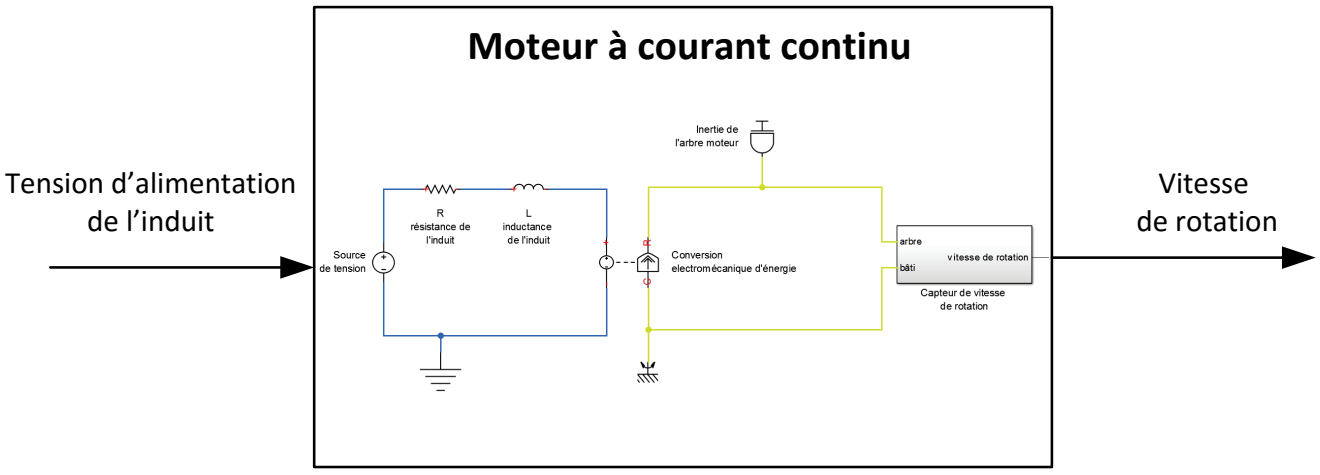


Figure 13 : modélisation multi-physique acausal du moteur à courant continu

3. La simulation des modèles

La simulation et la résolution des modèles par le logiciel de simulation imposent de connaître les valeurs de tous les paramètres numériques. Dans notre exemple tous les paramètres sont connus sauf le paramètre f caractérisant les frottements. Il sera donc considéré comme nulle en première approximation ce qui revient à négliger l'influence des frottements.
Quelle que soit l'approche choisie, de type « white box » ou « multi-physique », les résultats obtenus par simulation numériques sont identiques (Figure 14). Dans les deux cas les phénomènes physiques pris en compte sont les mêmes, la différence réside dans la méthode utilisée pour construire le modèle et la méthode de résolution numérique utilisée qui diffère.

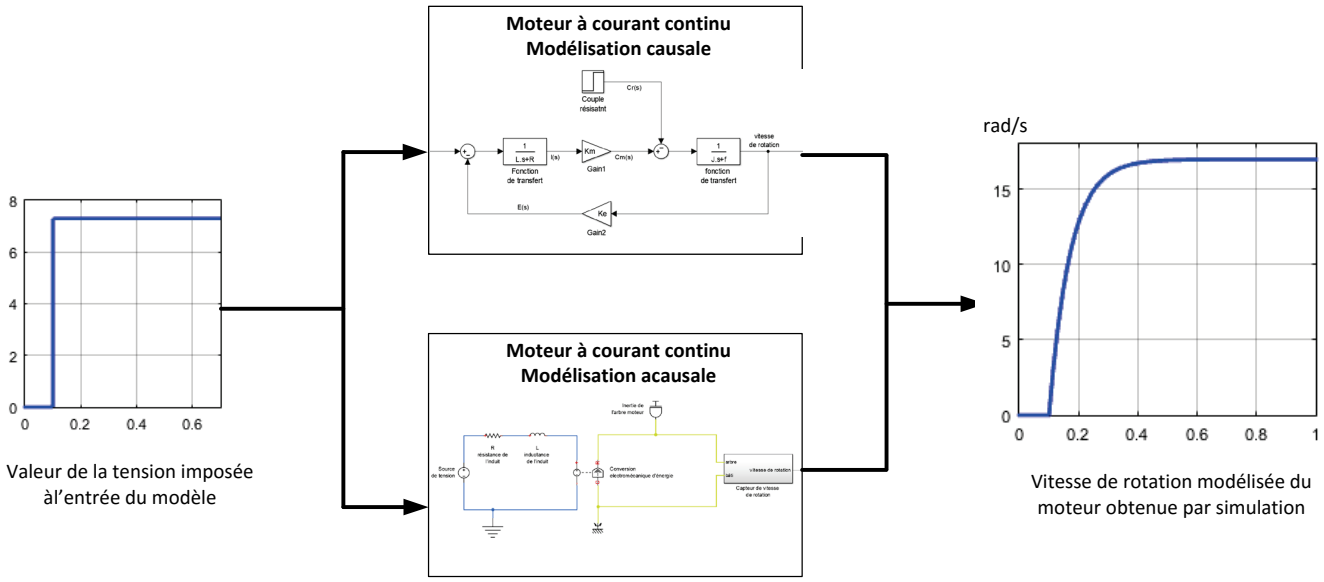


Figure 14 : résolution des modèles et visualisation des résultats

Les résultats obtenus permettent d'avoir une première approximation de l'évolution de la vitesse de rotation du moteur en fonction du temps. Dans le cas général cette approche est très souvent insuffisante et le seul moyen de vérifier la validité du modèle est d'évaluer les écarts entre les performances modélisées et les performances mesurées sur le système réel. En imposant une tension identique aux bornes de l'induit du moteur réel, il est possible de relever l'évolution de la vitesse de rotation réelle du moteur comme le montre la Figure 15.

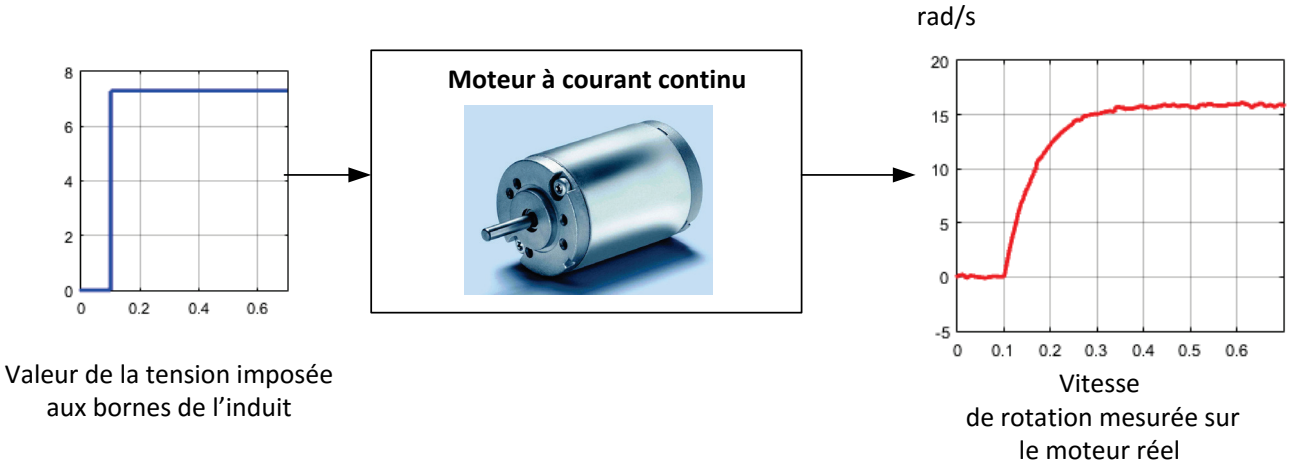


Figure 15 : mesure de l'évolution de la vitesse de rotation du moteur en fonction du temps

4. La comparaison des performances simulées et mesurées

Les performances simulées et mesurées sont alors comparées sur la Figure 16.

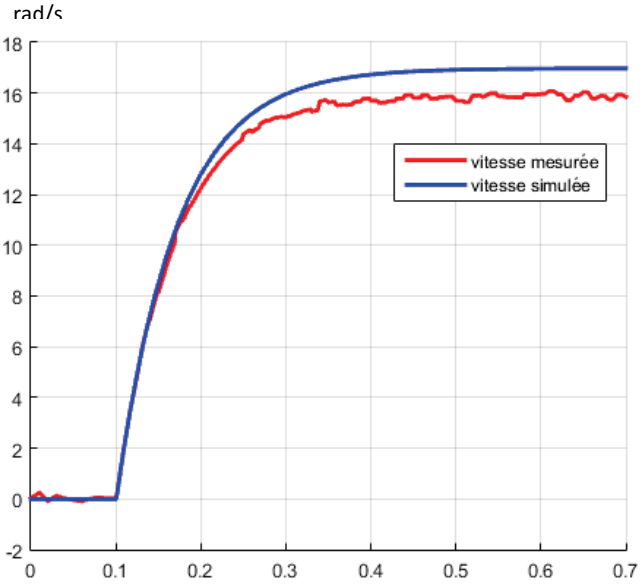


Figure 16 : évaluation des écarts entre les performances simulée et les performances mesurées

Dans le cas général et à ce stade, la mise en évidence d'écarts de comportement entre le modèle et le réel est très fréquent. Il faut alors évaluer ces écarts et trouver des pistes pour améliorer la modélisation en prenant en compte un phénomène nouveau ou en adaptant la valeur de certains paramètres.
Dans notre exemple l'analyse des écarts montre que le modèle n'est pas validé. L'absence de prise en compte des frottements est à l'origine de cet écart. Il faut alors évaluer les paramètres inconnus, ici le paramètre de frottement f. L'approche de type « grise box » va permettre de compléter le processus de modélisation en proposant une méthode de détermination des paramètres inconnus.

5. La modélisation « grise box »

Le principe de cette modélisation est d'estimer les paramètres inconnus du modèle par comparaison entres les performances simulées et les performances mesurées. Un algorithme intégré au logiciel de simulation va effectuer le travail d'analyse des écarts et proposer des valeurs possibles pour les paramètres inconnus permettant l'annulation des écarts. La méthode consiste à effectuer un essai sur le système réel en relevant le signal imposé à l'entrée du système et le signal obtenu à la sortie (Figure 17).

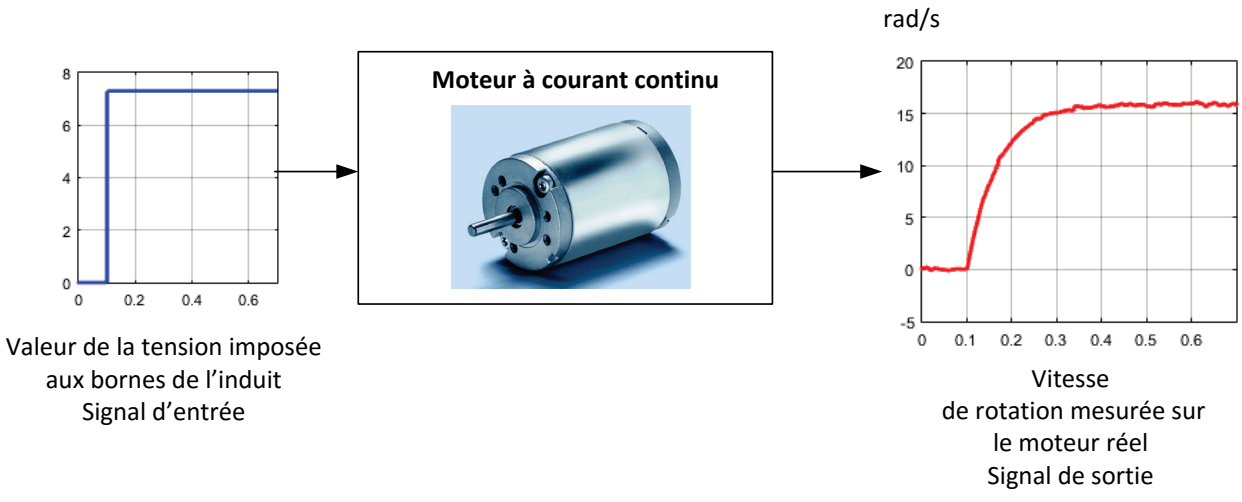


Figure 17 : essai réalisé en vue de l'estimation des paramètres inconnus

Un signal d'entrée identique est alors imposé à l'entrée du modèle et le logiciel va donner des valeurs possibles pour les paramètres inconnus de manière à annuler les écarts entre le réel et le modèle. Le modèle va chercher à atteindre la réponse cible, celle du système réel (Figure 18). Il est possible d'utiliser aussi bien le modèle causal que le modèle acausal pour réaliser cette optimisation.

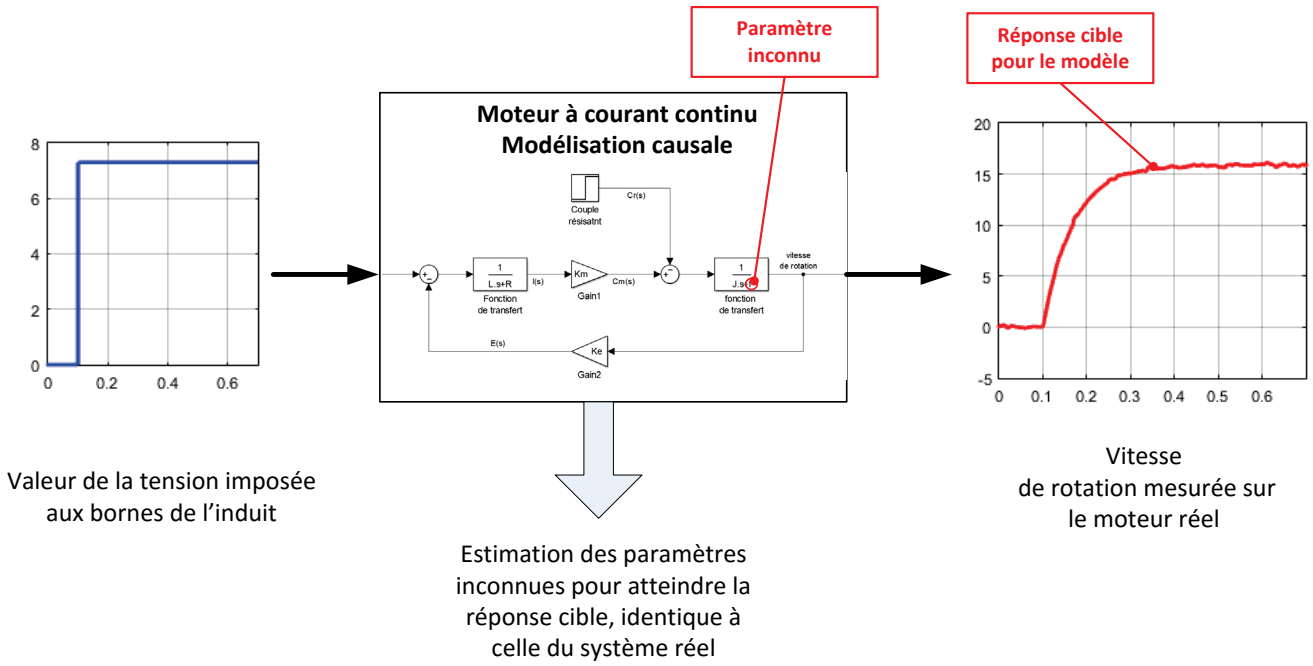


Figure 18 : illustration du principe de l'estimation de paramètre

La Figure 19 montre la réponse cible en rouge que va chercher à atteindre l'algorithme d'estimation de paramètres. La réponse du modèle avant estimation des paramètres est représentée sur la courbe bleue et la réponse du modèle après estimation des paramètres sur la courbe verte. L'intérêt est évident et permet d'obtenir un modèle validé dont le comportement est très proche de celui du système réel. Les écarts entre les performances mesurées et les performances simulées ont été annulés.

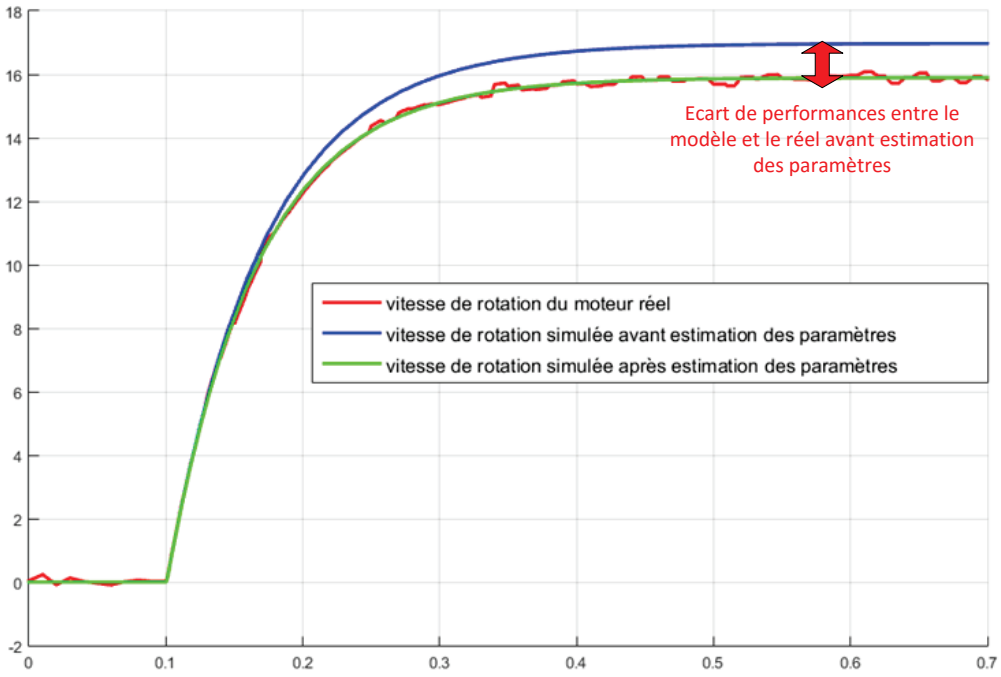


Figure 19 : les réponses du modèle avant et après estimation des paramètres inconnus

A l'issu de cette phase le projet dispose de modèles validés. Il est alors possible de remplacer le matériel (ici le moteur) par son modèle. La mise en œuvre de la phase de conception et de réglage de la commande se fera uniquement à l'aide du logiciel de simulation en travaillant sur le modèle. Les tests sur le matériel réel se feront après que les performances du modèle atteignent les performances souhaitées par le cahier des charges.

Nous pouvons alors utiliser le modèle validé du moteur pour construire l'asservissement en position du moteur à courant continu représenté sur la Figure 20.

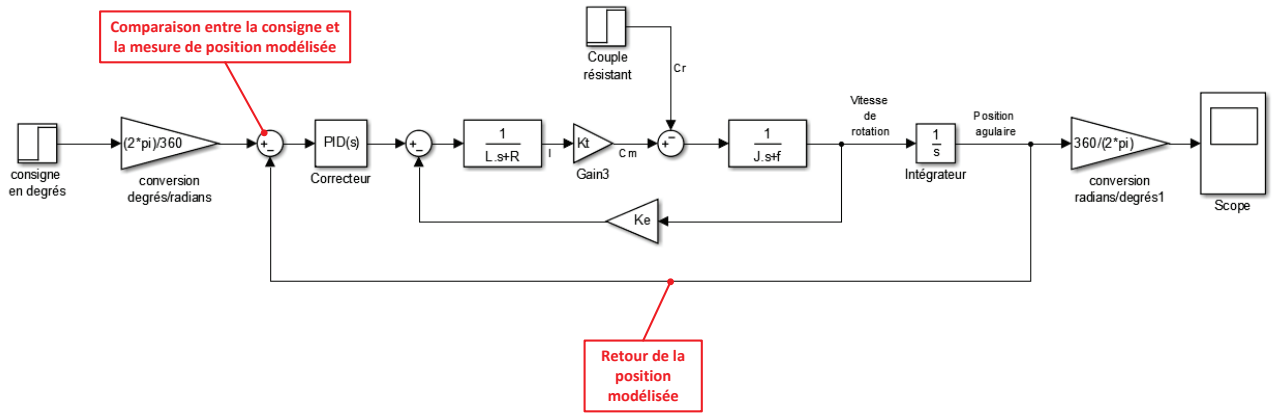


Figure 20 : modélisation de l'asservissement de position du moteur

Dans un premier temps le correcteur n'est pas actif et la réponse de l'asservissement de position est relevée sur la Figure 21.

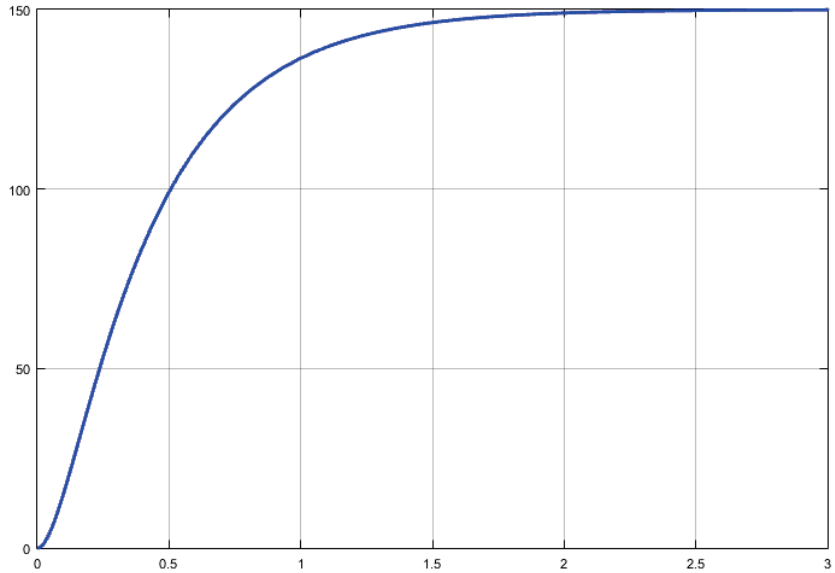


Figure 21 : réponse de l'asservissement de position non corrigé

Le diagramme des exigences impose les critères de performance suivants :

- Temps de réponse à 5% inférieur à 0.8 s
- Temps de montée à 80% inférieur à 0.4 s
- Ecart statique inférieur à 5% de la consigne

Il apparaît que les performances modélisées de l'asservissement ne sont pas conformes aux performances souhaitées. Le temps de réponse à 5% est de l'ordre de 1.3 s et le temps de montée à 80 % de l'ordre de 0.7s.

Il faut donc optimiser la commande du moteur en procédant au réglage du correcteur PID. Le logiciel de simulation propose alors de nombreuses fonctionnalités permettant de mettre en œuvre la démarche. La Figure 22 illustre l'utilisation de la démarche de réglage d'un correcteur dans le domaine fréquentiel à l'aide d'outils spécifiques.

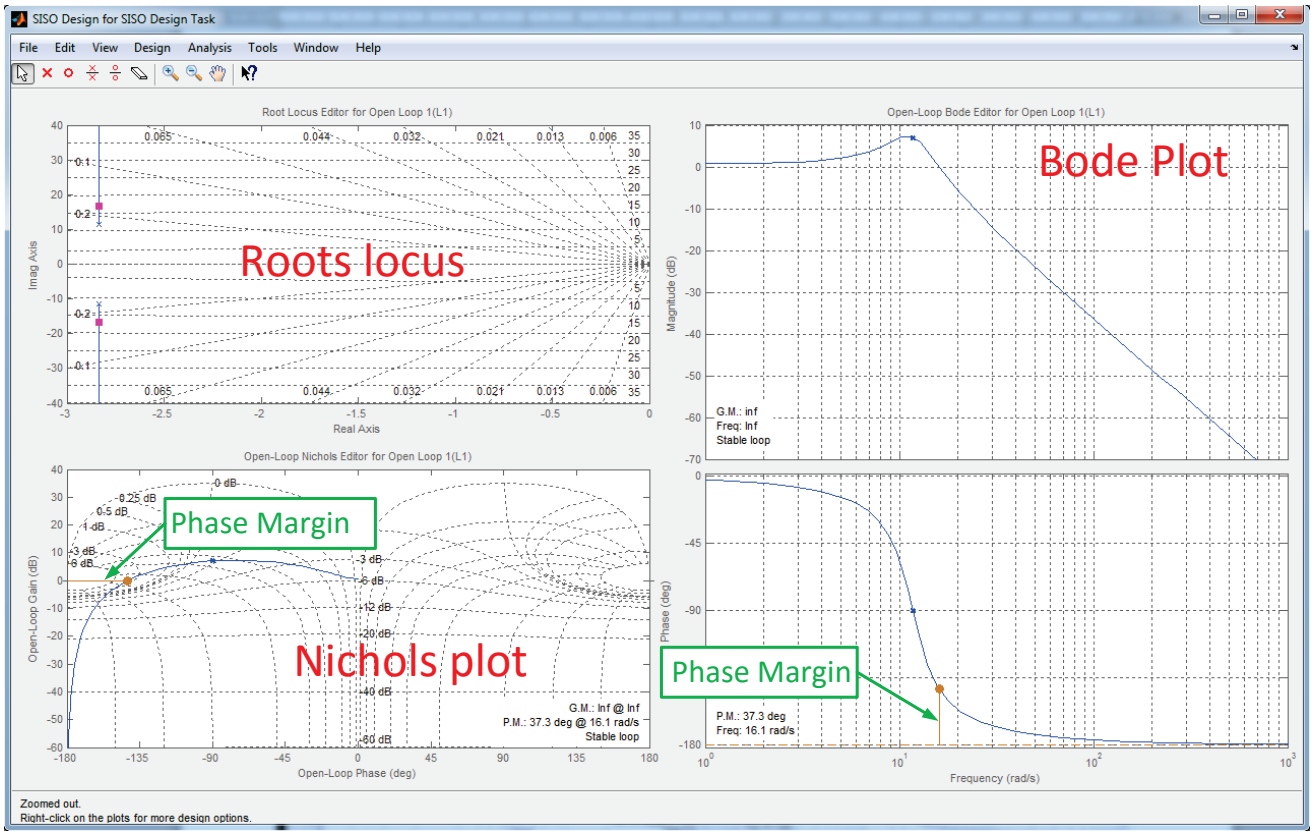


Figure 22 : illustration de l'utilisation d'un outil de contrôle commande pour régler un correcteur

Ces méthodes seront largement développées dans la suite de l'ouvrage et demandent de réaliser de nombreux essais sur le modèle. Le modèle de l'asservissement étant construit sur la base du modèle validé du moteur les résultats obtenus par simulation seront très proches des résultats mesurés sur le système réel.
A l'issu de cette phase de réglage, le correcteur est déterminé pour obtenir la réponse représentée sur la Figure 23.

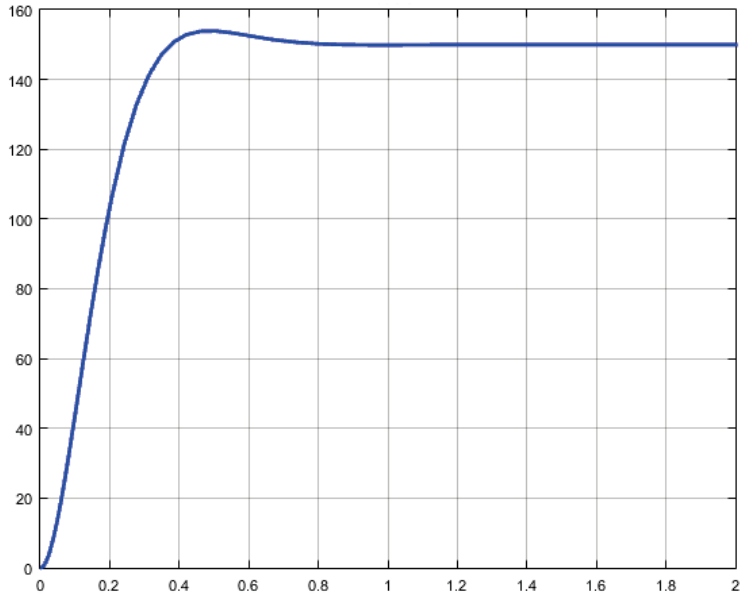


Figure 23 : réponse de l'asservissement de position corrigée

Le temps de réponse à 5% est maintenant de l'ordre de 0.3 s et le temps de montée à 80 % de l'ordre de 0.25s. Les performances modélisées sont conformes aux performances souhaitées et la phase de **Codage Implémentation** va pouvoir commencer.

6. Le Model-in-the-loop (MIL)

Cette étape se situe entièrement dans le domaine de la simulation et correspond à la simulation du modèle complet précédemment décrit. Le modèle est décrit entièrement en code MATLAB, il s'exécute entièrement sur l'ordinateur avec lequel a été construit le modèle.

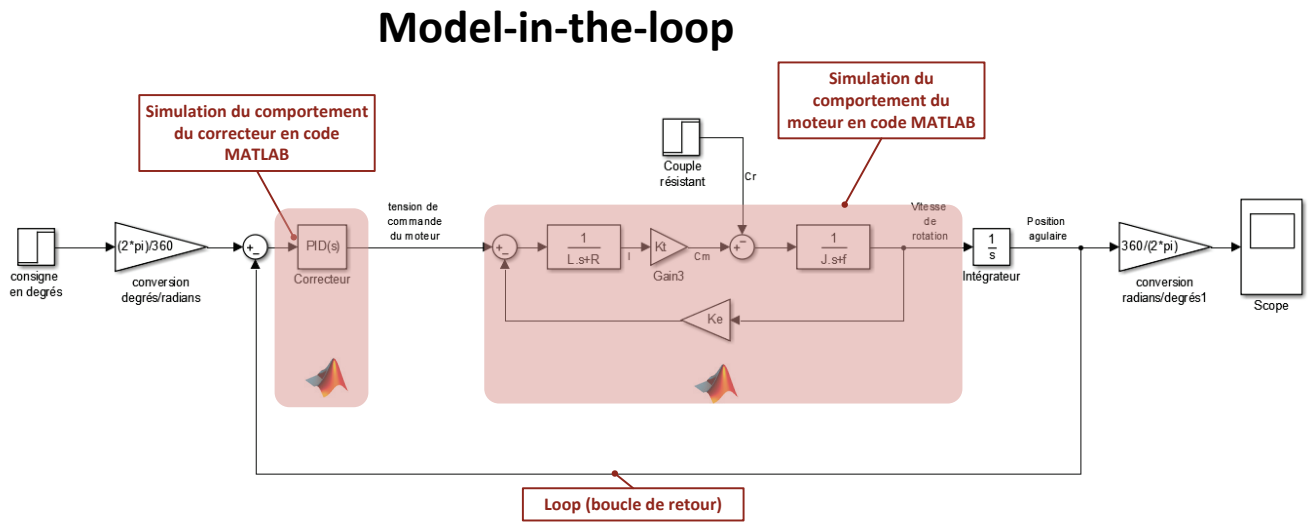


Figure 24 : le Model-in-the-loop (MIL)

D. La phase de Codage Implémentation

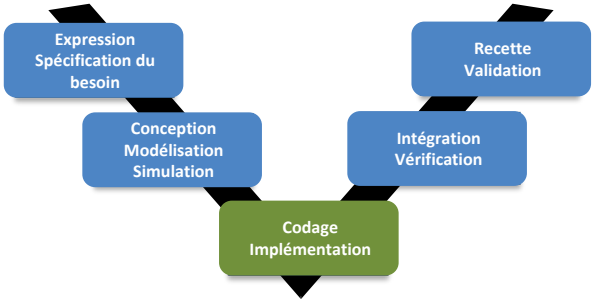


Figure 25 : la phase de Codage Implémentation

Cette phase permet de migrer progressivement du domaine de la simulation vers la phase de test sur le matériel réel.

A ce stade il est important de revenir sur la structure du modèle validé qui vient d'être finalisée à l'issu de la phase de **Conception Modélisation Simulation**. La Figure 26 montre les deux parties du modèle qu'il faudra bien distinguer pour comprendre les étapes qui vont suivre.

- la partie du modèle correspondant au correcteur qui était à la base des objectifs de conception du projet.
- la partie du modèle correspondant au moteur qui permet de disposer d'une modélisation validée des actionneurs.

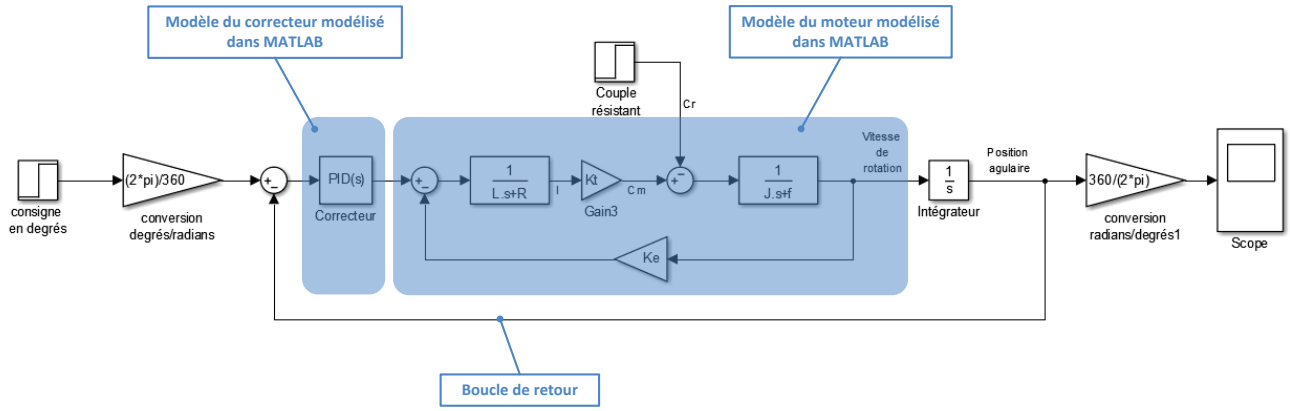


Figure 26 : structure du modèle à l'issu de la phase de Conception Modélisation Simulation

Cette phase permet de traduire des parties du modèle en Code C. Le code C est le langage de programmation utilisé par la carte de commande qui ne peut pas interpréter directement le code MATLAB. Une première méthode consiste alors à isoler la partie du modèle que l'on veut remplacer par du code C et à faire le bilan des entrées et des sorties de cette partie du modèle. Il est possible de réaliser ce codage manuellement. Dans ce cas une équipe de codeurs spécialiste du langage C vont réaliser les programmes qui généreront la même relation entre les entrées et les sorties que celle obtenue avec du code MATLAB. Cette approche est longue et mobilise des compétences de haut niveau en codage et en programmation.

Les logiciels de simulation permettent également de générer du code C automatiquement et MATLAB propose des interfaces permettant d'automatiser ce processus afin de gagner un temps précieux.

Qu'il soit automatique ou manuel, le processus de codage en langage C, peut générer des erreurs et le comportement du modèle codé en langage C peut différer de son comportement codé en MATLAB. Afin de détecter et de corriger ces erreurs avant qu'elles ne puissent impacter la phase de test sur du matériel réel, il faut mettre en place deux configurations de fonctionnement :

- le software-in-the-loop (SIL)
- le processor-in-the-loop (PIL)

1. Le Software-in-the-loop (SIL)

Dans cette configuration, une partie du modèle est remplacé par du code C qui va s'exécuter sur l'ordinateur. Cela permet de vérifier que la génération de code C n'a pas engendré d'erreur en terme de traduction de code C vers le code MATLAB.
Dans notre exemple nous allons remplacer le correcteur codé dans MATLAB par une fonction C qui va générer du code C en remplacement du code MATLAB et exécuter ce code C sur l'ordinateur (Figure 27).

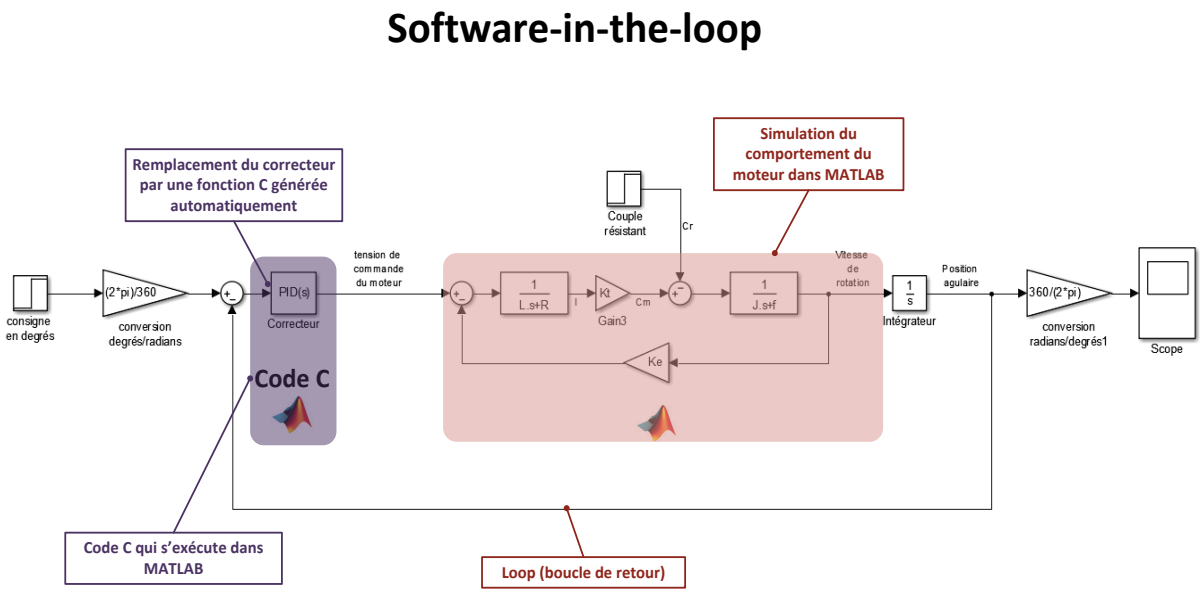


Figure 27 : le Software-in-the-loop(SIL)

Il faut alors vérifier que le comportement du modèle est le même, qu'il soit piloté avec le correcteur codé en code MATLAB ou avec le correcteur codé en code C. Une fois cette vérification effectuée, les concepteurs peuvent exclurent toute erreur due à la génération de code C.

2. Le Processor-in-the-loop (PIL)

Cette configuration permet d'éviter les erreurs d'exécution du code C. En effet les conditions d'exécution du code C dépendent largement des capacités et des performances de la cible matérielle sur laquelle il s'exécute. En mode software-in-the-loop, le code s'exécute sur le processeur de l'ordinateur, en mode processor-in-the-loop, le code s'exécute sur la cible matérielle qui embarquera le code dans les conditions réelles de fonctionnement. Ces deux processeurs peuvent gérer différemment l'exécution d'un même programme. Le correcteur modélisé est alors remplacé par du code C qui s'exécutera directement sur la carte de commande. Dans cette phase, le processeur de la carte pilote le modèle du moteur.

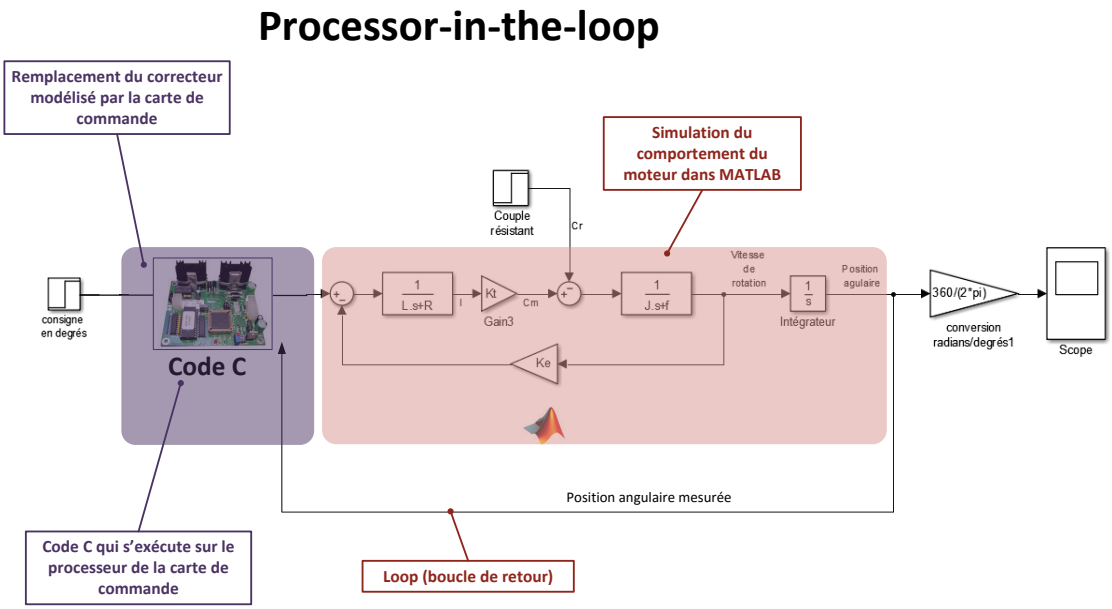


Figure 28 : le Processor-in-the-loop

A l'issue de ces deux phases, les erreurs de génération et d'exécution de code C ont été corrigées et la phase de test sur les actionneurs réels va pouvoir commencer.

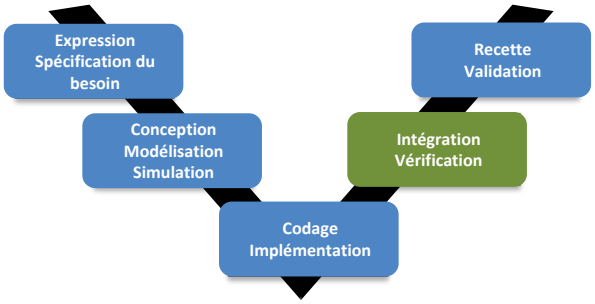


Figure 29 : la phase d'Intégration Vérification

1. Le Hardware-In-the-Loop (HIL)

Cette étape consiste à remplacer le modèle du moteur par le moteur réel. Le correcteur et les calculs associés peuvent être effectués soit par l'ordinateur (mode externe), soit par la carte de commande qui fonctionne de manière autonome (mode embarqué).

HIL - mode externe (Figure 30 et Figure 31)

Dans cette phase tous les calculs sont réalisés sur l'ordinateur et la carte de commande fonctionne uniquement en interface de puissance. Les fonctions de calcul du processeur de la carte de commande ne sont pas sollicitées.

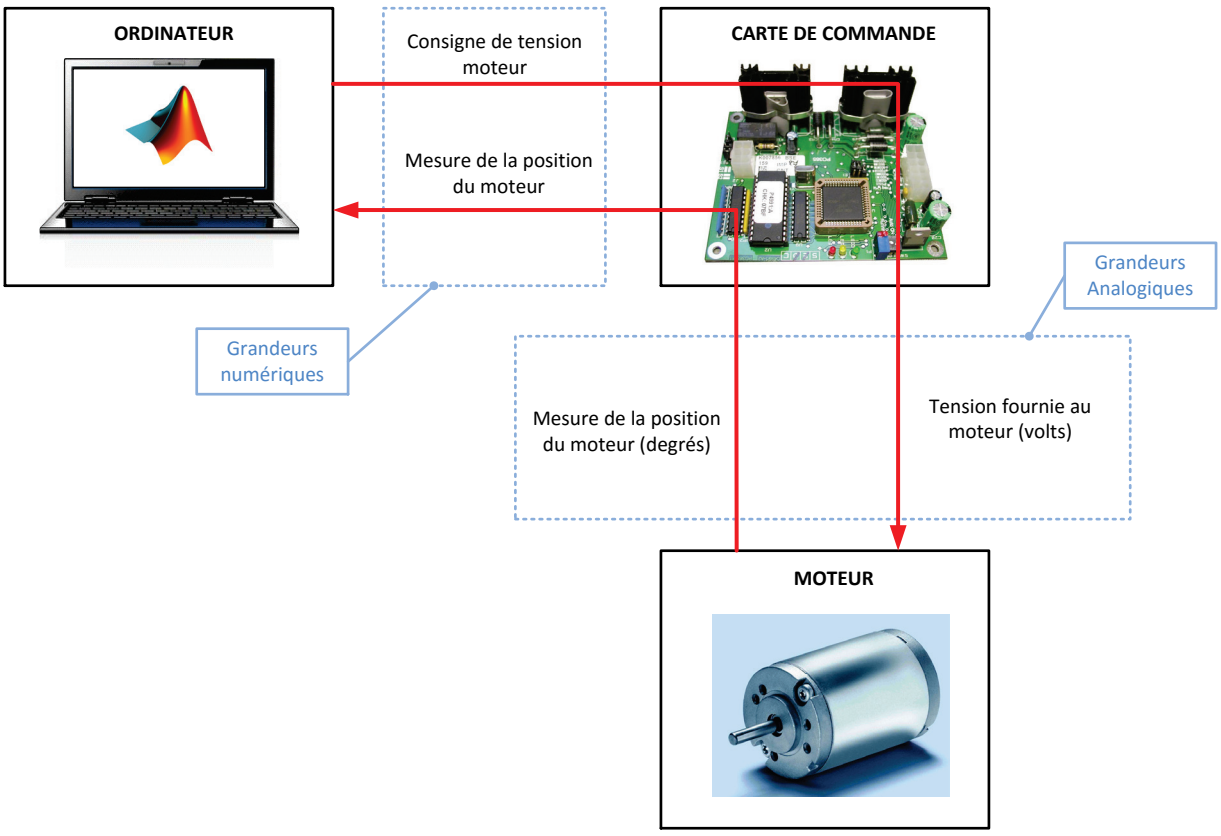


Figure 30 : architecture matérielle du hardware-in-the-loop en mode externe

Hardware-in-the-loop
Mode externe

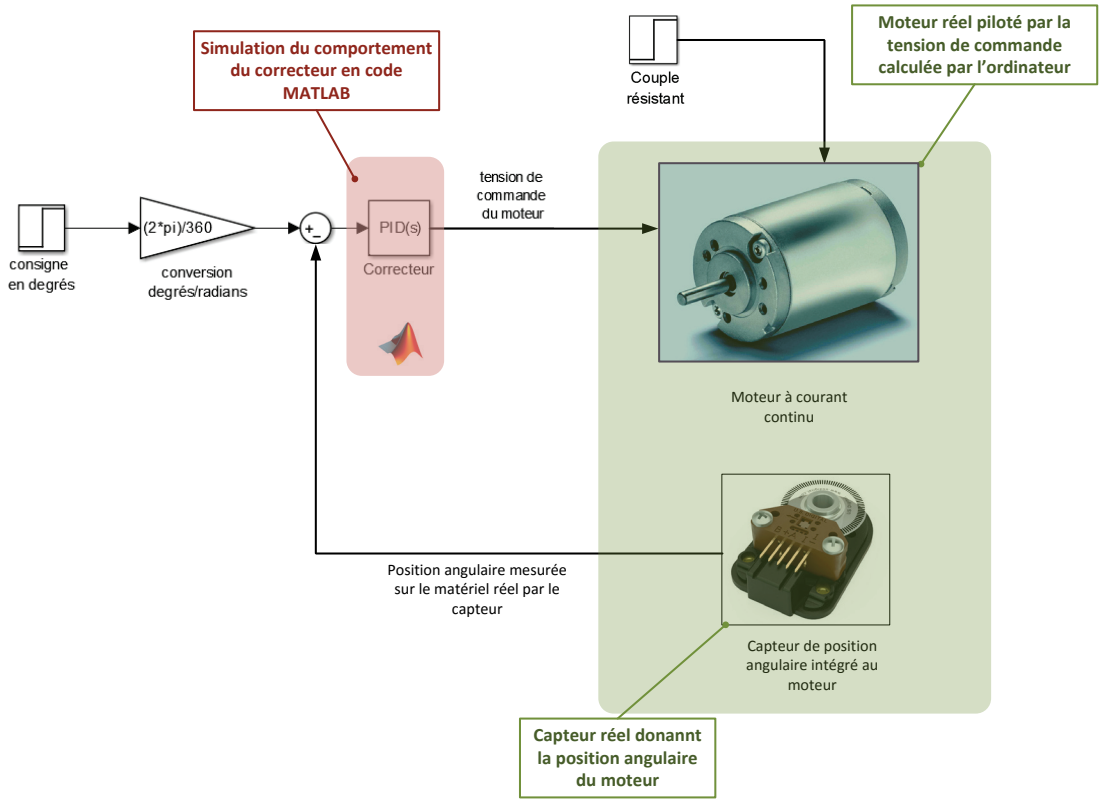


Figure 31 : le hardware-in-the-loop en mode externe: intégration du matériel dans la boucle

HIL - mode embarqué (Figure 32 et Figure 33)

Dans cette phase tous les calculs sont réalisés par la carte de commande. La logique de commande s'exécute sur le processeur de la carte de commande.

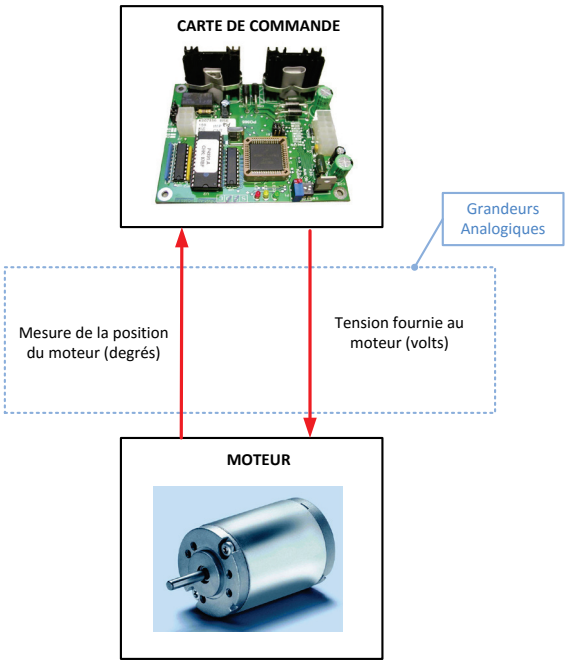


Figure 32 : architecture matérielle du Hardware-In-the-Loop en mode embarqué

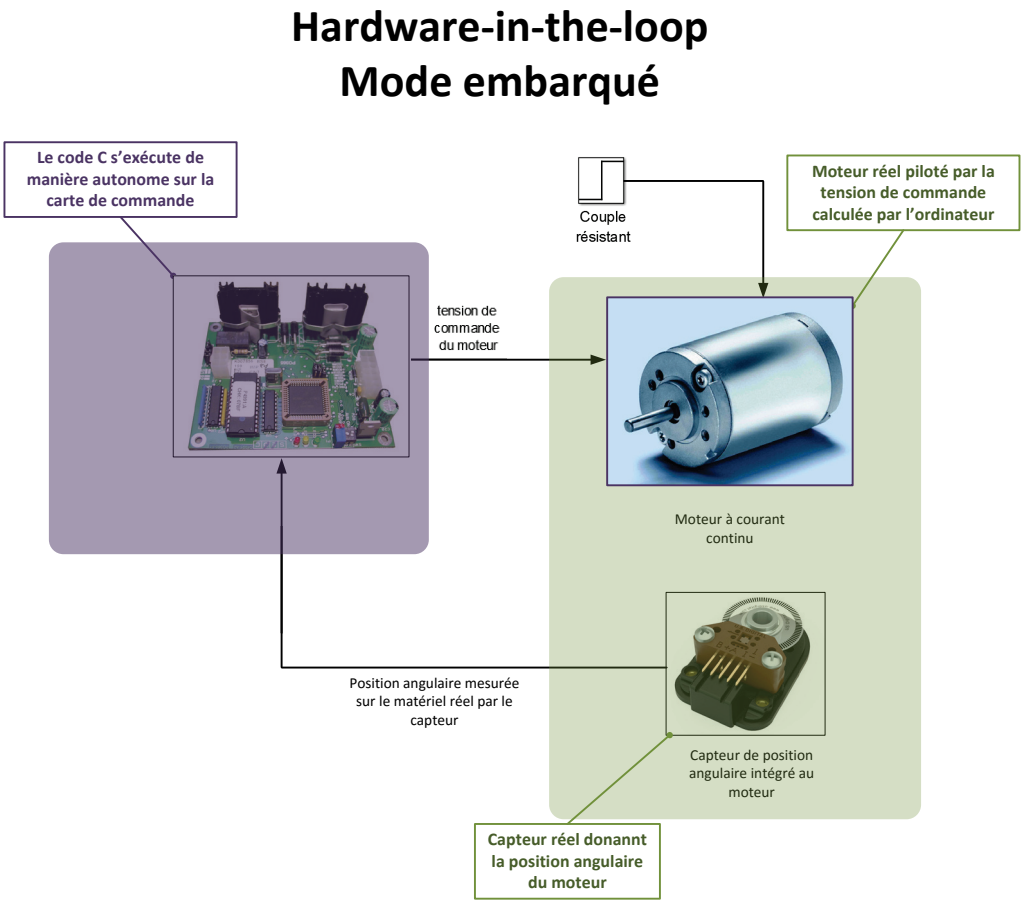


Figure 33 : le Hardware-In-the-Loop mode embarqué: fonctionnement autonome du matériel

L'une ou l'autre de ces configurations permettent de vérifier que le correcteur mis au point permet d'atteindre les performances spécifiée sur le matériel réel. Il est alors possible de relever la vitesse effective du moteur sur la Figure 34. Il est alors possible de passer à la phase de **Validation**.

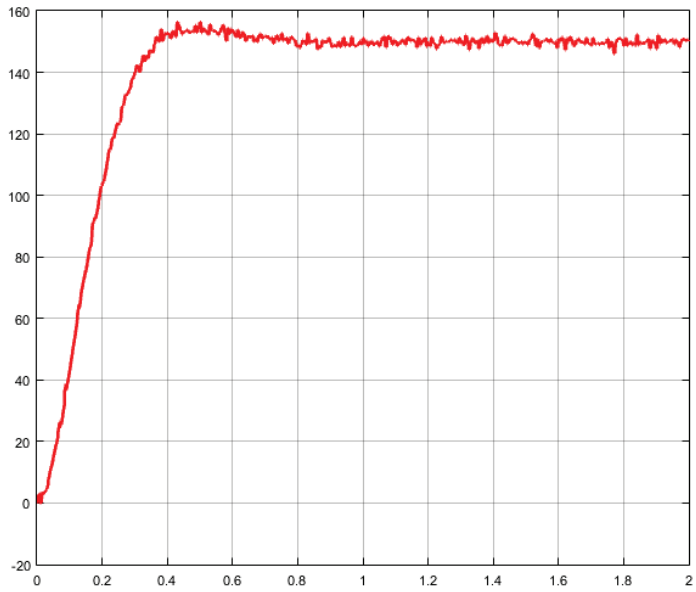


Figure 34 : visualisation de la vitesse de sortie du moteur à l'issu de la phase Hardware-in-the-loop

F. La phase de Validation Recette

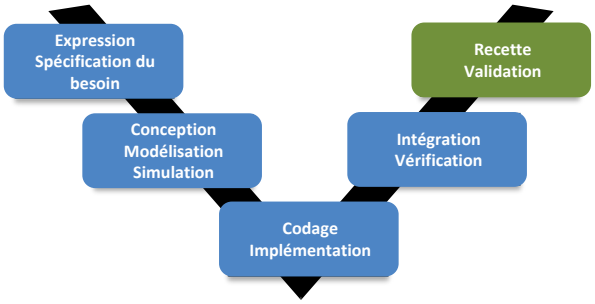


Figure 35 : la phase de Validation Recette

Dans cette phase, il reste à vérifier que les performances mesurées correspondent aux performances spécifiées dans le cahier des charges. L'analyse de la Figure 34 permet de voir que le temps de réponse à 5% est de l'ordre de 0.3 s et le temps de montée à 80 % de l'ordre de 0.25 s. Les performances réelles sont conformes aux performances modélisées et aux performances souhaitées.
La phase de test a été réduite au strict minimum et les spécifications imposées par le cahier des charges sont respectées.

Chapitre 2 : Introduction et présentation des outils de modélisation

I. Le logiciel MATLAB-Simulink

Le logiciel **MATLAB-Simulink** propose de nombreux outils permettant de mettre en œuvre une démarche de modélisation globale multi-physique. La première étape est de sélectionner une palette d'outils adaptés permettant de modéliser les systèmes dans le cadre de la formation des ingénieurs.
Il est préférable, avant d'aborder la phase de prise en main de ces outils, d'avoir une vision globale de leurs fonctionnalités et de leur approche causale ou acausale. La pertinence de la démarche de modélisation réside avant tout dans le choix de l'outil adapté pour modéliser les différentes parties du système. La maitrise structurelle de ces outils et de leur potentiel garantira la réussite du processus de modélisation.
Dans un premier temps, nous allons décrire les fonctionnalités de ces outils. Ensuite nous ferons le lien entre ces outils et le descripteur « Chaîne d'énergie /Chaîne d'information » afin de mettre en place une stratégie de modélisation globale des systèmes.

A. Description et hiérarchie des outils utilisés

1. MATLAB

MATLAB est un puissant outil de calcul permettant de saisir des instructions sous la forme de lignes de commandes. **MATLAB** possède des bibliothèques de fonctions regroupées dans des « Toolbox » relatives à un domaine de connaissance. **Control System Toolbox** pour le contrôle commande des sytèmes, **Signal Process Toolbox** pour le traitement du signal…
La Figure 36 présente un exemple de programme écrit en MATLAB (script) et la Figure 37 présente les résultats obtenus.

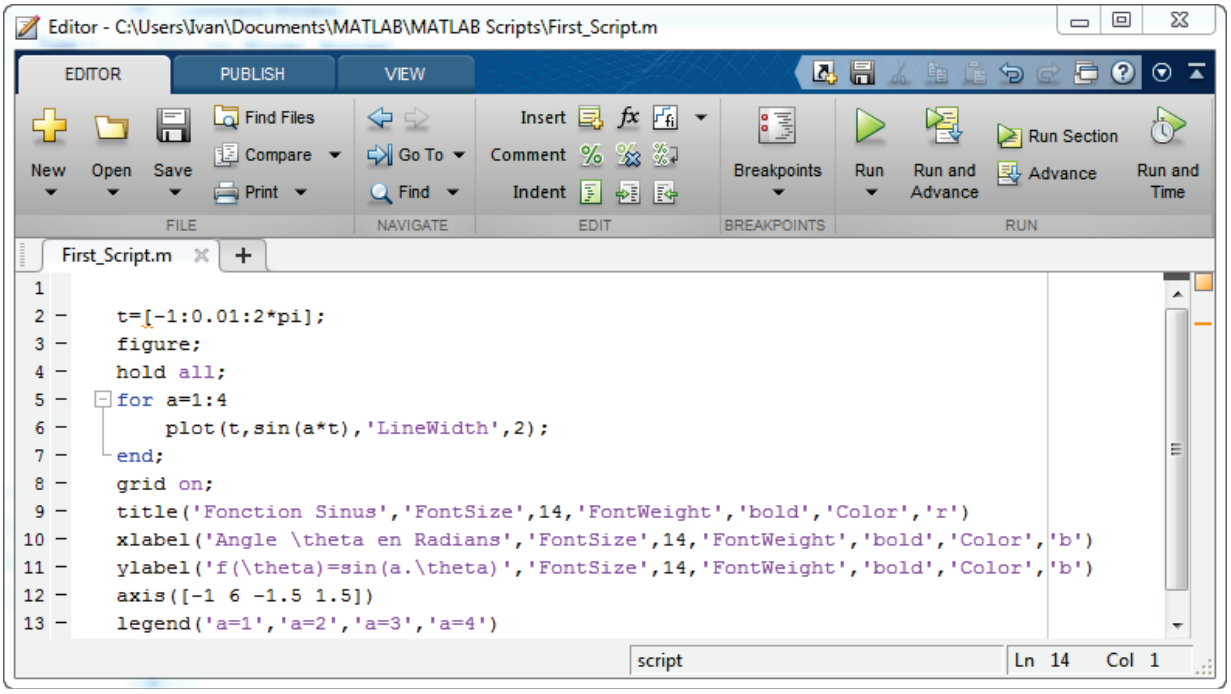


Figure 36 : exemple de script écrit en langage MATLAB

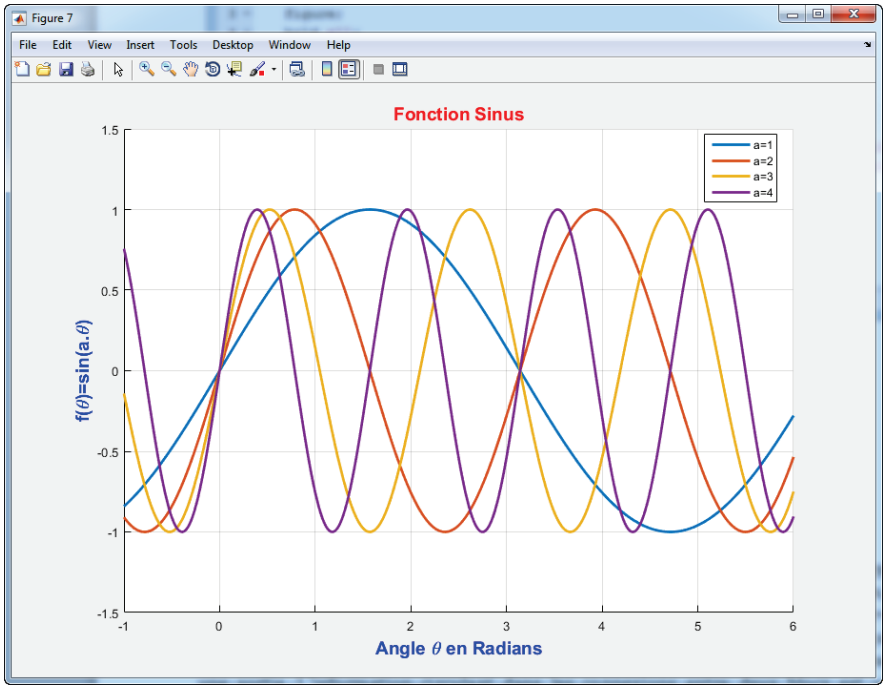


Figure 37 : le résultat obtenu après exécution du script

2. Simulink

Simulink est l'interface graphique de **MATLAB** qui permet de s'affranchir du code et de la syntaxe indispensable à la saisie des lignes de commandes. **Simulink** possède des bibliothèques de blocs, regroupés dans des Blocksets (Simulink Control Design pour le contrôle commande des systèmes par exemple). **Simulink** propose une approche causale de la modélisation. Le comportement dynamique d'un système est caractérisé par un bloc contenant, par exemple, la fonction de transfert du système avec une entrée et une sortie. L'information circulant dans les connexions entre deux blocs est un signal numérique orienté. La sortie du bloc est calculée numériquement en déterminant pour chaque pas de calcul la transformation du signal d'entrée imposé par le contenu du bloc. Simulink propose également toute une palette d'outils très évolués pour le contrôle commande des systèmes asservis.

Cette approche largement utilisée, nécessite une parfaite connaissance des lois physiques qui caractérisent le comportement des systèmes. Toute phase de modélisation commence par l'écriture des équations différentielles caractéristiques du phénomène physique étudié et l'obtention des fonctions de transfert de tous les sous-systèmes composant le système étudié.

La Figure 38 présente un **modèle Simulink** et la Figure 39 montre les résultats obtenus:

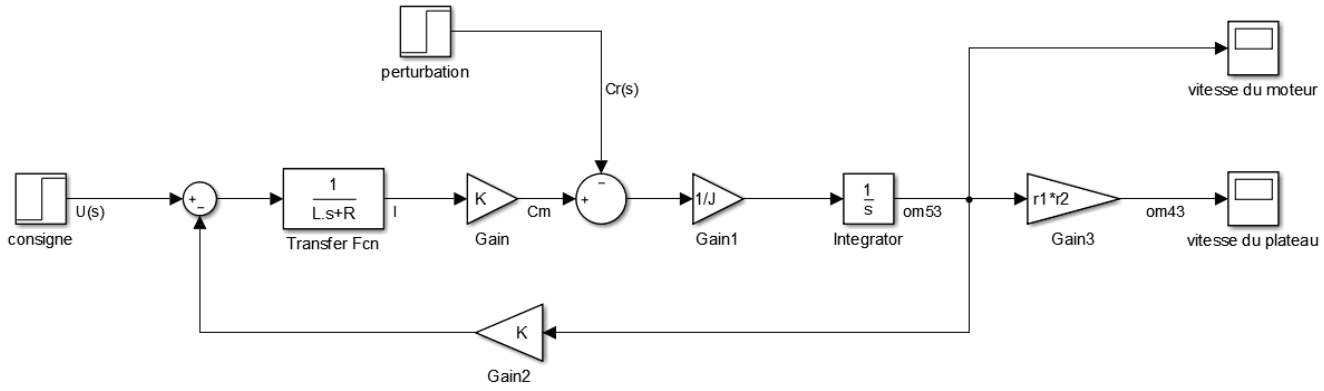


Figure 38 : exemple de modélisation Simulink

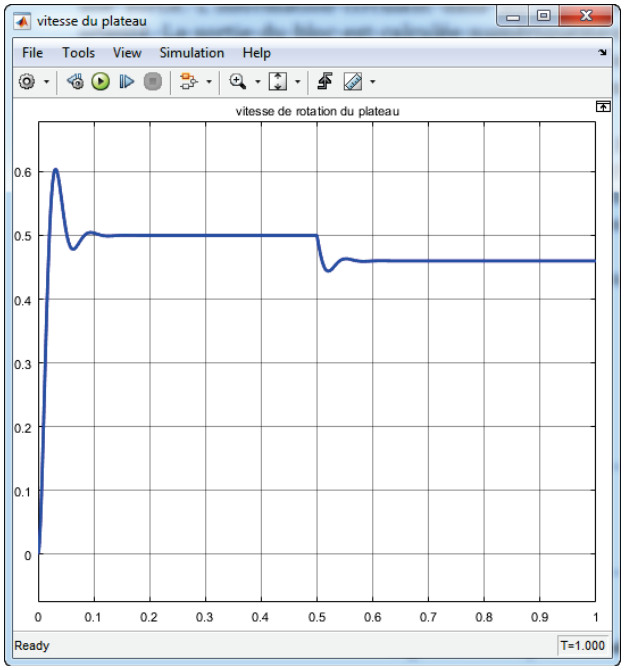


Figure 39 : la visualisation dans un scope des résultats obtenus

3. Simscape

Simscape propose une approche acausale de la modélisation et permet de faire de la modélisation par assemblage de composants. Le comportement physique des composants est pris en compte directement par le logiciel, il est donc possible de modéliser un système sans avoir à écrire l'équation différentielle qui caractérise son comportement.
Chaque domaine physique est représenté par une couleur. Les connexions entre deux composants d'un même domaine ne sont pas orientées, ont une signification physique et transmettent un niveau d'information supérieur aux connexions en modélisation causale. Ces connexions peuvent être un fil électrique (transfert d'information de type courant et tension), un arbre moteur (transfert d'information de type couple et vitesse angulaire), l'extrémité de la tige d'un vérin (transfert d'information de type force et vitesse linéaire)… Le principe de calcul s'appuie sur un bilan de puissance à chaque nœud du modèle et ne repose pas sur le principe de causalité, d'où le nom de modélisation acausale.

La Figure 40 présente un exemple de modèle réalisé avec **Simscape** et la Figure 41 montre les résultats obtenus.

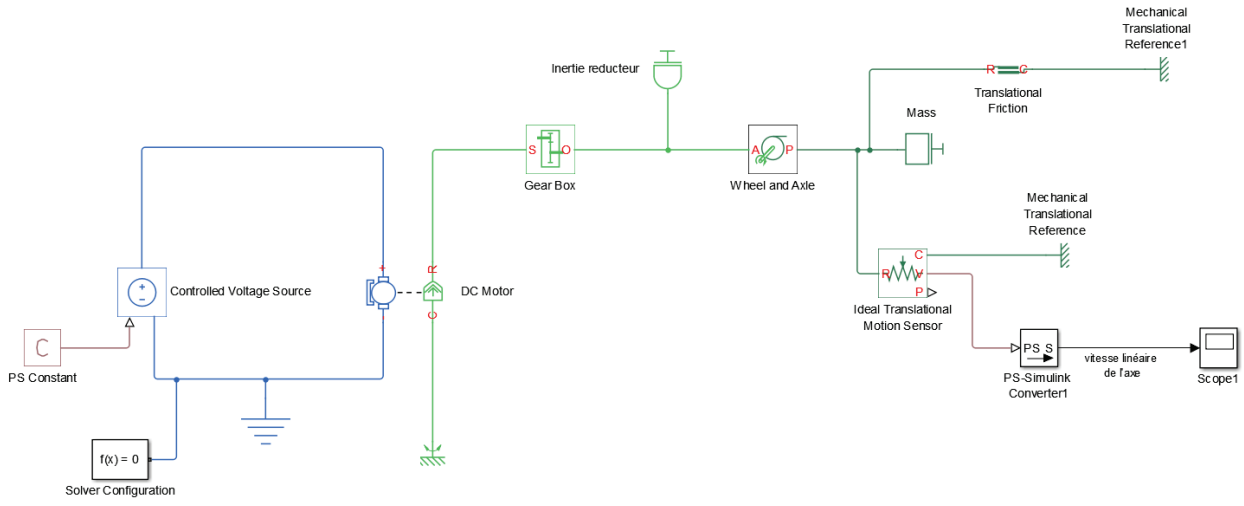


Figure 40 : exemple de modélisation Simscape

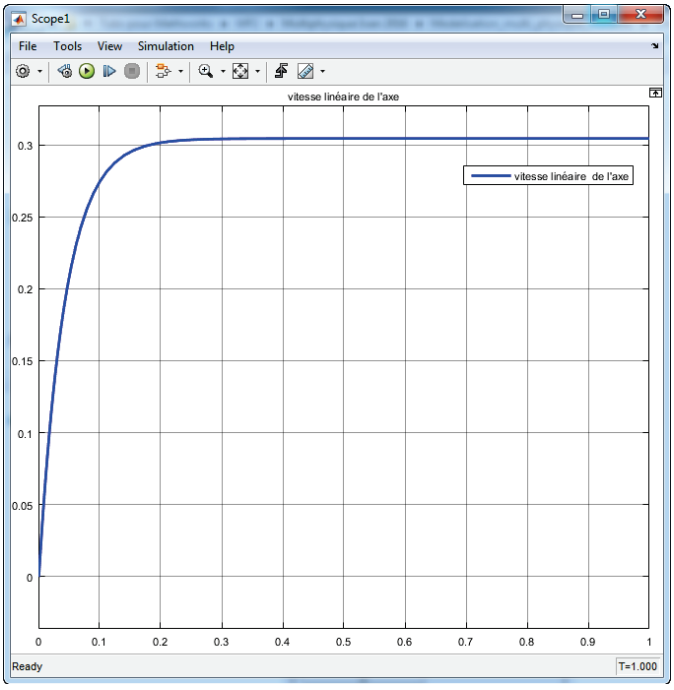


Figure 41 : la visualisation dans un scope des résultats obtenus

Simscape inclue une bibliothèque de composants élémentaires dans toutes les technologies (mécanique, électrique, hydraulique, pneumatique, thermique, magnétique…) et comprend également différents modules :

- **SimMechanics :** ce module permet d'importer des modèles CAO 3D et de les intégrer directement dans une modélisation multi-physique, un asservissement... Les propriétés cinétiques des pièces sont prises en compte et le comportement dynamique automatiquement intégré.
- **SimPowerSystems** : composants électriques (toutes les technologies de moteurs, alimentation, filtres, préactionneurs électriques…)
- **SimElectronics** : composants pour l'électronique et la mécatronique (servo-moteurs, commande PWM, pont en H, convertisseurs, portes logiques…)
- **SimHydraulics** : composants permettant de réaliser des circuits hydrauliques (toutes les technologies de pompes, distributeurs, vérins, clapets, soupapes…)
- **SimDrivelines** : composants de transmission de puissance (réducteurs, freins, embrayages, différentiels, accouplements…)

Chacun de ces modules explore un champ technologique spécifique et propose une modélisation de niveau supérieur aux bibliothèques de base de **Simscape**.

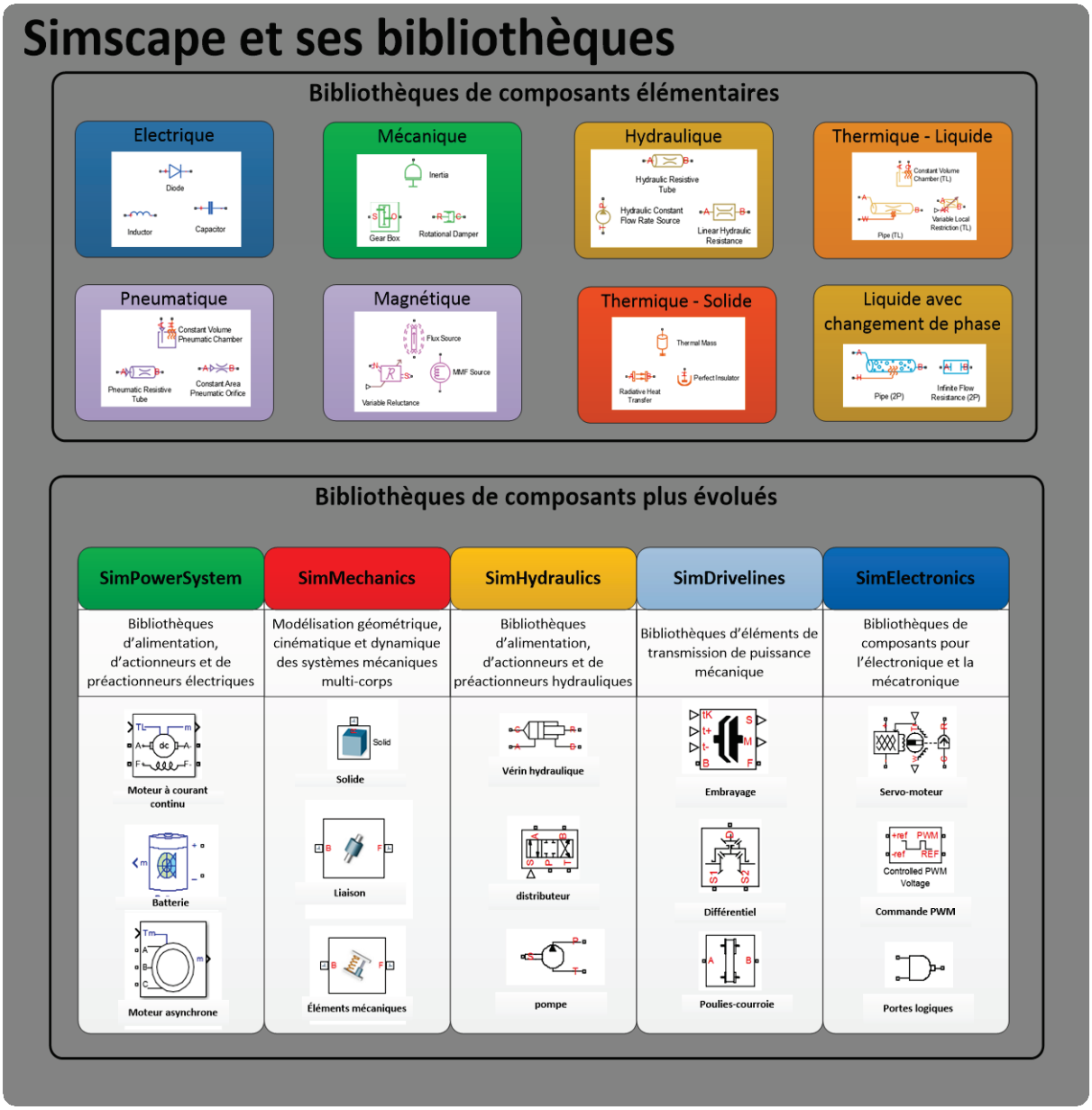


Figure 42 : Simscape et ses bibliothèques

4. Stateflow

Stateflow prend en charge la modélisation du comportement combinatoire et séquentiel des systèmes à partir d'une description sous la forme de diagrammes d'états, de flux logiques ou de tables de vérité. La Figure 43 présente un exemple de modélisation de machine à états avec **Stateflow**.

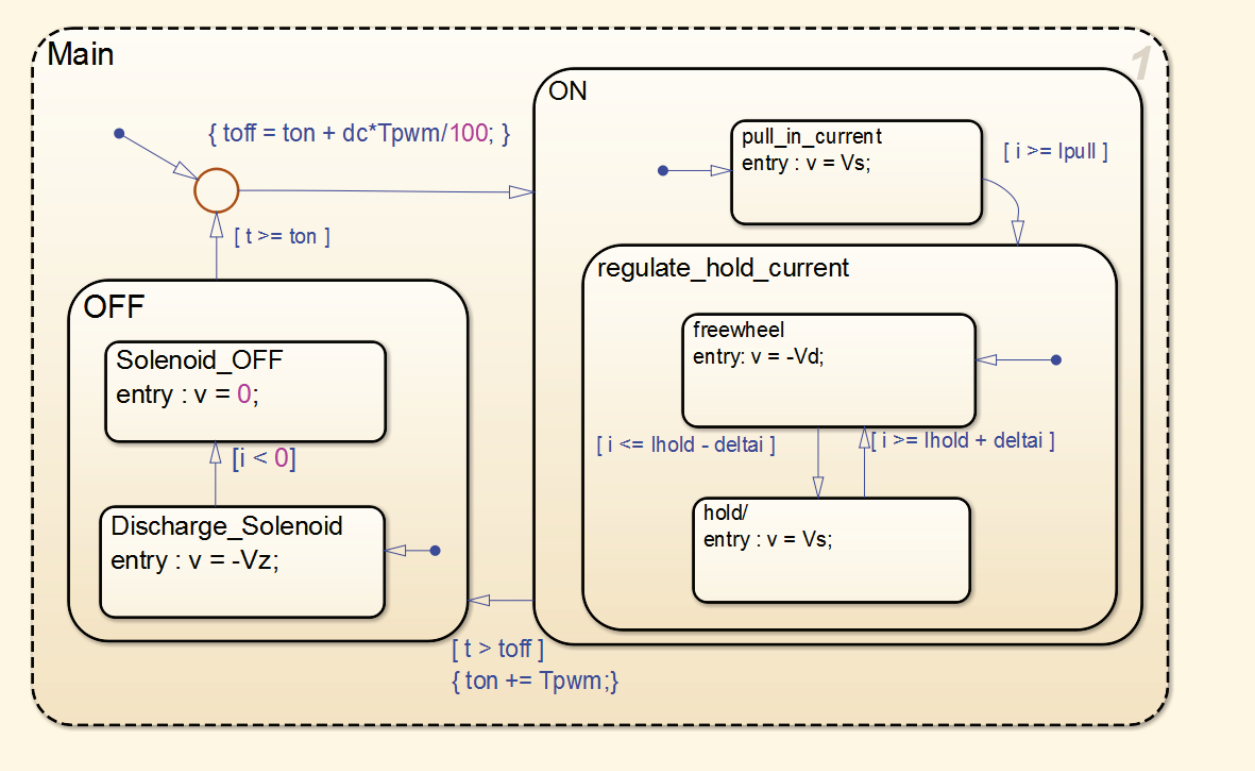


Figure 43 : modélisation du comportement d'une boîte de vitesse automatique avec Stateflow

5. Utilisation des outils de modélisation

Tous ces outils de modélisation communiquent entre eux et permettent de mener une démarche de modélisation globale d'un système. Il est possible dans un même modèle de combiner modélisation causale et acausale, d'intégrer une maquette CAO 3D avec **SimMechanics**, de prendre en compte un comportement séquentiel avec **Stateflow** tout en utilisant les outils de contrôle commande de **Simulink**. Il est possible de faire communiquer le modèle avec **MATLAB** et d'utiliser des fonctions d'exportation de résultats, de lancer une simulation à partir d'un programme **MATLAB** en faisant varier automatiquement les paramètres souhaités…

La Figure 42 et la Figure 44 décrivent l'organisation hiérarchique de la palette d'outils sélectionnée dans l'environnement **MATLAB-Simulink.**

Dans un premier temps, nous allons présenter l'environnement du logiciel afin de se familiariser avec les procédures de lancement des différents outils. Ensuite nous aborderons la prise en main des outils sur la base d'exemples qui peuvent être abordés avec des étudiants dans le cadre de la formation des ingénieurs. La performance dans la démarche de modélisation repose avant tout sur la connaissance des fondements de chaque outil. Il sera ensuite aisé pour l'utilisateur d'approfondir l'exploitation de ces outils en fonction des besoins de la démarche de modélisation menée.

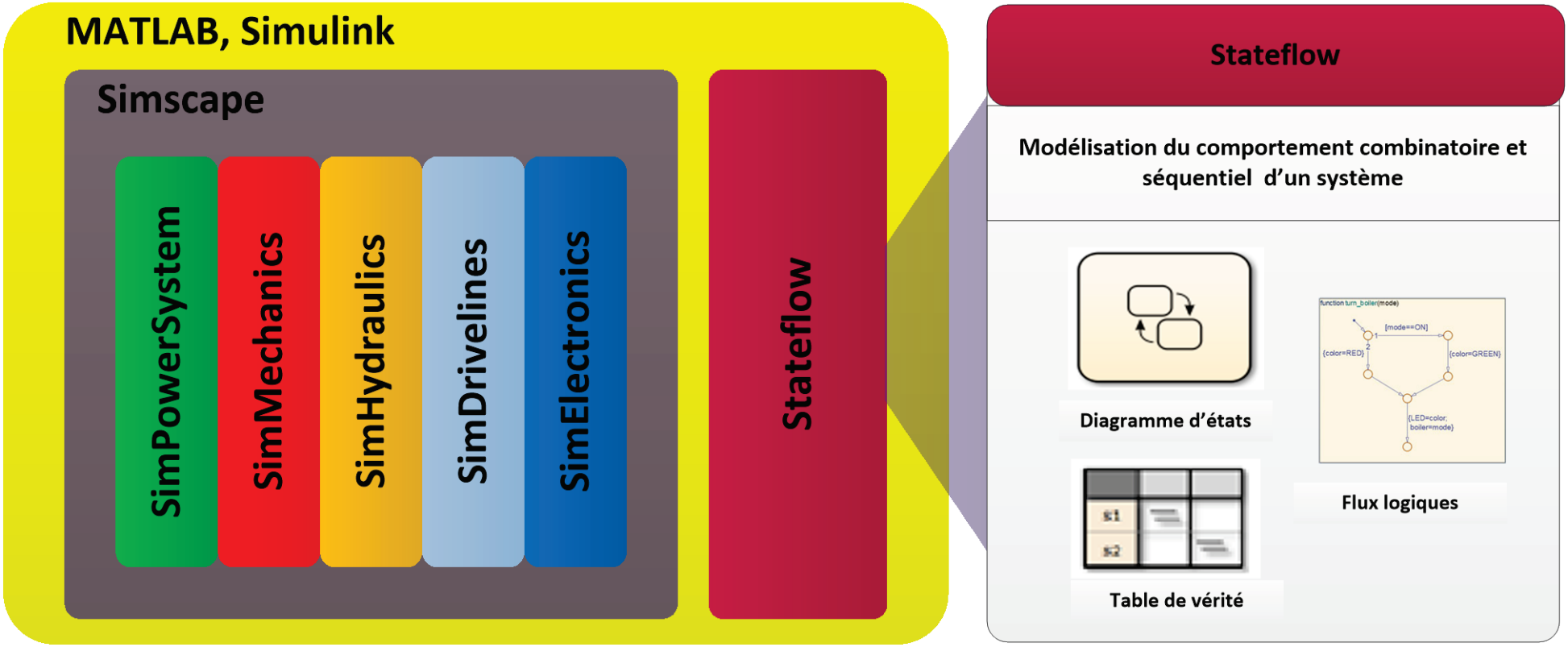


Figure 44 : les outils MATLAB pour la modélisation multi-physique

II. Présentation de l'environnement MATLAB – Simulink

A. Lancement du logiciel

A partir de l'icône 

situé sur le bureau, lancer le logiciel MATLAB.

B. La fenêtre de l'environnement MATLAB

A l'ouverture de MATLAB, plusieurs fenêtres sont accessibles.

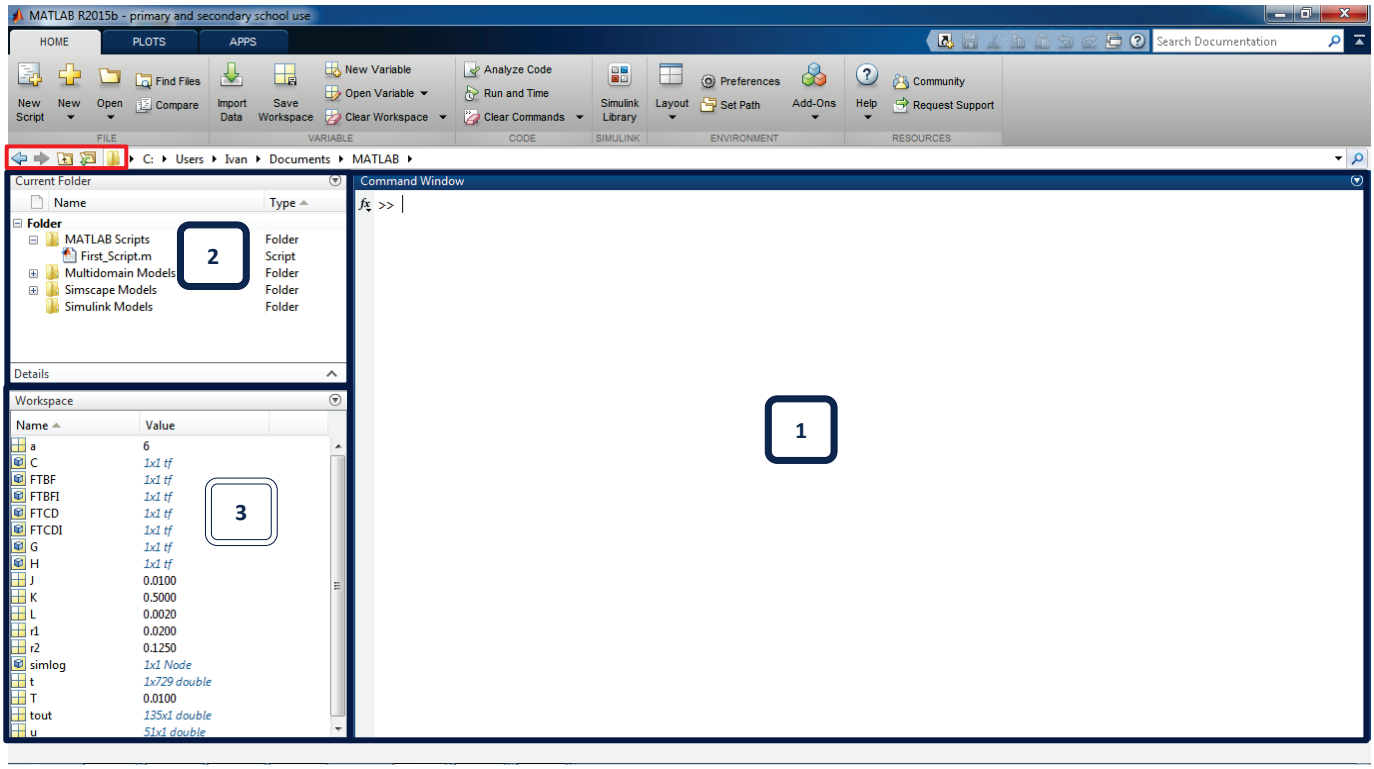


Figure 45 : la fenêtre de l'environnement MATLAB

1 :Command Windows

Fenêtre de commande MATLAB. C'est la fenêtre avec laquelle, l'utilisateur dialogue avec MATLAB (saisie de ligne de commande, d'instructions…)

2 :Current Folder

Cette fenêtre indique le dossier courant dans lequel travaille le logiciel

La barre d'outils  située dans la partie supérieure de cette fenêtre permet de naviguer

dans les dossiers.

3 :Workspace

Cette fenêtre indique le nom et le format de toutes les variables créées et utilisées lors de la session courante.
Il est possible de sauvegarder le Workspace dans un fichier.

1. La barre de commande MATLAB

La Figure 46 présente les principales commandes de la barre de commandes MATLAB.

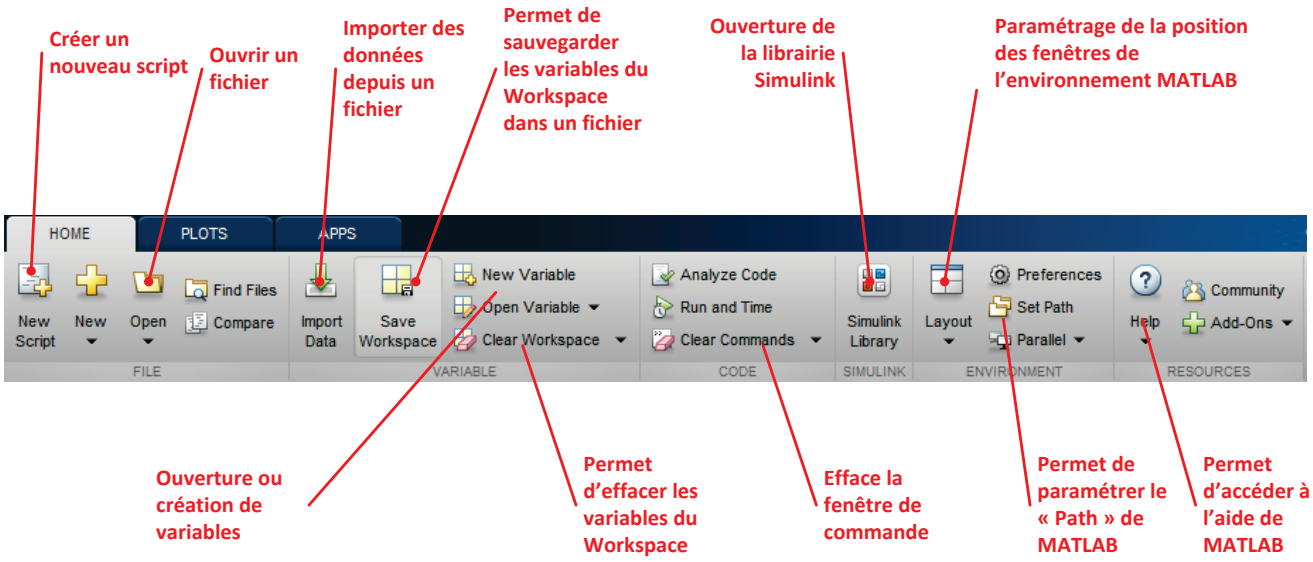


Figure 46 : la barre de commande de l'environnement MATLAB

C. La fenêtre de l'environnement Simulink.

Pour lancer Simulink cliquer sur l'icône 

afin d'ouvrir la librairie des composants Simulink.

La librairie de Simulink s'ouvre (Figure 47) et propose toute une palette de composants pour la modélisation causale ou acausale.

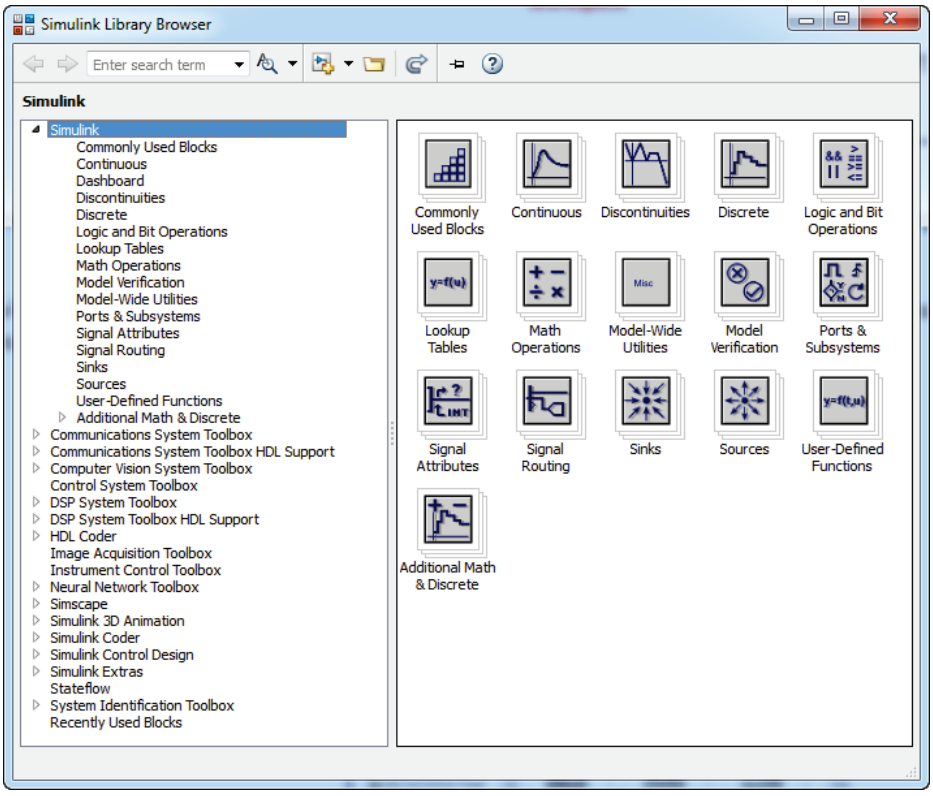


Figure 47 : la bibliothèque de Simulink

Pour créer un nouveau fichier Simulink, cliquer sur l'icône 

La fenêtre de l'environnement **Simulink** s'ouvre. La Figure 48 présente les principales commandes accessibles depuis cette fenêtre.

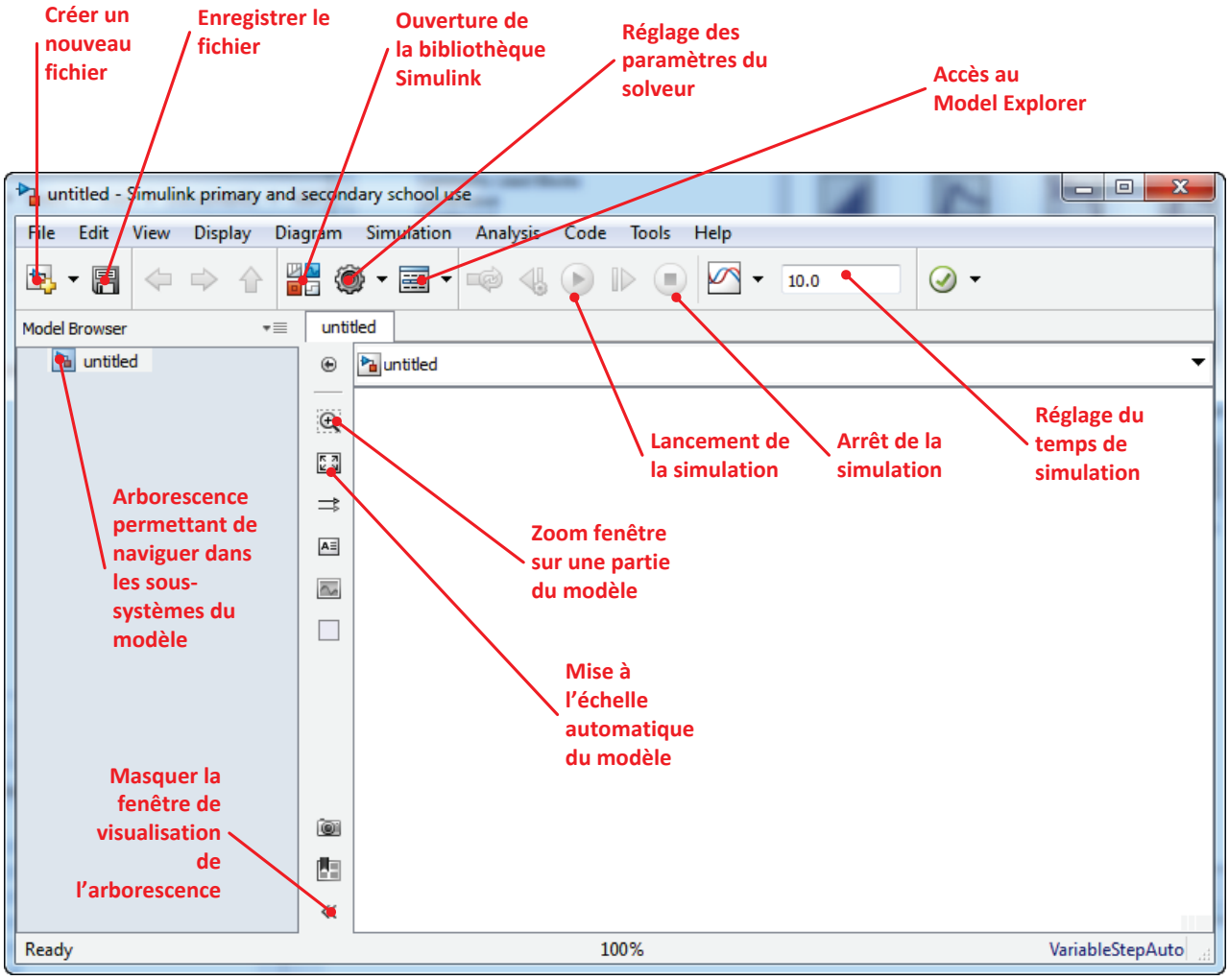


Figure 48 : la fenêtre de l'environnement Simulink

Le placement des composants se fera par une simple action de glisser-déposer des composants depuis la bibliothèque **Simulink** vers l'intérieur de cette fenêtre.

D. Configuration de MATLAB – Simulink

1. Nommer un fichier dans MATLAB/Simulink

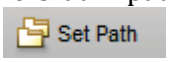
Vous pouvez donner le nom de votre choix à un fichier dans MATALB/Simulink en respectant cependant ces règles :

- Ne pas utiliser d'espace
- Ne pas utiliser de caractères accentués

2. Le « path » de MATLAB

Le « path » de **MATLAB** est constitué de l'ensemble des dossiers que **MATLAB** peut lire lors d'une session courante. Tout fichier appartenant à l'un des dossiers du « path » de **MATLAB** peut être accessible sans en indiquer le chemin. Il convient donc d'inclure dans le « path » tous les dossiers qui contiennent nos fichiers de travail. Cette stratégie d'accès aux fichiers permet de ne pas avoir à spécifier le chemin d'accès aux fonctions et aux fichiers. Il suffit que ces derniers se trouvent dans le « path » spécifié par l'utilisateur pour qu'ils soient automatiquement retrouvés. L'utilisateur tapera le nom d'un script ou fera appel à une fonction dans la fenêtre de commande pour que l'exécution soit immédiate. Cela implique par exemple que deux scripts ne doivent pas porter le même nom.
Il donc important de maitriser la structure du « path » de **MATLAB** et de n'inclure que les dossiers utiles à la session de travail. Il existe deux procédures pour ajouter des dossiers au « Path ». L'une ajoutera les dossiers sélectionnés pour la session de travail courante et pour toutes les sessions futures. L'autre ajoutera les dossiers sélectionnés uniquement pour la session de travail en cours. Il est important de bien comprendre cette différence et de n'inclure définitivement dans le « path » que les dossiers vraiment utiles pour toutes les sessions afin d'éviter les conflits éventuels entre des fichiers du même nom.

3. Ajout de dossiers dans le « path » pour toutes les sessions

Pour ajouter des dossiers au « path » de **MATLAB** pour la session courante et pour toutes les sessions futures, **cliquer** sur 

dans la barre de commande de **MATLAB** pour ouvrir la fenêtre de définition du « path ».

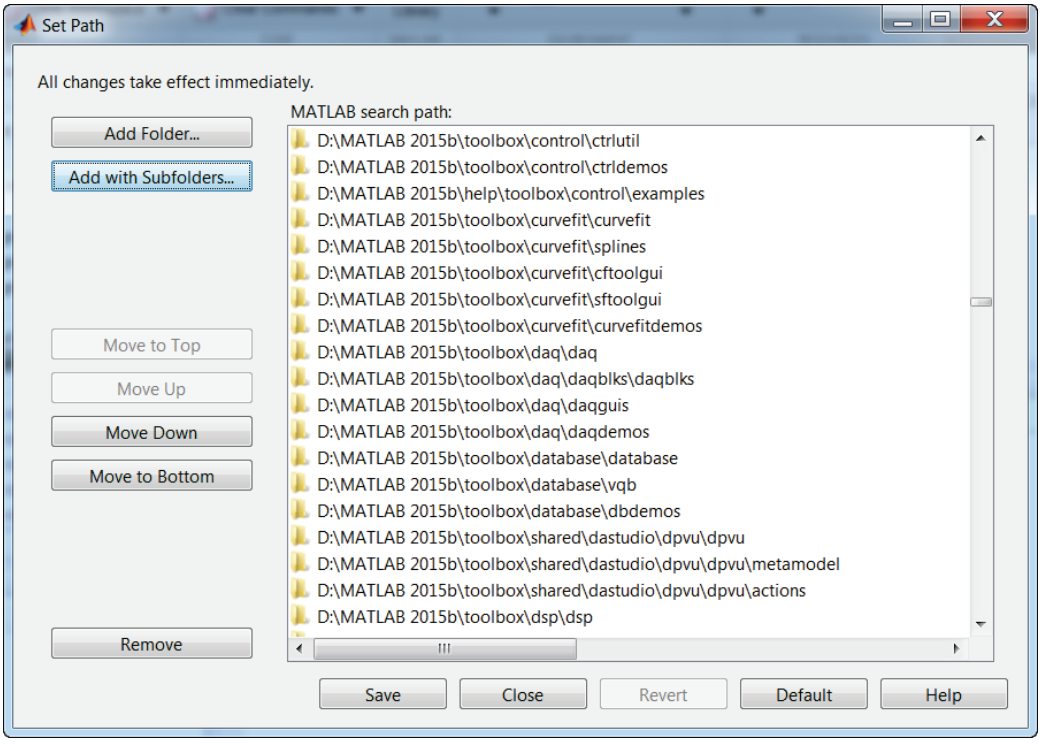


Figure 49 : le path de MATLAB

Sélectionner **Add with Subfolders**, puis choisir les dossiers que vous voulez ajouter dans le « path » du logiciel (les sous-dossiers seront automatiquement ajoutés)

Sélectionner **Save**, puis **Close**.
A la suite de cette opération, le « Path » est sauvegardé et sera le même lors de l'ouverture des prochaines sessions.

4. Ajout de dossiers dans le « path » pour la session courante

Il est également possible de modifer le « Path » pour la session en cours uniquement. Pour cela, cliquer avec le bouton droit de la souris sur le dossier que vous voulez ajouter au « Path » dans la fenêtre « Current Folder » (Figure 50). pour faire apparaître le menu contextuel. Sélectionner ensuite **Add to Path** puis **Selected Folders and Subfolders.**

Il est également possible d'utiliser cette procédure pour supprimer des dossiers du « Path » en sélectionnant **Remove from Path** puis **Selected Folders and Subfolders.**

Afin de pouvoir utiliser l'ensemble des fichiers de simulation utilisé dans l'ouvrage, ajouter le dossier **Modelisation_Multi_Physique_Modeles** dans le « Path » de MATLAB.

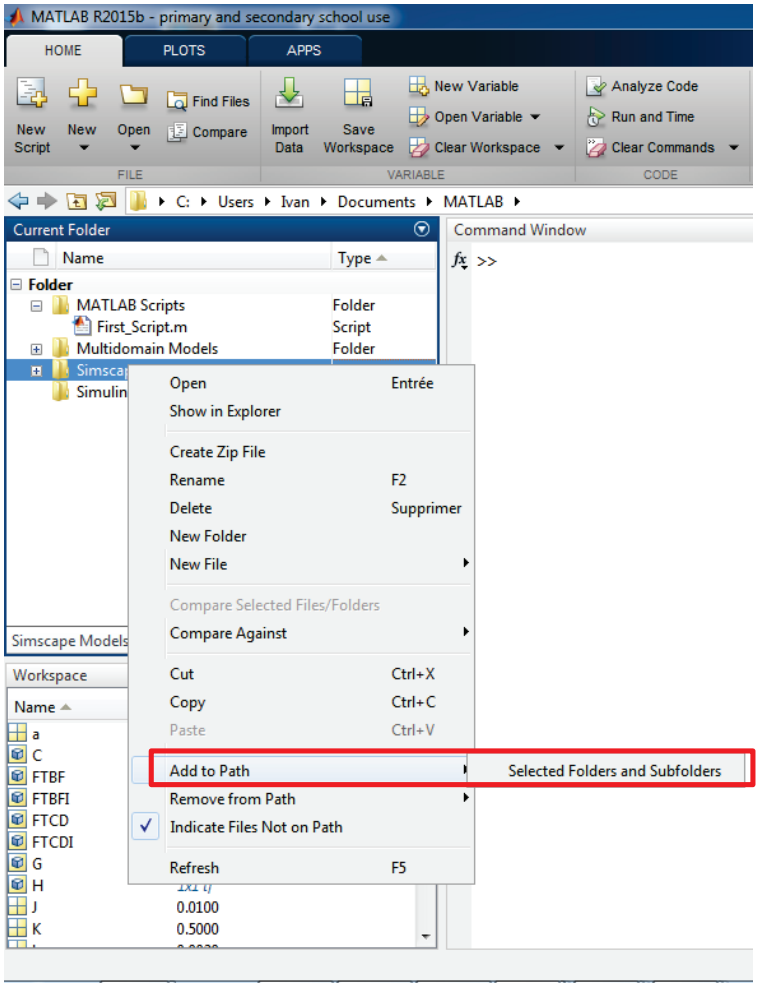


Figure 50 : ajout de deossier au « Path » pour la session courante

Les dossiers qui n'appartiennent pas au « path » apparaissent éclaircis dans la fenêtre **Current Folder** de l'environnement **MATLAB**.

III. Stratégie de conception d'un modèle multi-physique

A. Lien avec le diagramme Chaîne d'énergie/Chaîne d'information

Le descripteur chaîne d'énergie/chaîne d'information est particulièrement adapté à une approche de modélisation multi-physique. Il permet de mettre en évidence la structure de la transformation de l'énergie dans le système et ses interactions avec le dispositif de commande. De plus il est possible de réaliser une analogie complète entre ce descripteur et le modèle réalisé avec **MATLAB-Simulink**.

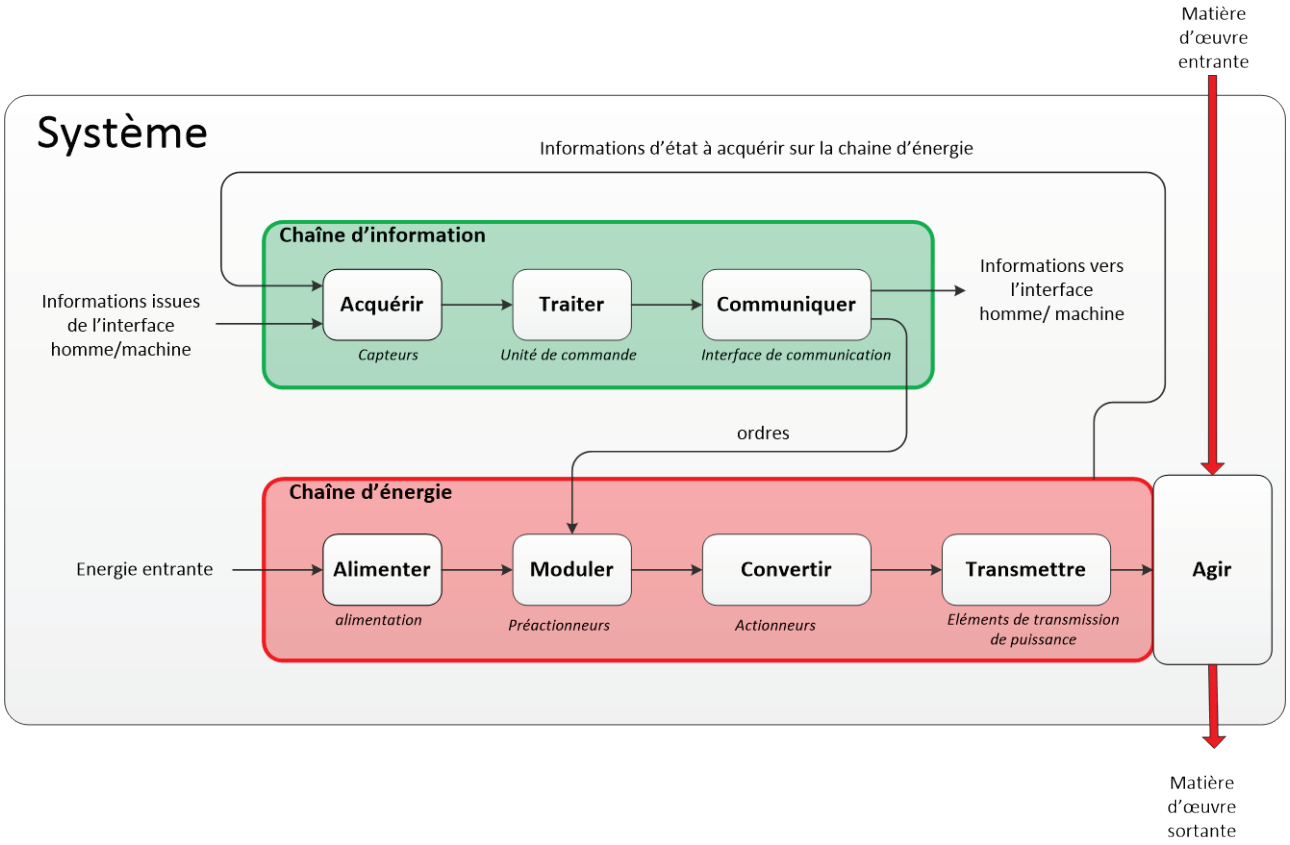


Figure 51 : le diagramme Chaîne d'énergie / Chaîne d'information

Pour mettre en place une stratégie de modélisation multi-physique, il est possible de s'appuyer sur ce descripteur en reliant les parties de ce diagramme à l'outil de modélisation à privilégier. Il s'agit de dégager une tendance, chaque problématique de modélisation est unique et de nombreuses stratégies sont envisageables.
Il aurait été également possible d'utiliser un descripteur issu du langage **SysML** comme le diagramme de blocs internes qui permet d'avoir le même type d'approche et de niveau de description que le diagramme chaîne d'énergie / chaîne d'information.

Le diagramme de la Figure 52 illustre cette démarche.

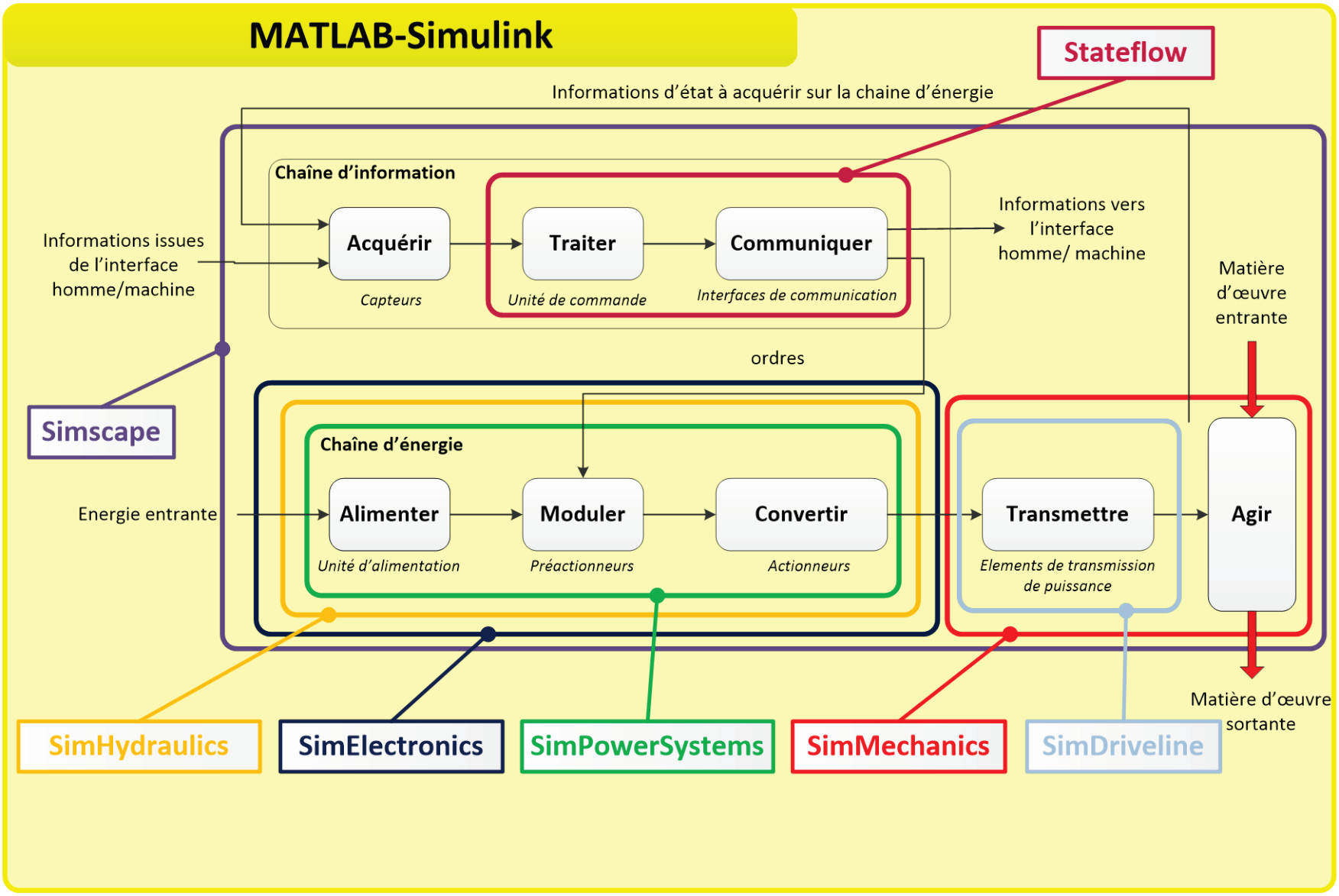


Figure 52 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB

IV. Application à un pilote hydraulique de bateau

Afin d'illustrer cette approche, nous allons appliquer la démarche proposée à un pilote hydraulique automatique de bateau. Le système fonctionne de la manière suivante.
Un moteur à courant continu actionne une pompe hydraulique à double sens de flux qui débite dans un vérin par l'intermédiaire d'un circuit de distribution hydraulique. La translation du vérin entraîne la rotation du bras de mèche sur lequel est fixée la barre du bateau. La rotation de la barre entraine la modification du cap du bateau. La chaîne d'information assure la commande du moteur en garantissant que le cap réel du bateau suive la consigne de cap.

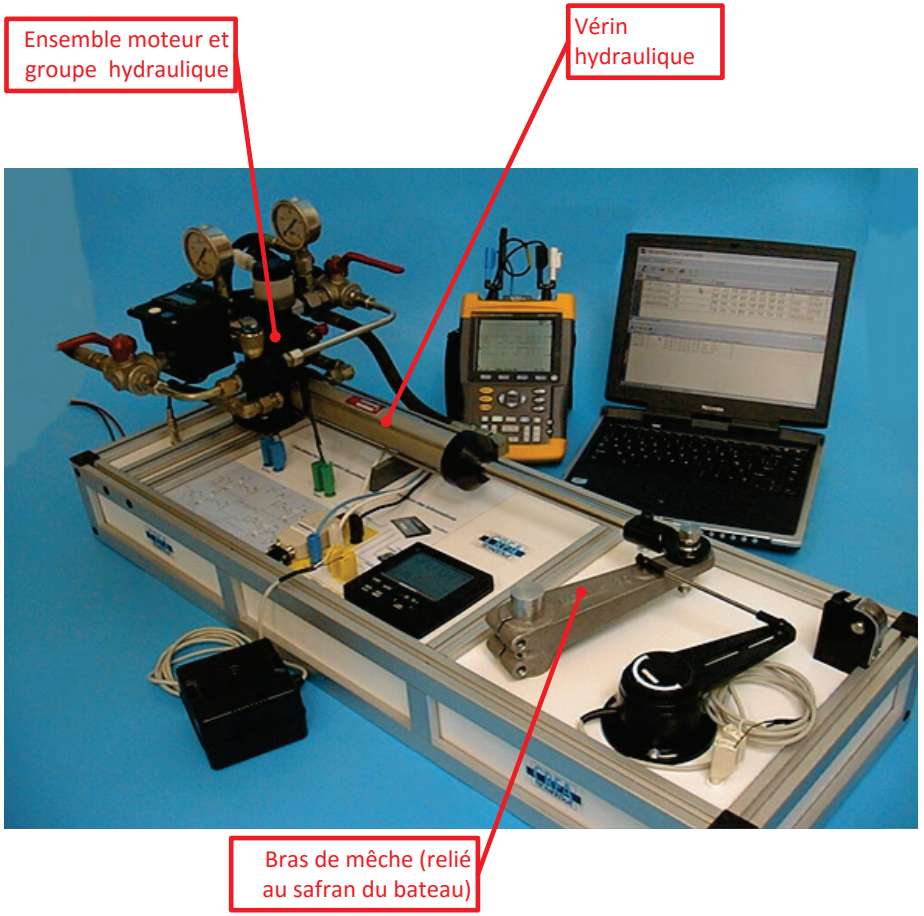


Figure 53 : photo du pilote automatique de bateau

La

Figure 54 présente le diagramme chaîne d'énergie/chaîne d'information du pilote de bateau et les outils choisis pour modéliser les différentes parties du système.

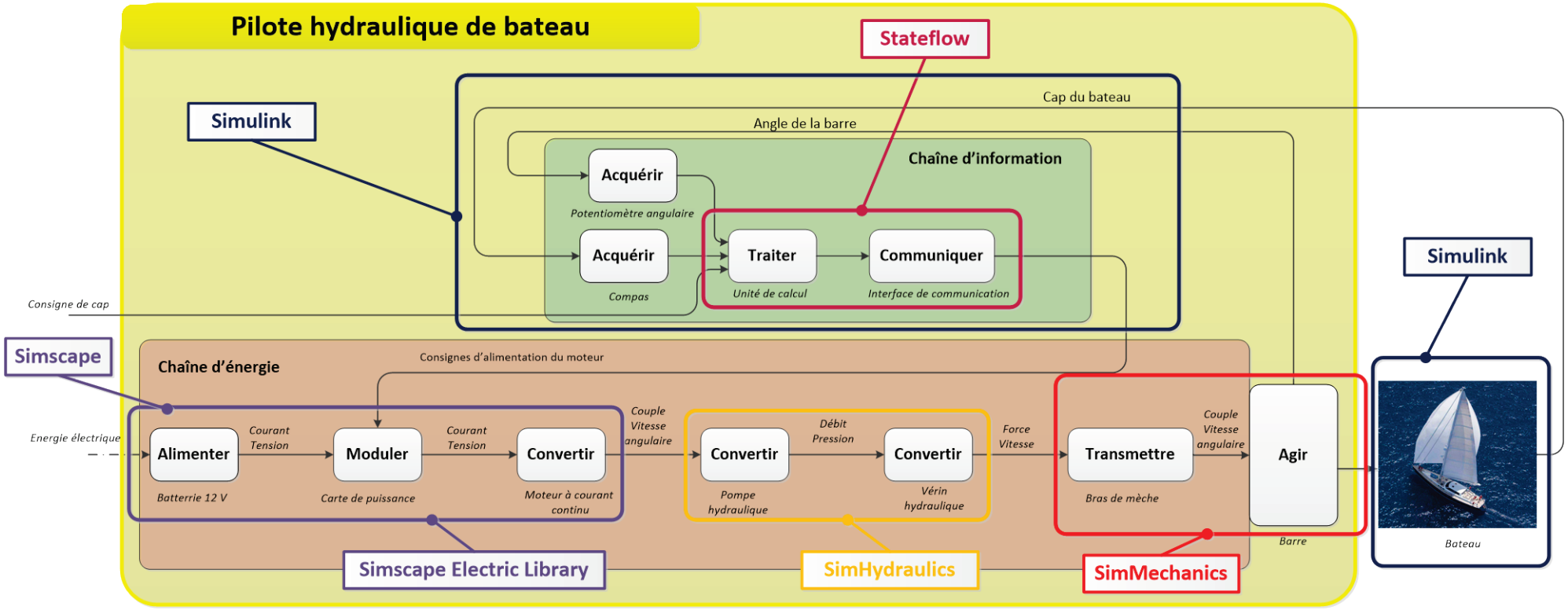


Figure 54 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB

B. Modèle multi-physique du pilote hydraulique de bateau réalisé avec MATLAB - Simulink

Le modèle multi-physique est structuré en suivant exactement le même formalisme que le descripteur. Les chaînes d'information et d'énergie apparaissent sous la forme de sous-systèmes. Les échanges d'énergie et d'information entre les blocs du descripteur sont les mêmes que les connexions entre les sous-systèmes du modèle multi-physique.
En double cliquant sur un sous-système, il est possible d'accéder à la modélisation. Le modèle multi-physique devient également un outil d'aide à la description qui permet d'explorer de manière dynamique les composants qui réalisent les fonctions.

La Figure 55 présente une copie d'écran du modèle multi-physique réalisé avec MATLAB – Simulink.

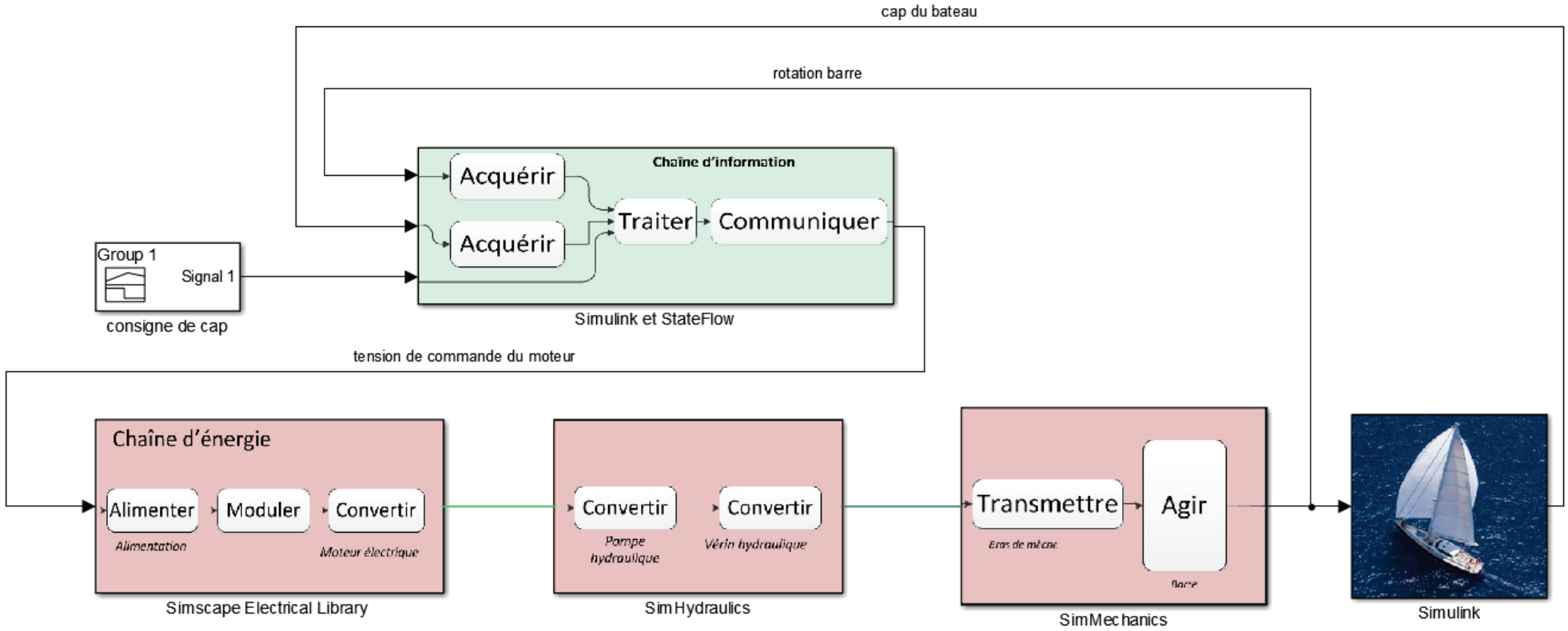


Figure 55 : copie d'écran du modèle multi-physique du pilote automatique de bateau réalisé avec MATLAB

C. Chargement et simulation du modèle

Afin de pouvoir utiliser l'ensemble des fichiers de simulation utilisé dans l'ouvrage, le dossier **Modelisation_Multi_Physique_Modeles** doit impérativement être inclus dans le « Path » de MATLAB (procédure décrite 45).

D. Visualisation des résultats issus du modèle multi-physique

Ouvrir le fichier **pilote_hydraulique.slx**. (dossier *Chapitre_1_Introduction/Pilote_hydraulique*)

Double-cliquez sur le bloc « consigne de cap » pour visualiser le signal de consigne.
La consigne de cap impose au bateau un cap de 20° pendant 300 s puis un cap de -40° pendant 700 s.

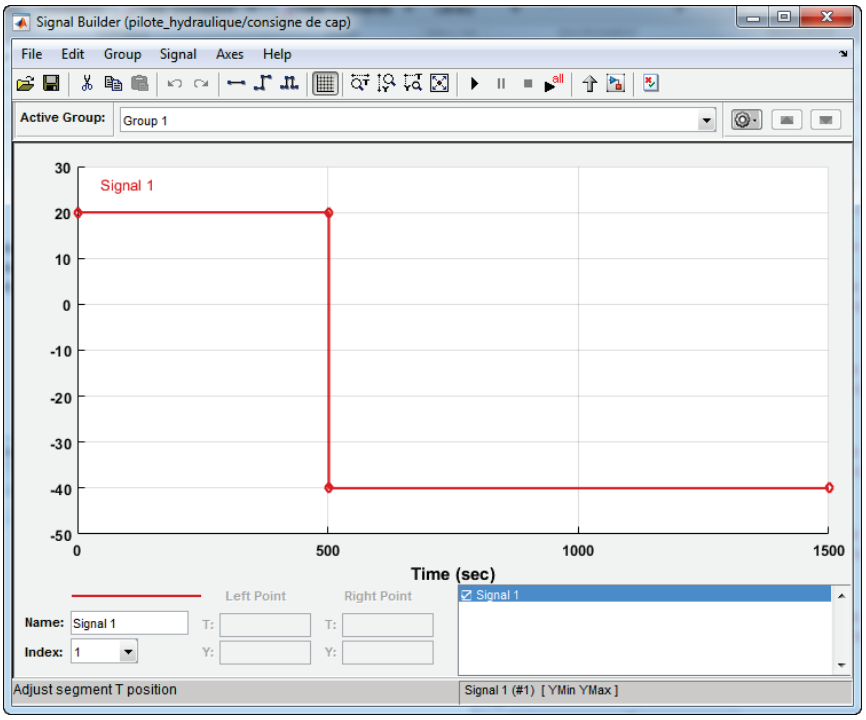


Figure 56 : visualisation de la consigne de cap

Lancer la simulation en cliquant sur  .
Lors du lancement de la simulation une fenêtre **Mechanics Explorers** s'ouvre permettant de visualiser dynamiquement les mouvements de la partie mécanique modélisée à l'aide de **SimMechanics** (Figure 57).

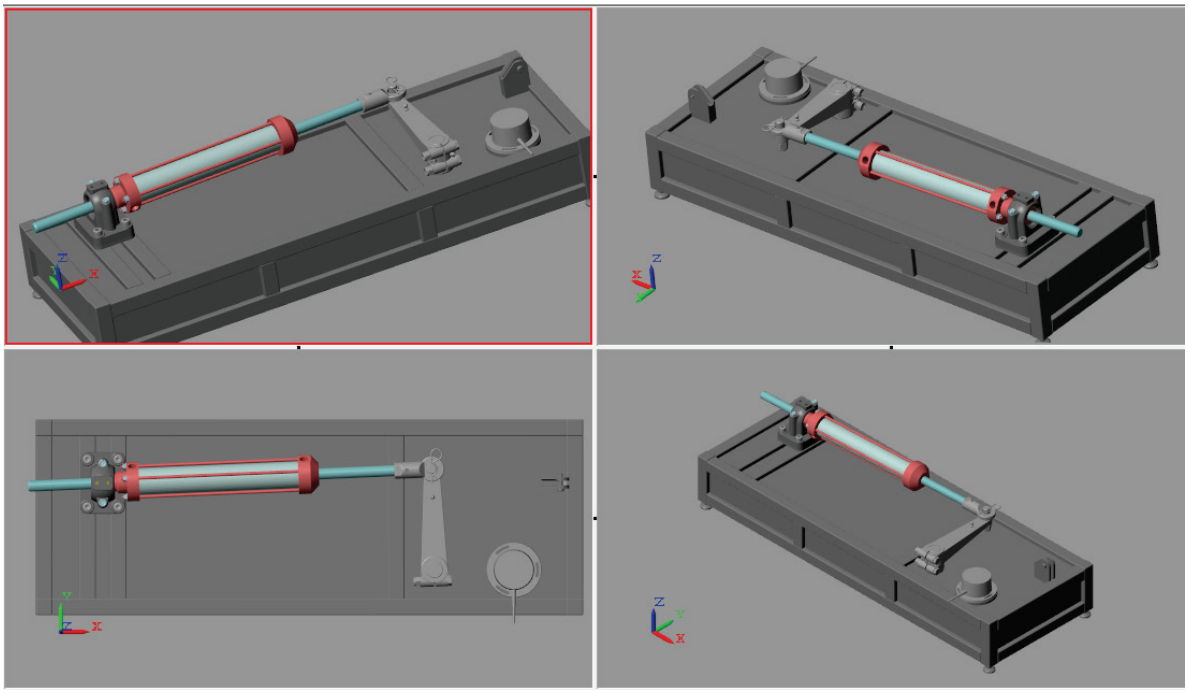


Figure 57 : fenêtre de visualisation de SimMechnanics

Une barre de visualisation permet de choisir le mode d'observation du modèle.

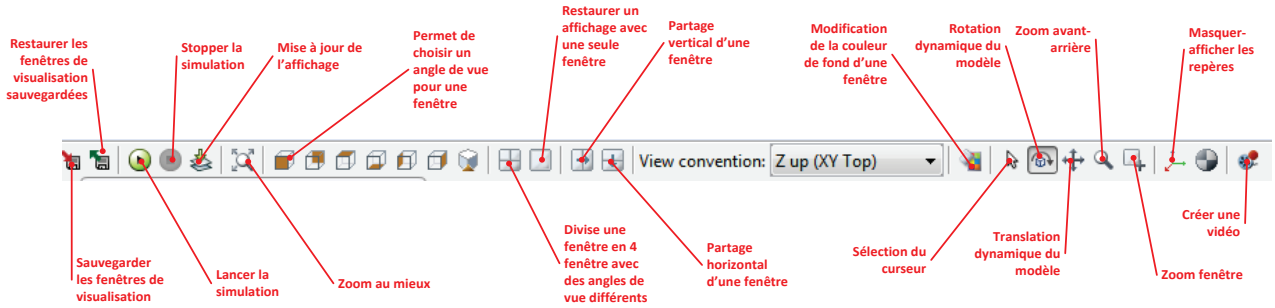


Figure 58 : barre de visualisation de SimMechanics

Cliquez sur l'icône  dans la partie supérieure de la fenêtre **Mechanics Explorers** (Figure 58) pour revenir à un affichage avec une seule fenêtre (Figure 59).

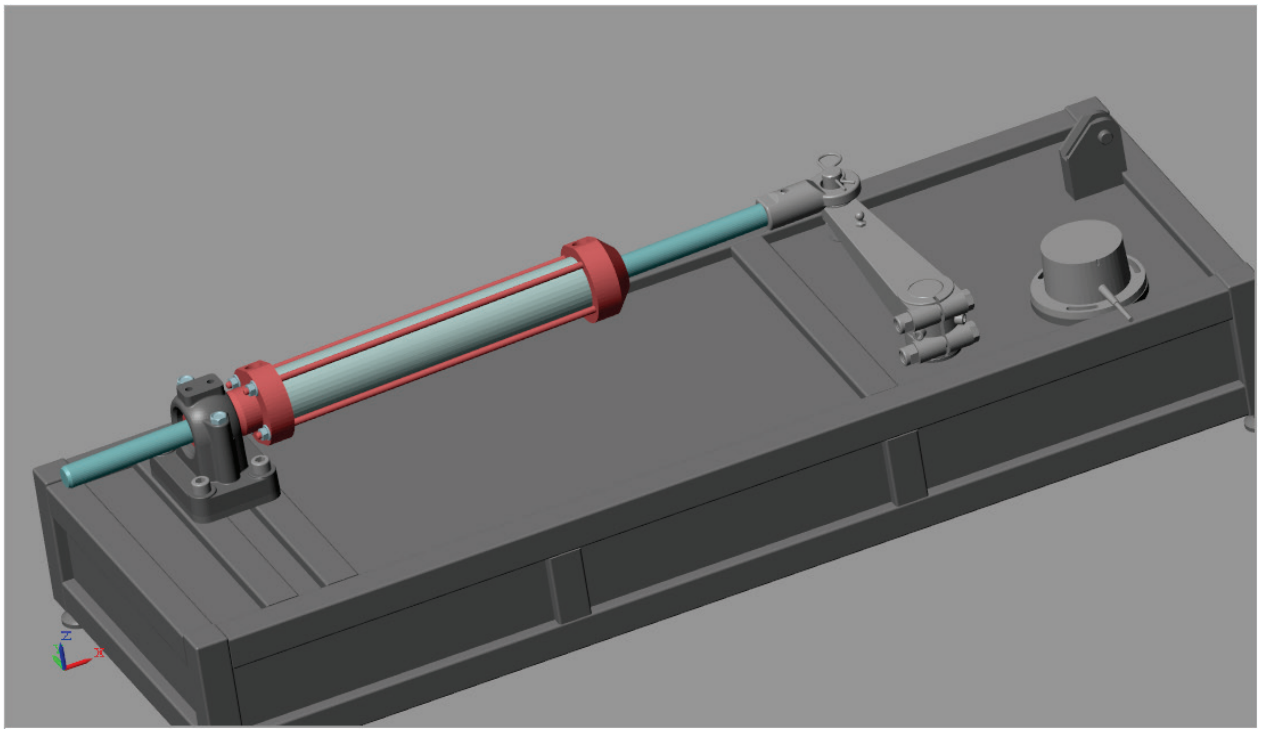


Figure 59 : fenêtre de visualisation de SimMechanics

Dans la partie inférieure de la fenêtre une barre temporelle permet de visualiser l'évolution temporelle de la simulation. Il est possible d'accélérer la visualisation à l'aide d'un curseur ou de revenir à un instant quelconque de la simulation (Figure 60).

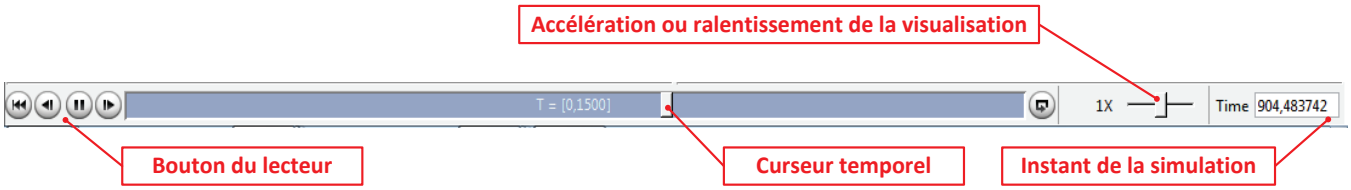


Figure 60 : barre temporelle de SimMechanics

A la fin de la simulation, les résultats sont facilement exploitables en double-cliquant sur les scopes. Il est possible de relever toutes les grandeurs physiques du modèle.
On peut visualiser la consigne de cap et le cap effectif suivi par le bateau :
Cliquer sur la mise à l'échelle automatique pour visualiser toute la courbe 

Il faudra prendre l'habitude d'utiliser la commande de mise à l'échelle automatique à l'ouverture de chaque scope pour bien visualiser le signal

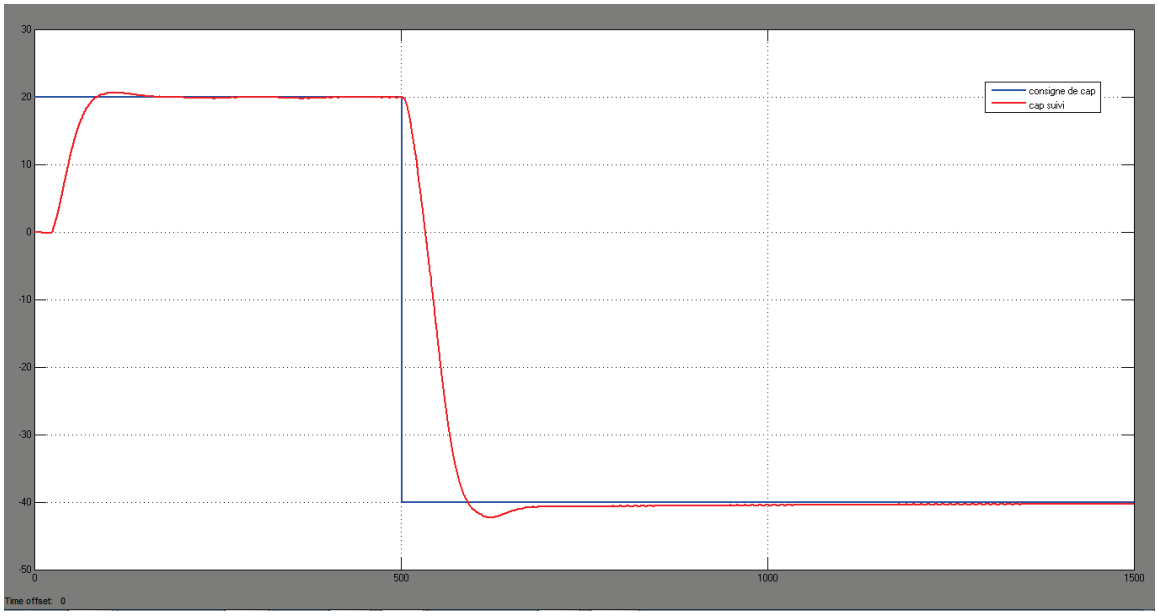


Figure 61 : visualisation de la consigne de cap et du cap effectif suivi par le bateau

Le courant, la tension et le couple du moteur :

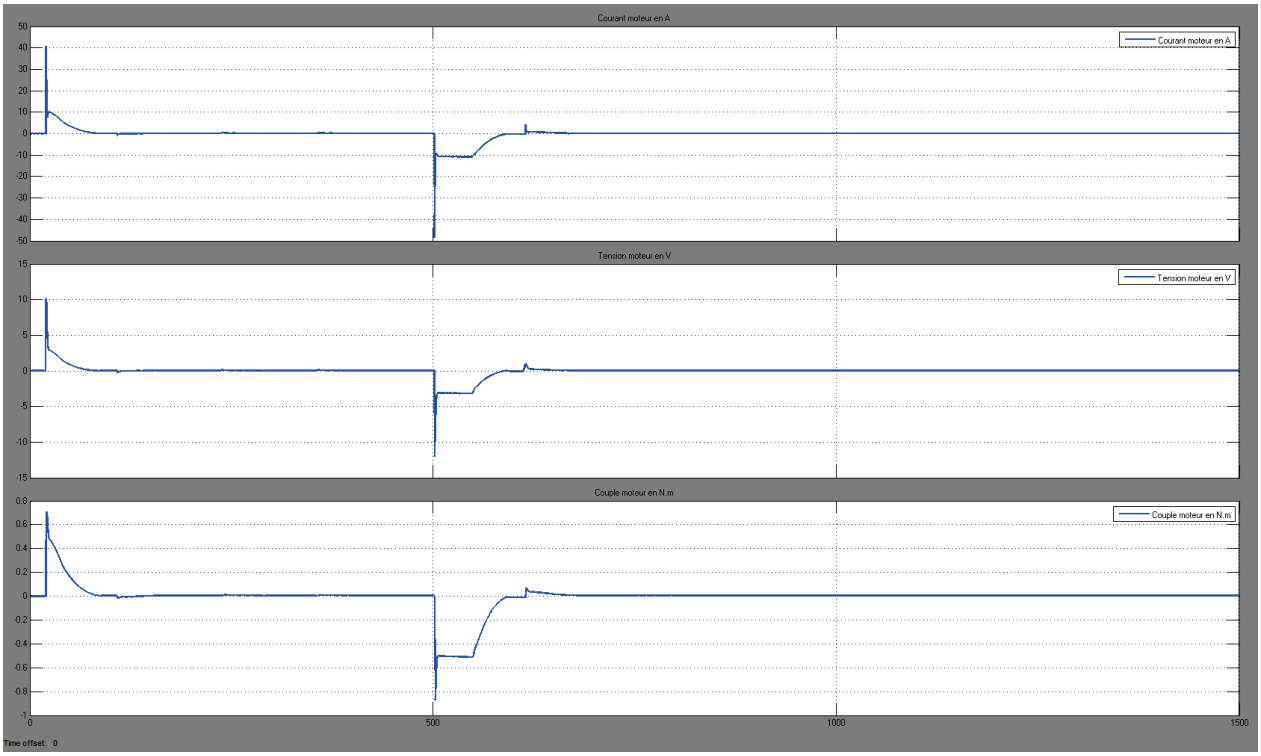


Figure 62 : visualisation des grandeurs relatives au moteur

L’angle de rotation de la barre :

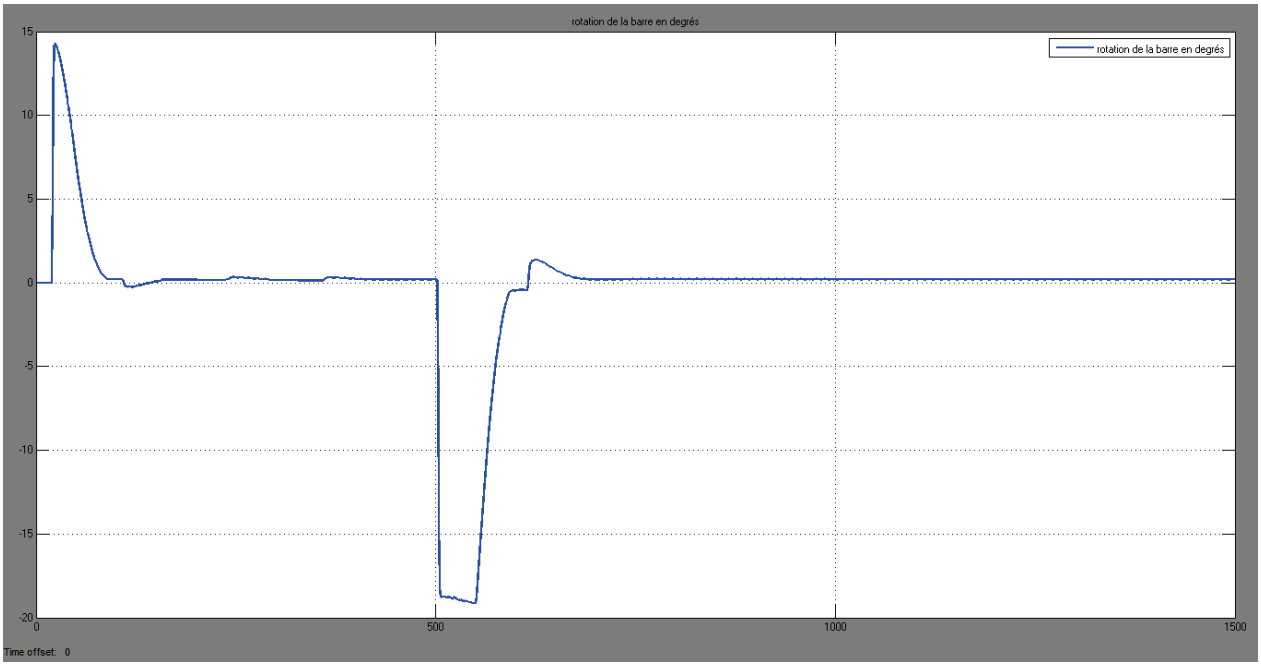


Figure 63 : visualisation de l’angle de rotation de la barre

Les pressions dans les chambres avant et arrière du vérin :

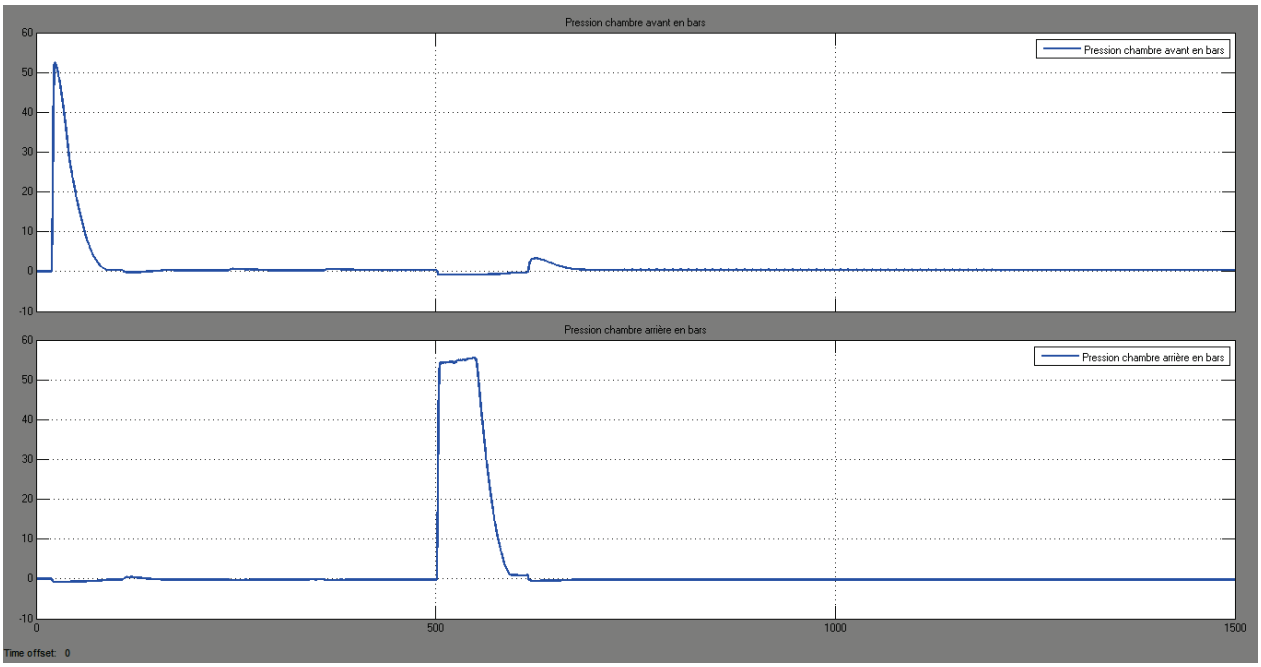


Figure 64 : visualisation des pressions dans les chambres avant et arrière du vérin

Les débits dans les chambres avant et arrière du vérin :

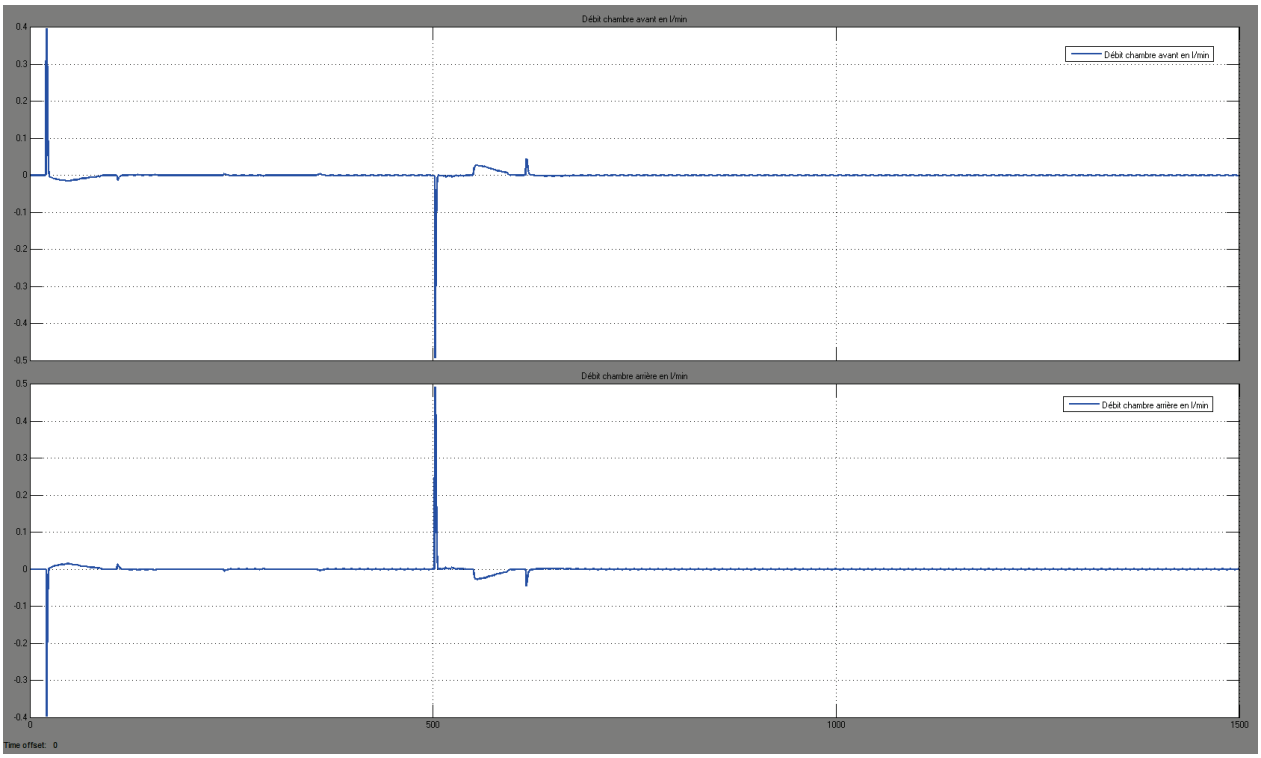


Figure 65 : visualisation des débits dans les chambres avant et arrière du vérin

E. Exploration du modèle

1. Exploration du modèle de la chaîne d'information : Simulink et Stateflow

La chaîne d'information est modélisée à l'aide de **Simulink** et de **Stateflow**. Le fonctionnement du système nécessite une boucle pour asservir la barre en position. Cette boucle est régulée par un correcteur PID. La seconde boucle permet de respecter la consigne de cap. La logique est programmée sous forme de diagrammes d'états avec **Stateflow** et permet au système de régulation de se mettre en action uniquement si le bateau quitte son cap durant plus de 20 s.

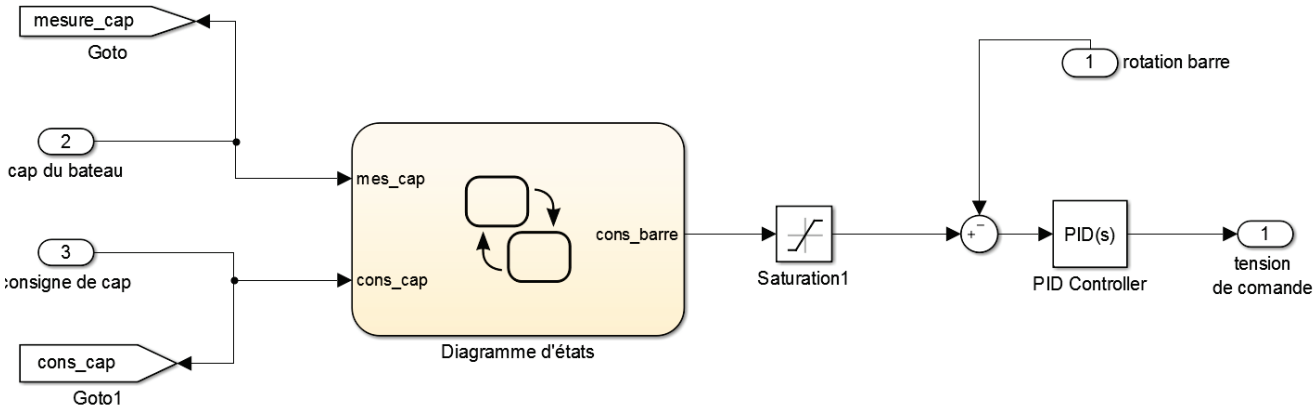


Figure 66 : modèle MATLAB – Simulink de la chaîne d'information du pilote automatique

Diagramme d'état de la boucle de cap :

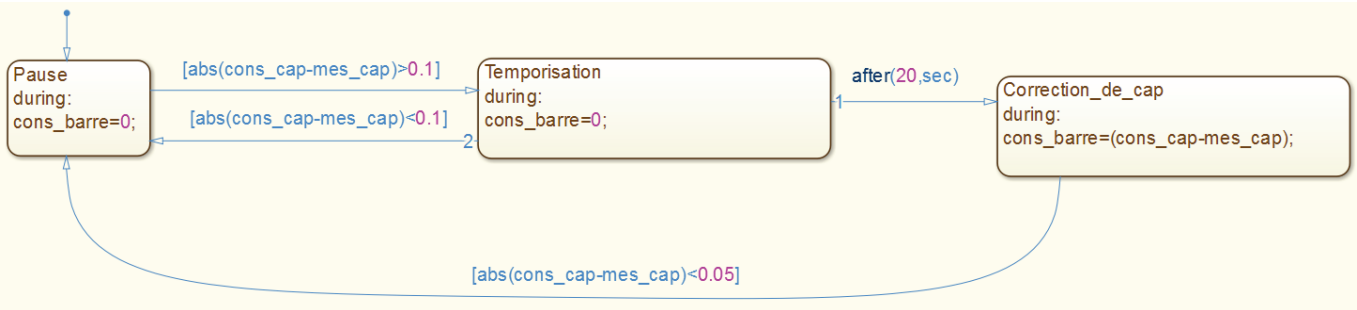


Figure 67 : modélisation de la commande de cap avec des diagrammes d'états.

2. Exploration du modèle de la chaîne d'énergie : Simscape Electric Library

La partie électrique de la chaîne d'énergie est modélisée à l'aide des éléments de base de la bibliothèque de composants électriques de **Simscape**. Le moteur à courant continu est modélisé à l'aide d'une inductance, d'une résistance et d'un composant permettant de convertir la puissance électrique en puissance mécanique. Des mesures de courant et de tension sont réalisées dans le circuit électrique, une mesure de couple est également effectuée sur l'arbre moteur.

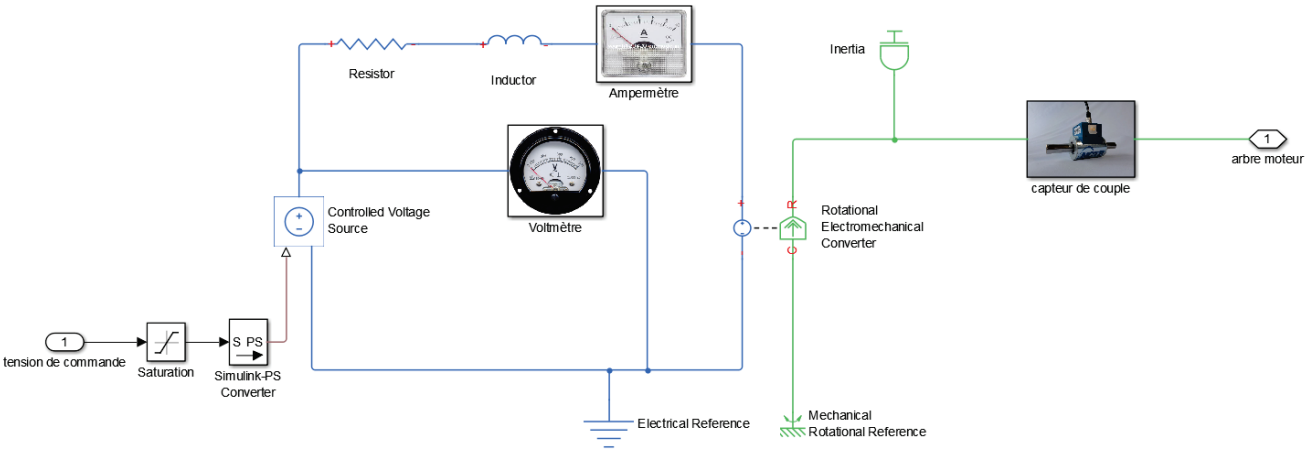


Figure 68 : modèle Simscape de la partie électrique de la chaîne d'énergie du pilote automatique

3. Exploration de la chaîne d'énergie : SimHydraulics

La partie hydraulique de la chaîne d'énergie est réalisée à l'aide de composants extraits de la bibliothèque **SimHydraulics**. La pompe à double sens de flux à cylindrée variable débite directement dans un vérin double effet. Des limiteurs de pression permettent l'évacuation du fluide lorsque la tige du vérin est en butée. Des mesures de pression et de débit sont réalisées en différents points du circuit hydraulique.

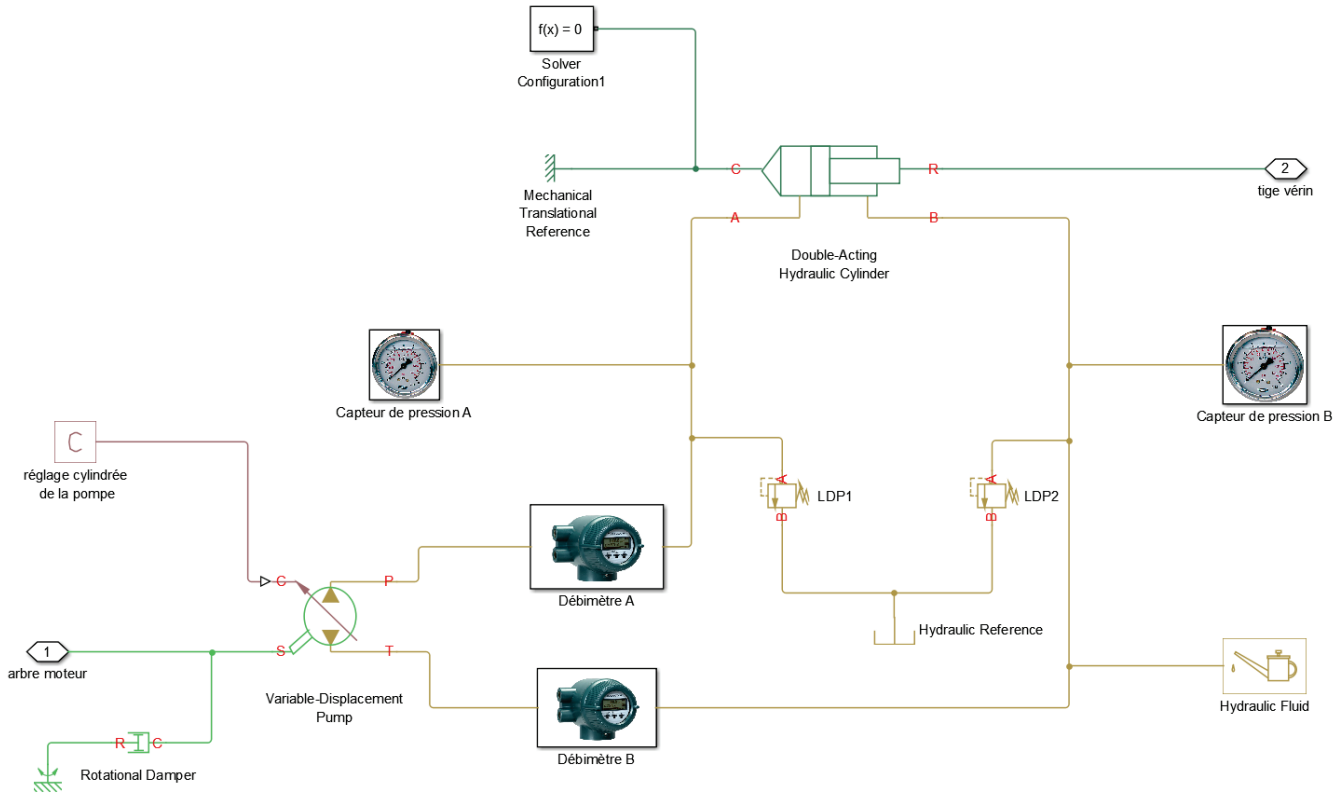


Figure 69 : modèle SimHydraulics de la partie hydraulique de la chaîne d'énergie du pilote automatique

4. Exploration de la chaîne d'énergie : SimMechanics 2G

La partie mécanique de la chaîne d'énergie est réalisée à l'aide de **SimMechanics** 2nde génération. Le modèle se présente comme un graphe des liaisons. Il fait apparaître les solides et les liaisons entre les solides. Des masques (images) ont été appliqués sur les solides afin d'améliorer la compréhension du modèle. Des mesures de déplacement de la tige du vérin et de rotation de la barre sont réalisées. Un bloc permet également de définir l'action non linéaire des efforts exercés par l'eau sur le safran. Un modèle **SimMechanics** peut se construire directement avec **MATLAB** ou être importé à partir d'un logiciel de CAO comme **Solidworks**.

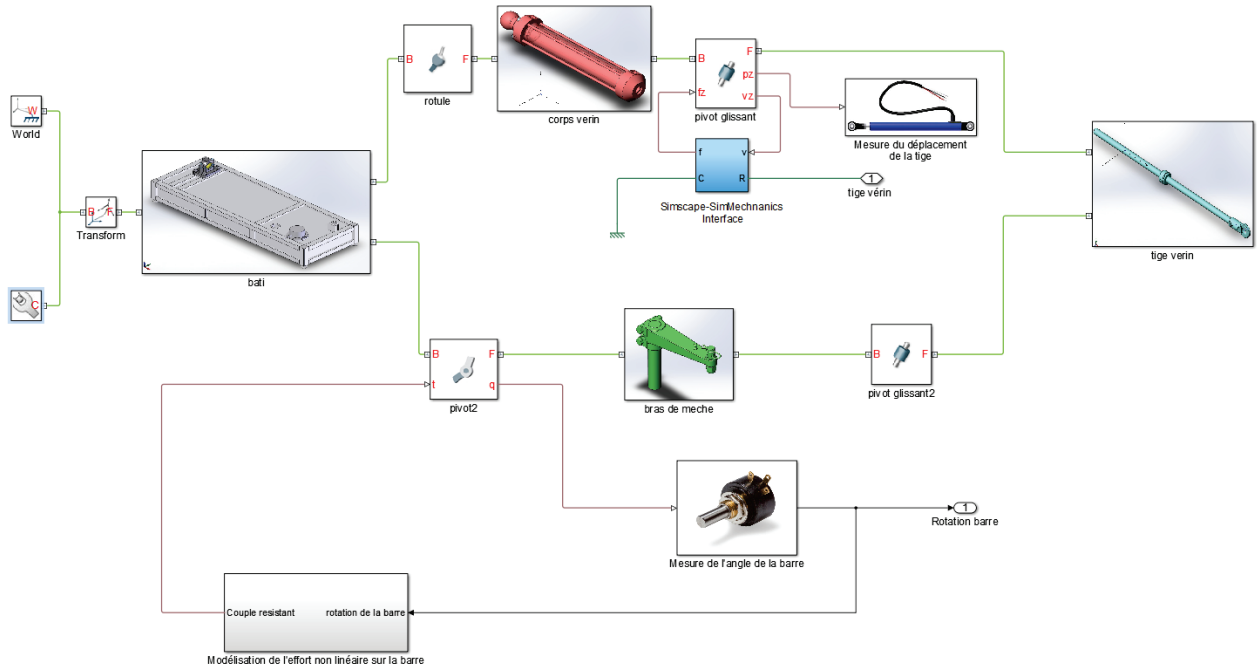


Figure 70 : Modélisation SimMechanics de la structure du pilote automatique

5. Exploration de la chaîne d'énergie : Simulink

Le modèle qui permet de prendre en compte la dynamique du bateau et les perturbations qu'il subit est réalisé avec **Simulink**.

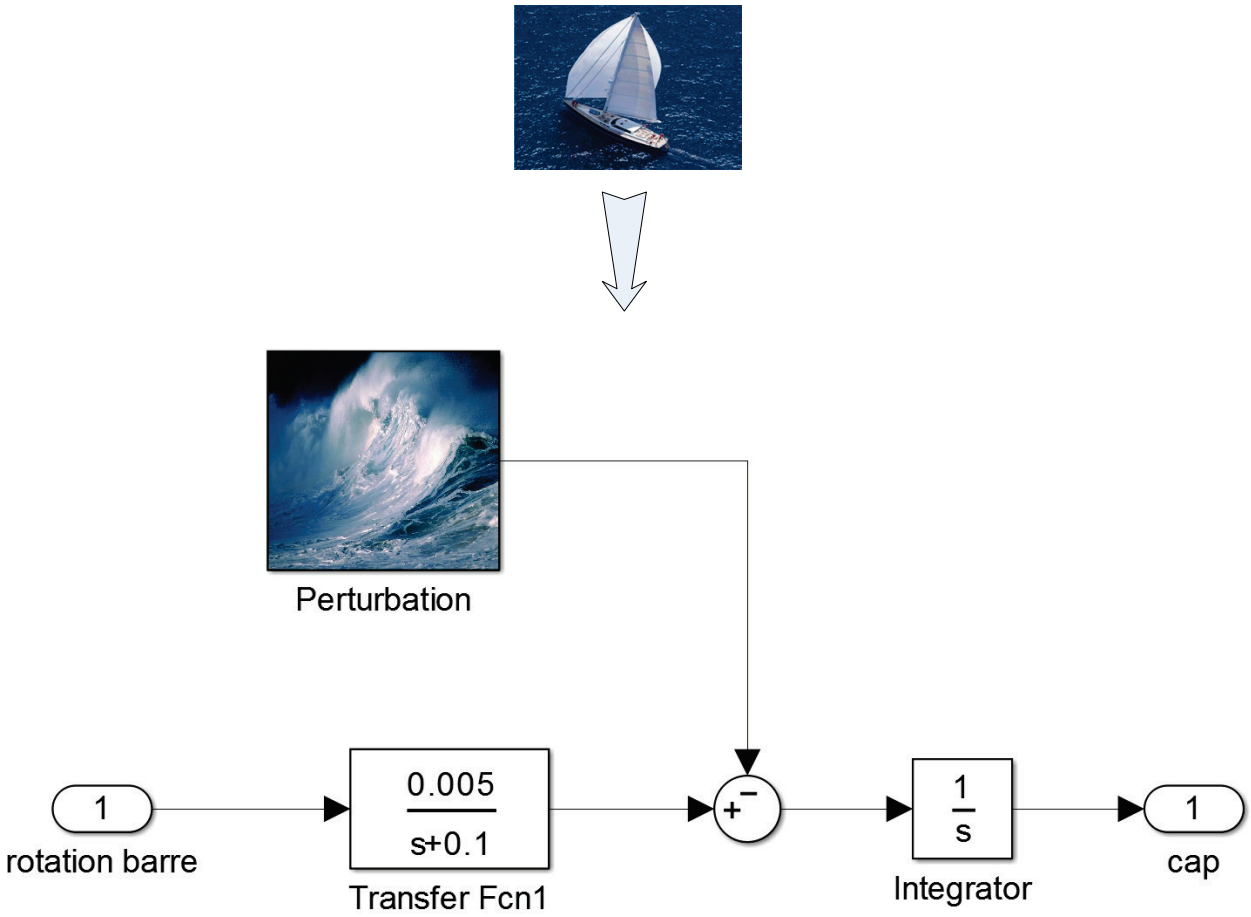


Figure 71 : Modélisation Simulink du comportement dynamique du bateau

F. Pilotage interactif du modèle

Il est également possible d'illustrer différemment le modèle en associant chaque sous-système à son équivalent visuel sur le système réel et de le piloter de manière interactive. Pour cela la bibliothèque de Simulink **Dashboard** sera utilisée. Son utilisation et le paramétrage des blocs est abordée plus tard dans l'ouvrage (page 157).

Ouvrir le fichier **« pilote_hydraulique_dashboard2015b.slx »**. Ce modèle est le même que le précédent, un autre point de vue a été choisi pour l'illustration des sous-systèmes et des éléments de paramétrage et de visualisation interactifs ont été ajoutés. Nous verrons l'importance qu'il faut accordée à la présentation du modèle afin de le rendre le plus accessible possible pour l'analyse et la modification.

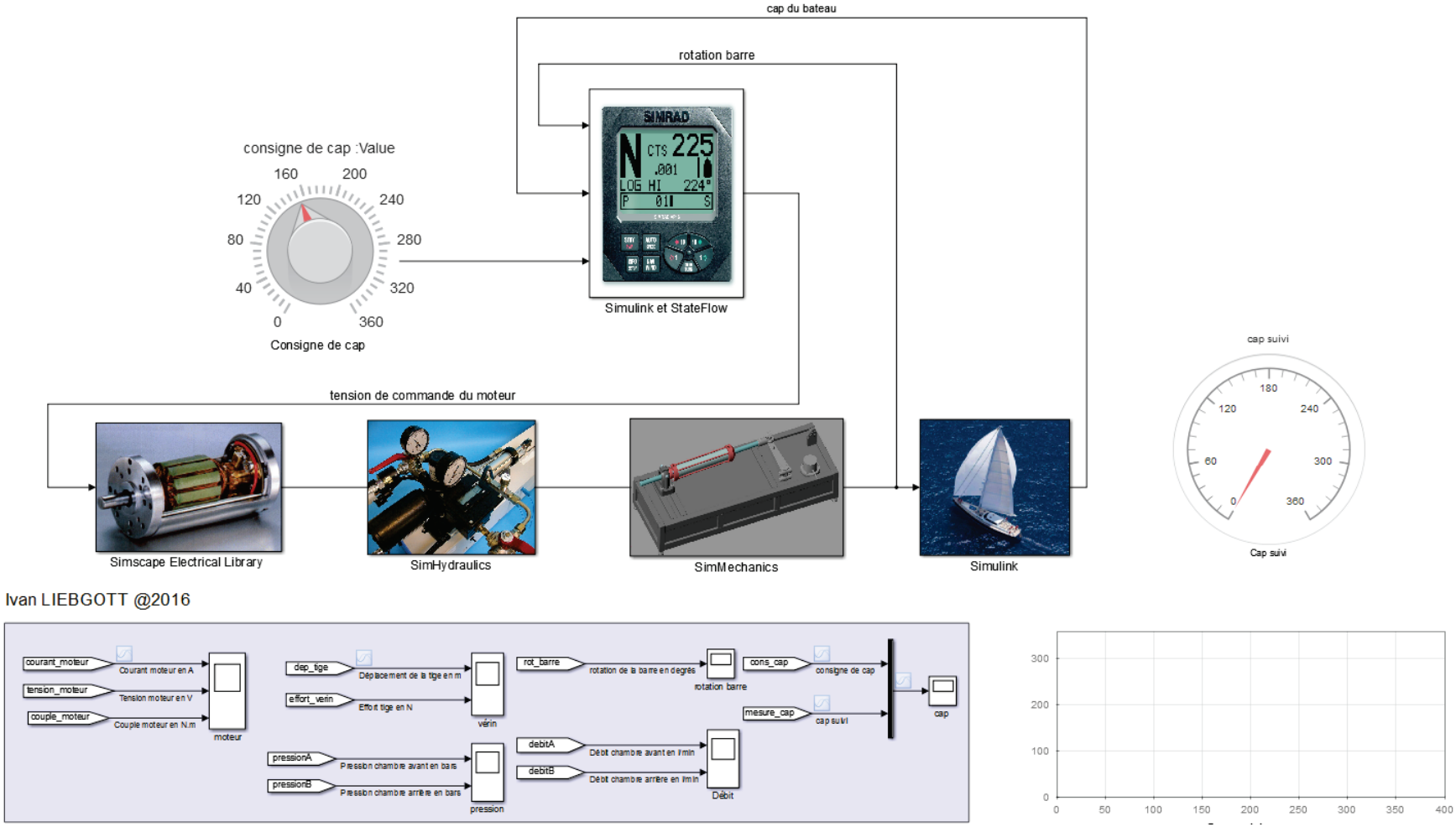


Figure 72 : modèle du pilote hydraulique avec pilotage interactif

Lancer la simulation et **Modifier** la consigne de cap en agissant sur le bouton rotatif durant la simulation.

Observer le cap suivi dans le scope interactif et sur le quadrant de visualisation.

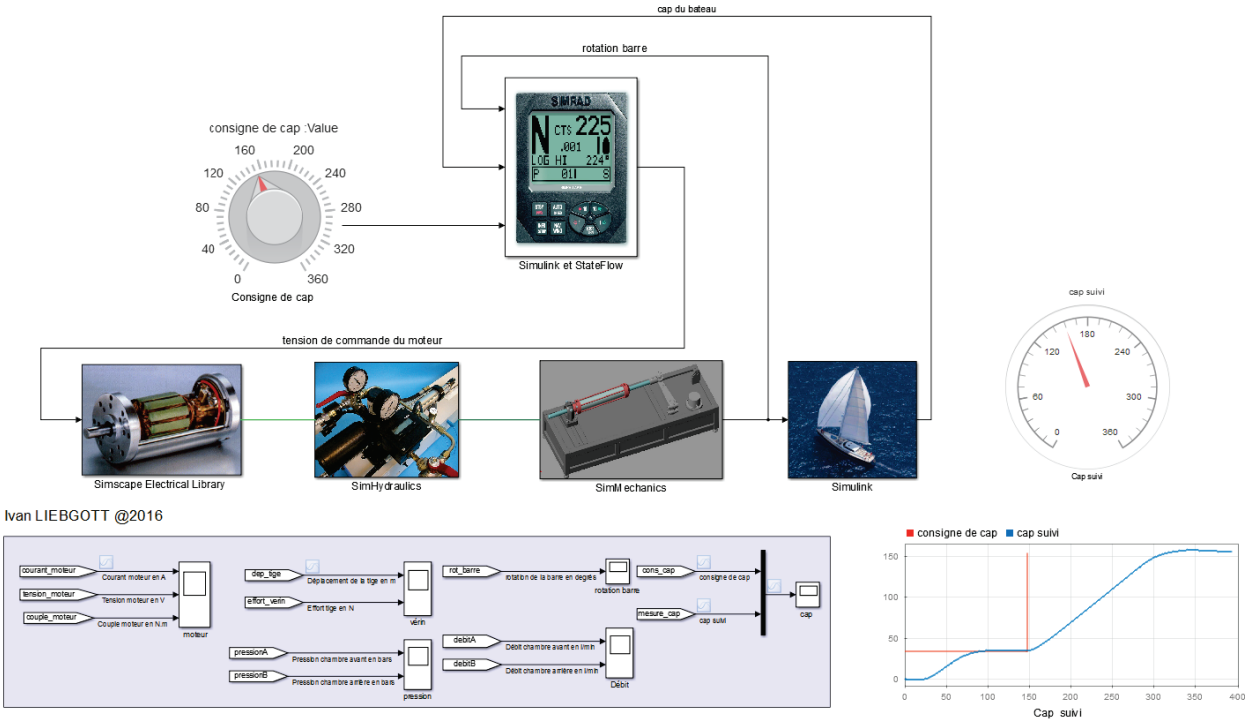


Figure 73 : utilisation du pilotage interactif d'un modèle

V. Exemples de modèles multi-physique et exploitations possibles

A. Le robot Maxpid

Le robot Maxpid est un sous-système d'un robot initialement utilisé pour la cueillette de fruit et le tri de déchets. Maxpid est asservi et peut orienter angulairement son bras conformément à une consigne. Un système de transformation de mouvement de type vis-écrou est utilisé pour transformer le mouvement de rotation de l'axe de sortie du moteur (vis) en mouvement de rotation du bras.

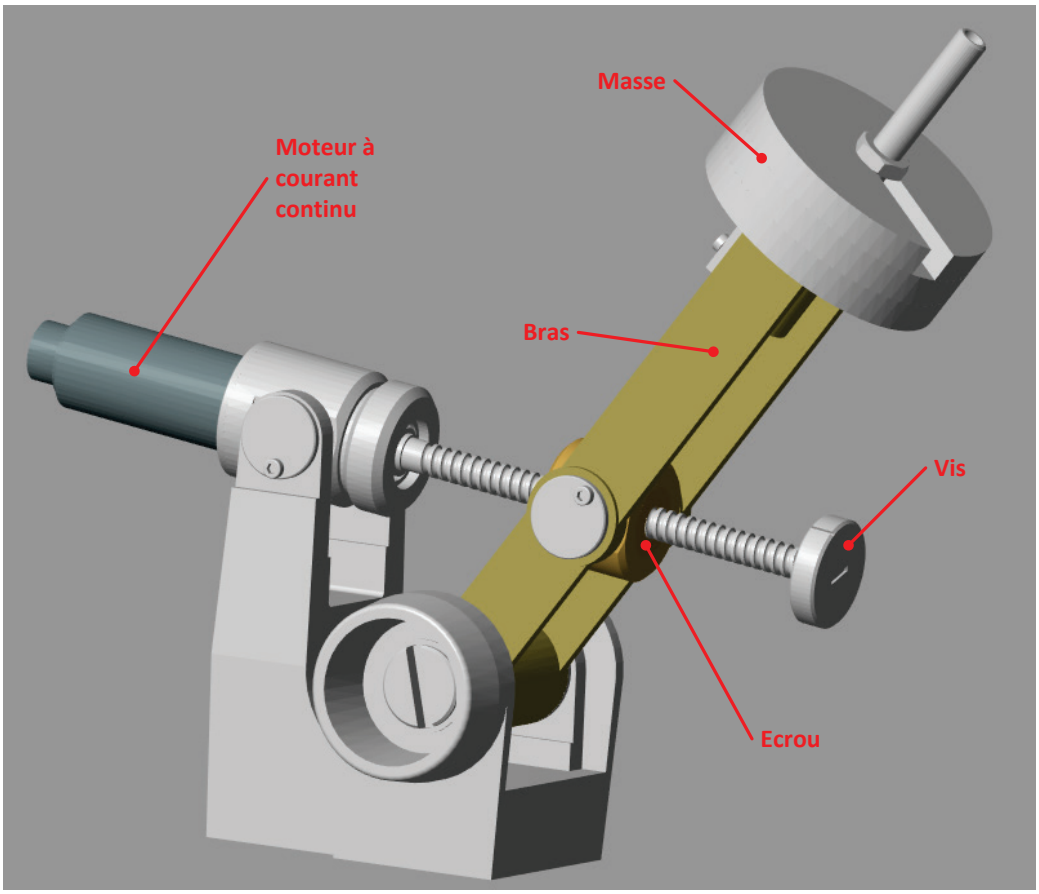


Figure 74 : présentation du robot Maxpid

La nécessité d'étudier l'asservissement de position du bras de Maxpid impose de disposer d'un modèle validé. Dans ce cas précis ce modèle est très difficile à mettre en place analytiquement car la partie mécanique est complexe et les équations différentielles qui caractérisent son comportement sont non linéaires. La résolution de ces équations n'est possible qu'en considérant des petites variations autour d'une position donnée ce qui est très contraignant dans la démarche d'analyse.
Le recours à la modélisation multi-physique présente dans ce type de cas un avantage évident. L'ajout de la maquette 3D dans le modèle va prendre en compte tous les effets dynamiques sans que le moindre calcul ne soit nécessaire. Le modèle sera valable pour toute la plage de variation angulaire de la position bras.

Ouvrir le fichier **« Maxpid.slx », Explorer** les sous-systèmes et **Lancer** la simulation.

La fenêtre **SimMechanics Explorer** permet de visualiser les mouvements du bras du robot.

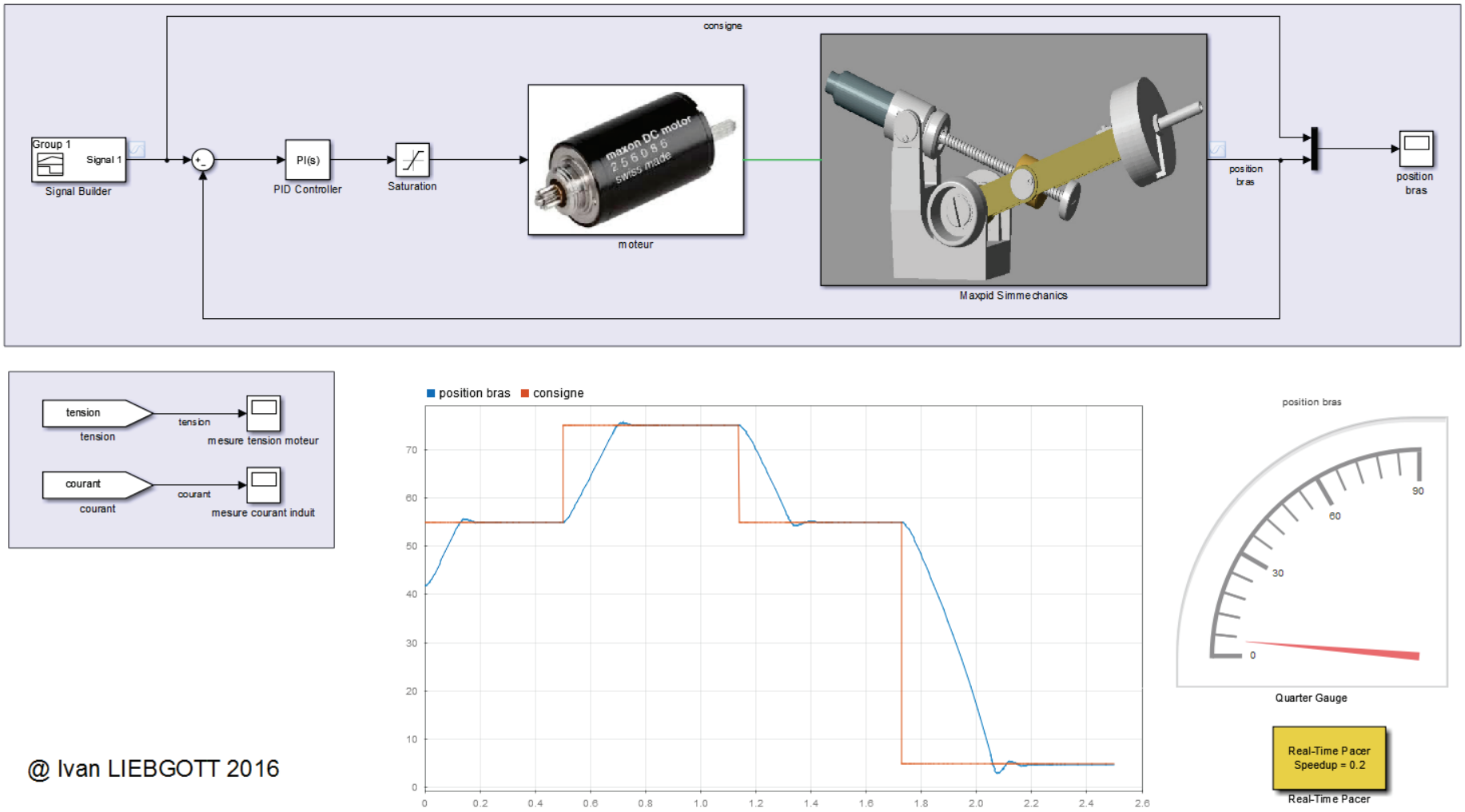


Figure 75 : modélisation multi-physique du robot MAXPID

Visualisation de la position du bras et de la consigne :

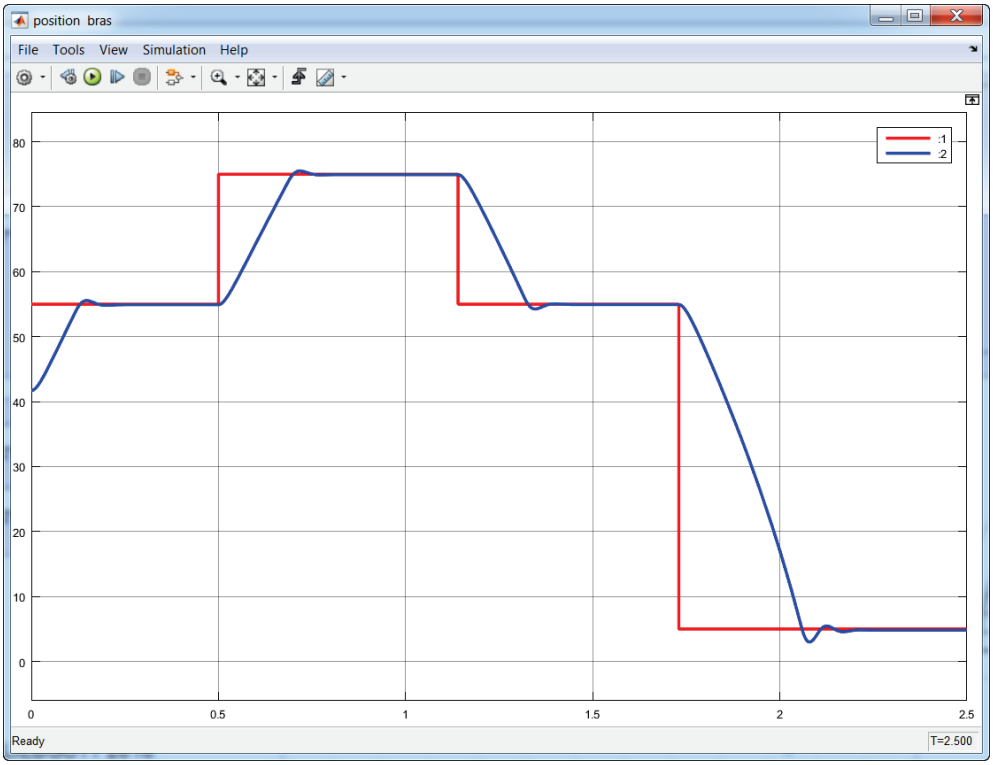


Figure 76 : position du bras du robot MAXPID

Visualisation de la tension de commande :

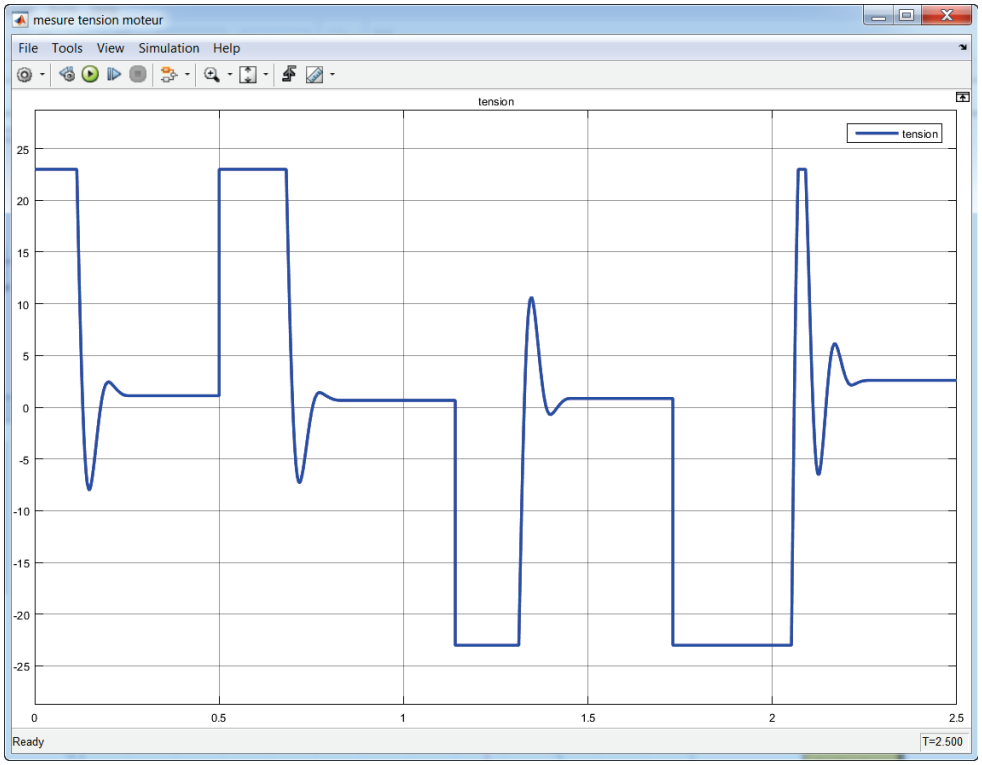


Figure 77 : visualisation de la tension de commande du moteur

Visualisation du courant dans l'induit du moteur :

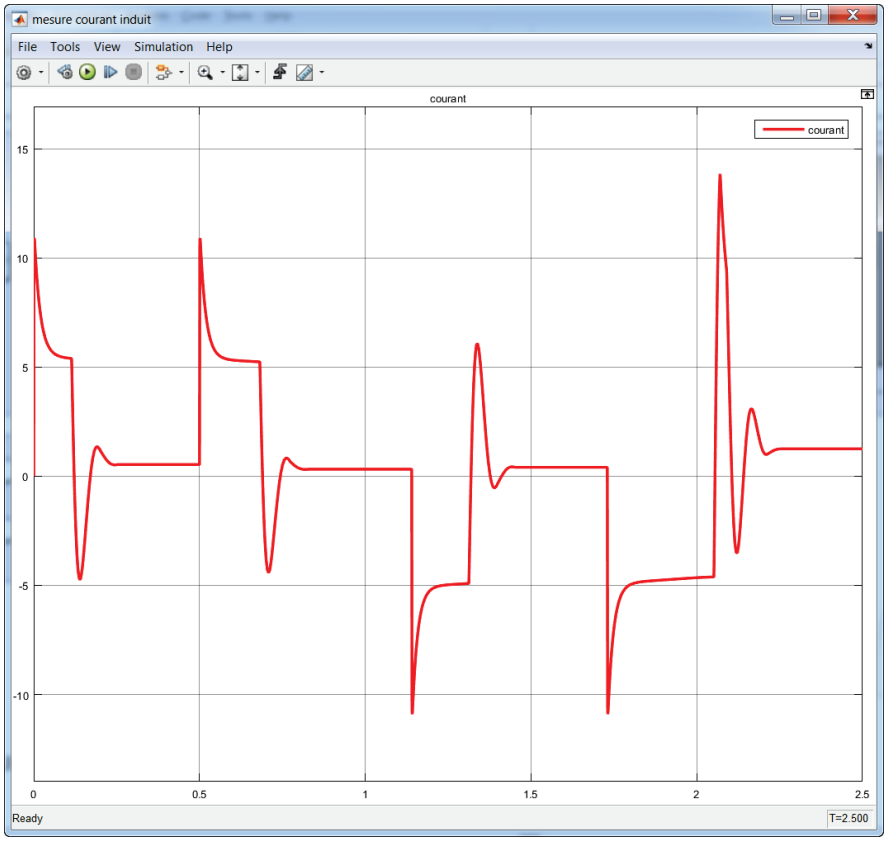


Figure 78 : visualisation du courant dans l'induit du moteur

Il est possible de modifier les paramètres de masse directement dans **SimMechanics** afin de voir l'influence de l'ajout de masse en bout de bras, et agir sur tout type de paramètre.

B. L’axe linéaire Control’X

Le système Control’X est un axe linéaire asservi. Un motoréducteur à courant continu est accouplé un système de transformation de mouvement de type poulies/courroie. Le chariot de l’axe est solidaire de la courroie et se déplace en translation.

Figure 79 : l’axe linéaire Control’X

Ouvrir le fichier **« ControlX.slx »** et **Explorer** les sous-systèmes.

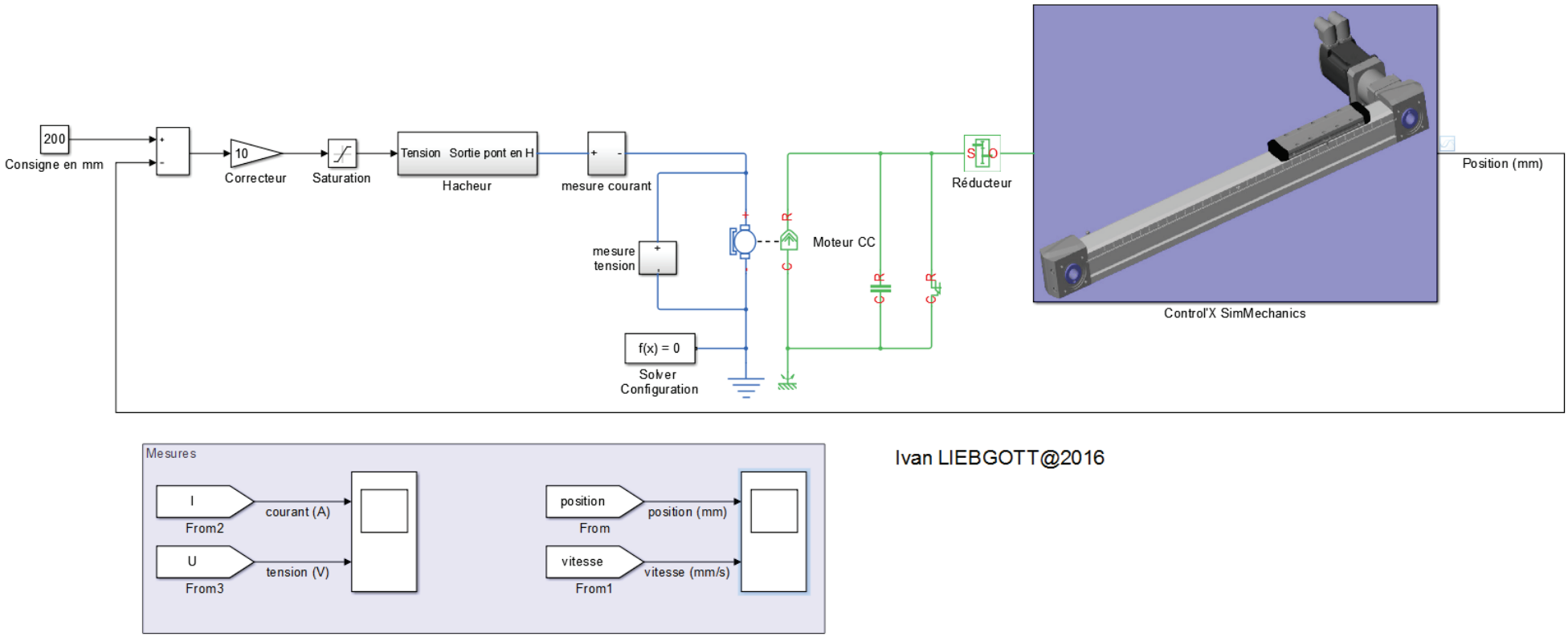


Figure 80 : modélisation multi-physique du système Control'X

Lancer la simulation et observer les résultats obtenus.

Visualisation de la vitesse et de la position de l'axe :

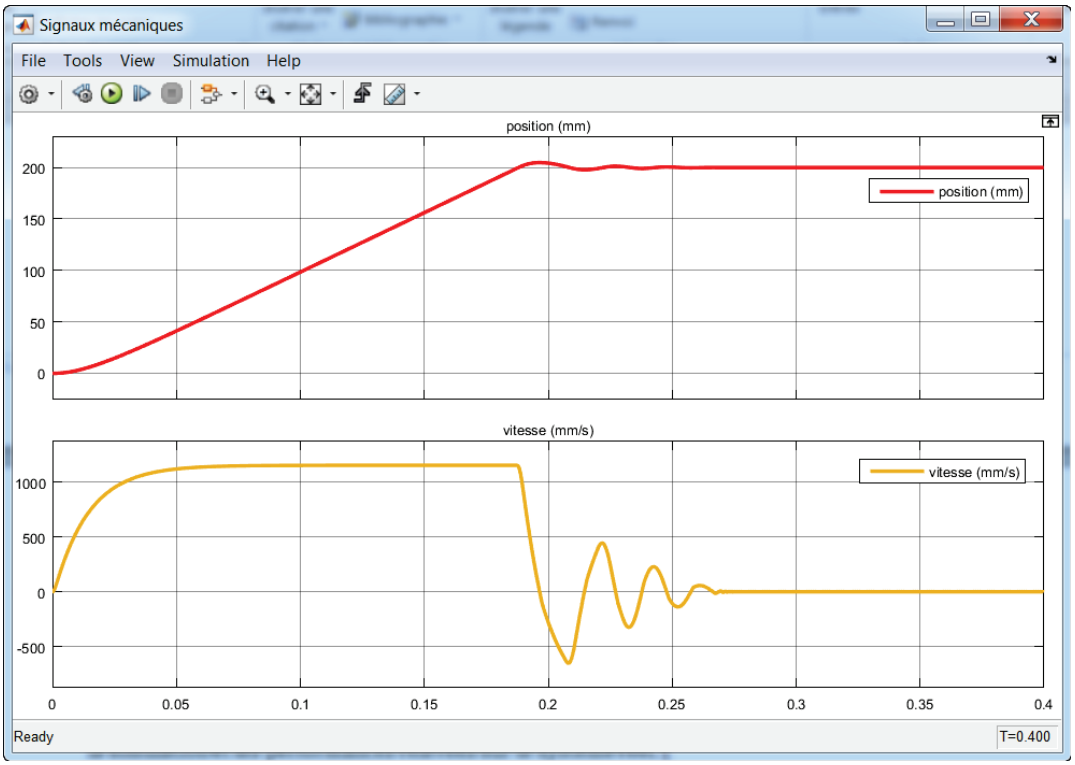


Figure 81 : visualisation de la vitesse et de la position de l'axe linéaire

Visualisation de la tension de commande du moteur et du courant dans l'induit :

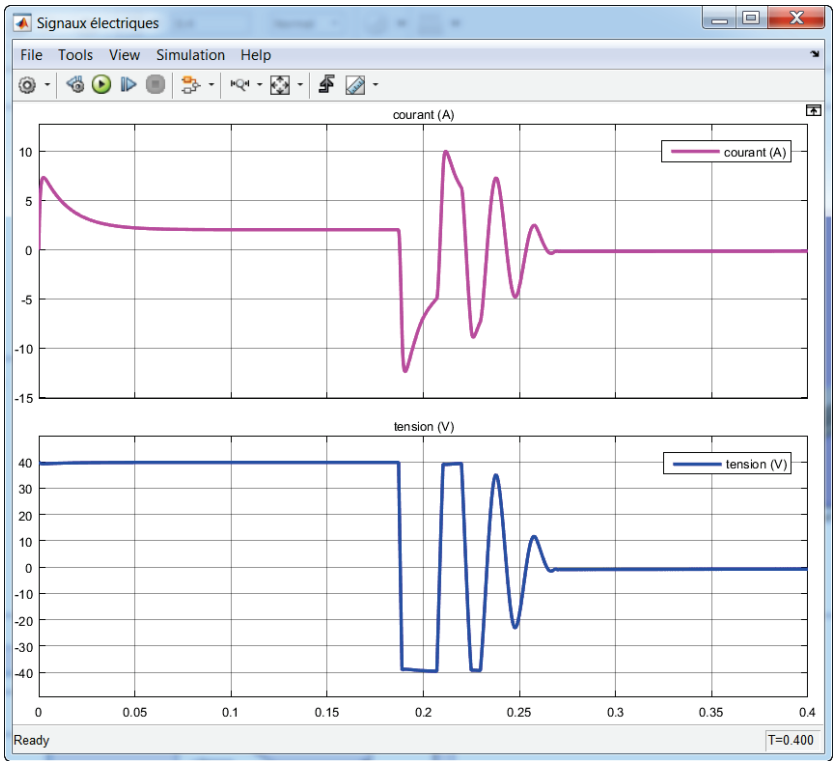


Figure 82 : visualisation de la tension de commande du moteur et du courant dans l'induit

La validation du modèle ne peut se faire que par comparaison avec les résultats expérimentaux. Le fichier **« ControlX_compar_reel_modele.slx »** contient une comparaison entre les performances issues de la simulation et les performances relevées sur le système réel.

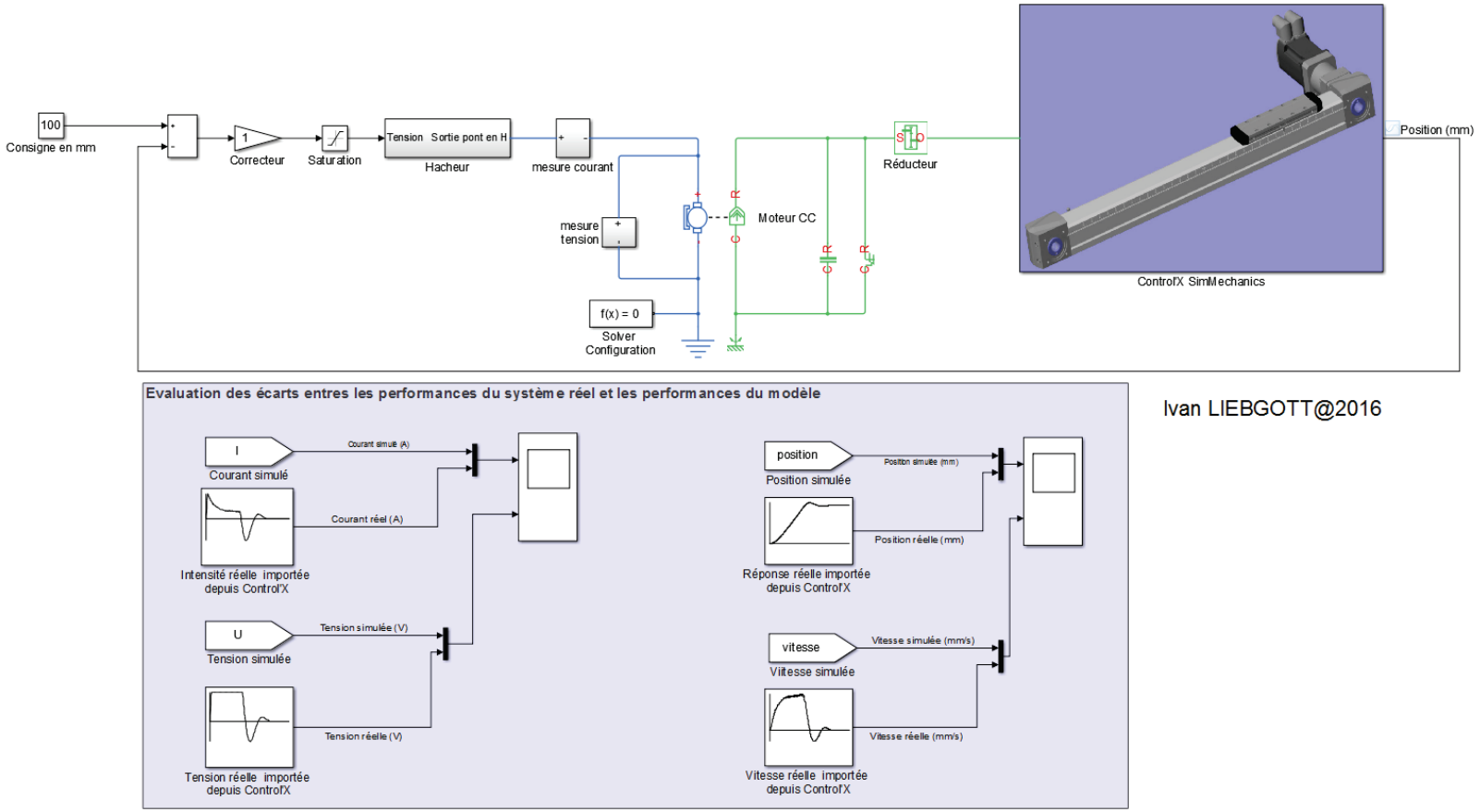


Figure 83 : évaluation des écart entre les performances du modèles et celles de l'axe réel

Lancer la simulation et observer les résultats obtenus.

Evaluation des écarts modèle/réel constatés sur la position et la vitesse :

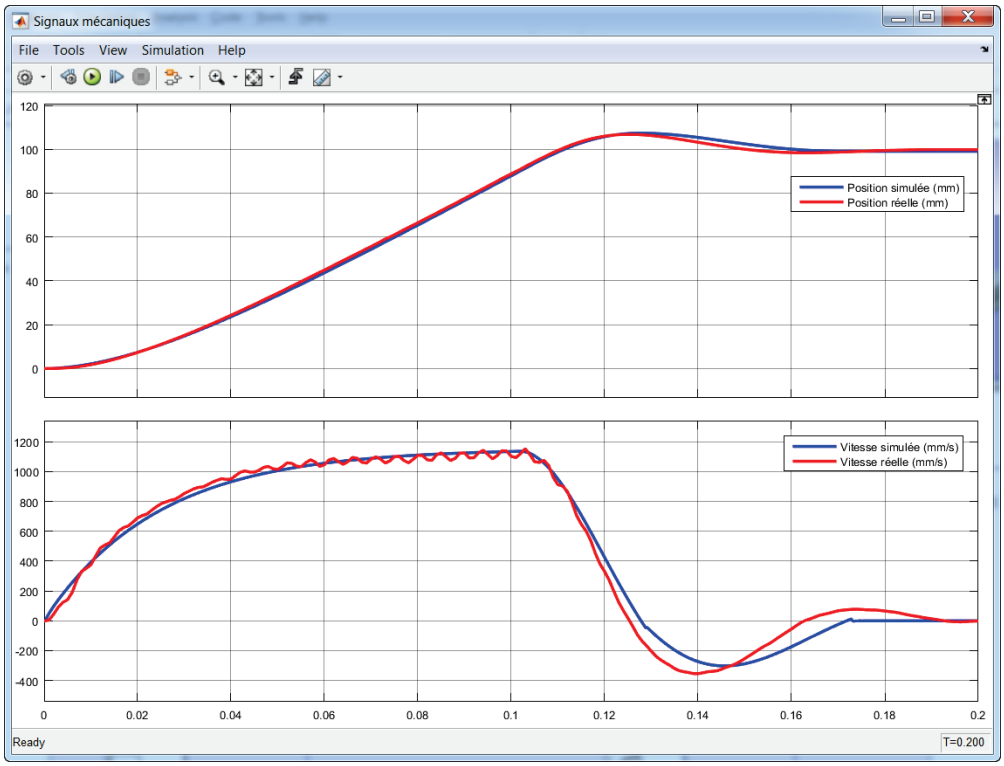


Figure 84 : évaluation des écarts réel/modèle pour la vitesse et la position de l'axe

Evaluation des écarts modèle/réel constatés sur la position et la vitesse :

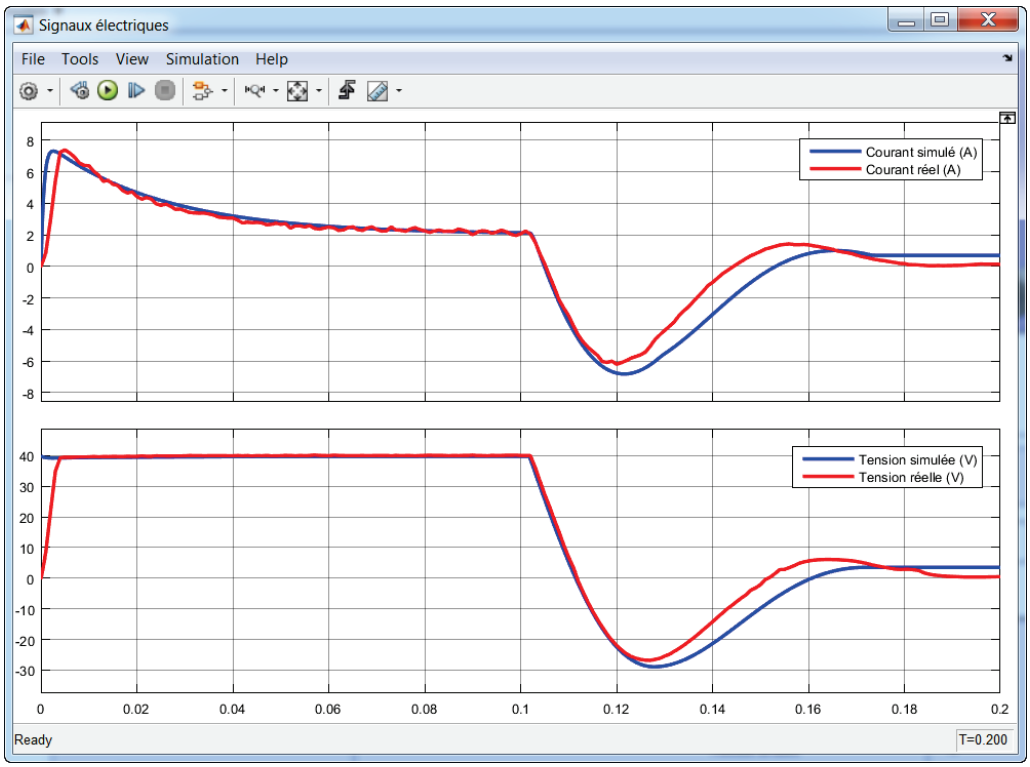


Figure 85 : évaluation des écarts réel/modèle pour le courant d'induit et la tension de commande

Les écarts constatés sont très faibles ce qui permet de mettre en évidence la pertinence de la démarche de modélisation multi-physique. Il sera possible de construire des modèles simples qui auront un comportement très proche des systèmes réels.

C. Comment faire un modèle multi-physique avec MATLAB-Simulink ?

Le modèle multi-physique fait intervenir de nombreux outils de modélisation. Pour construire un modèle multi-physique, il faudra dans un premier temps prendre en main les bases des outils suivants :

- MATLAB
- Simulink
- Simscape et ses libraires fondamentales (mécanique, électrique, hydraulique…)
- SimMechanics
- Stateflow

Il faudra également être capable d'utiliser les composants des bibliothèques spécifiques en fonction des besoins de la modélisation :

- SimHydraulics
- SimPowerSystems
- SimElectronics
- SimDrivelines

Cet ouvrage propose un apprentissage progressif de ces différents outils au travers d'exemples qui seront exploitables dans le cadre de la formation des ingénieurs. La première partie permet de se familiariser avec **Simscape** et la modélisation par composants (acausale). Ensuite, nous verrons les bases de la programmation en **MATLAB** et comment **MATLAB** et **Simulink** peuvent communiquer. La suite présentera une introduction à l'utilisation de **Stateflow** et de **SimMechanics**. La fin de l'ouvrage présente des outils d'analyse intéressants en lien direct avec la démarche de conception proposée dans l'ouvrage, comme la « **System Identification Toolbox** » ou « **Simulink cControl Design** » qui permet de mener les études de **contrôle commande** des systèmes asservis. Des méthodes de conception et de réglage des correcteurs sont présentées en utilisant des outils graphiques interactifs particulièrement adaptés à la formation et à la compréhension des phénomènes.

La maîtrise de ces différents outils permettra d'aborder une grande partie des problèmes de modélisation qui pourront être rencontrés dans le cadre de la formation des ingénieurs.

Chapitre 2 : Prise en main de Simscape

Ce chapitre a pour objectif de présenter les bases théoriques de la modélisation multi-physique acausale en utilisant **Simscape** et ses bibliothèques.

I. Introduction à la modélisation acausale avec Simscape

Au sein de l'environnement **MATLAB-Simulink**, l'outil **Simscape** propose une approche acausale de la modélisation et possède une vaste bibliothèque de composants élémentaires dont le comportement physique est déjà modélisé. La démarche de modélisation acausale consiste à réaliser un assemblage de composants pour modéliser le comportement d'un système. Le principe de calcul s'appuie sur un bilan de puissance à chaque nœud du modèle et ne repose pas sur le principe de causalité, d'où le nom de modélisation acausale.

Prenons l'exemple d'un circuit R-L.

A t=0 s, une source de tension alimente un circuit RL. Le modèle doit permettre d'analyser l'évolution du courant $i(t)$ qui traverse le circuit.

R=20 Ω
L=0.01 mH

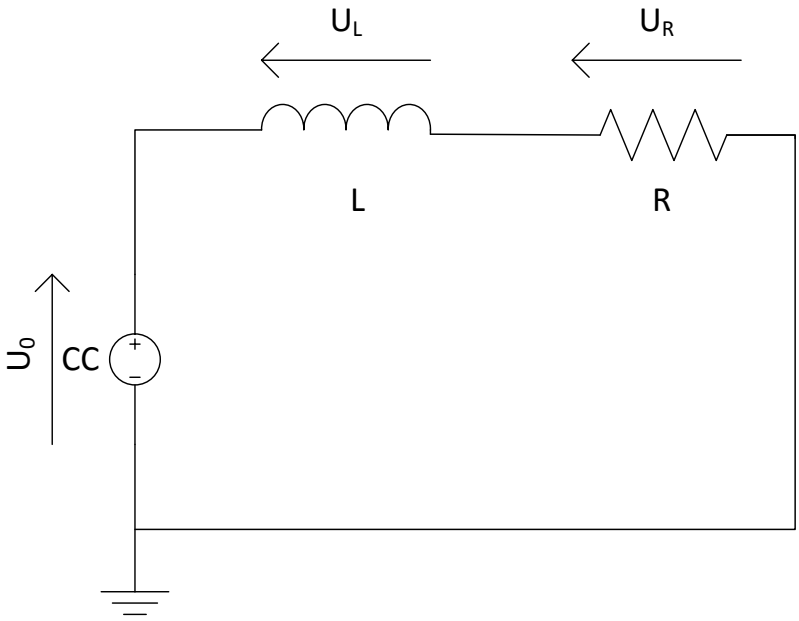


Figure 86 : circuit R-L

A. Choix des composants

La modélisation acausale du circuit RL est obtenue en utilisant les composants suivants :

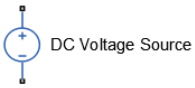
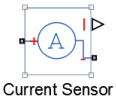
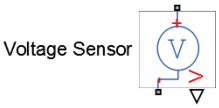
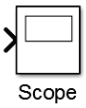
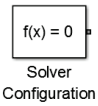
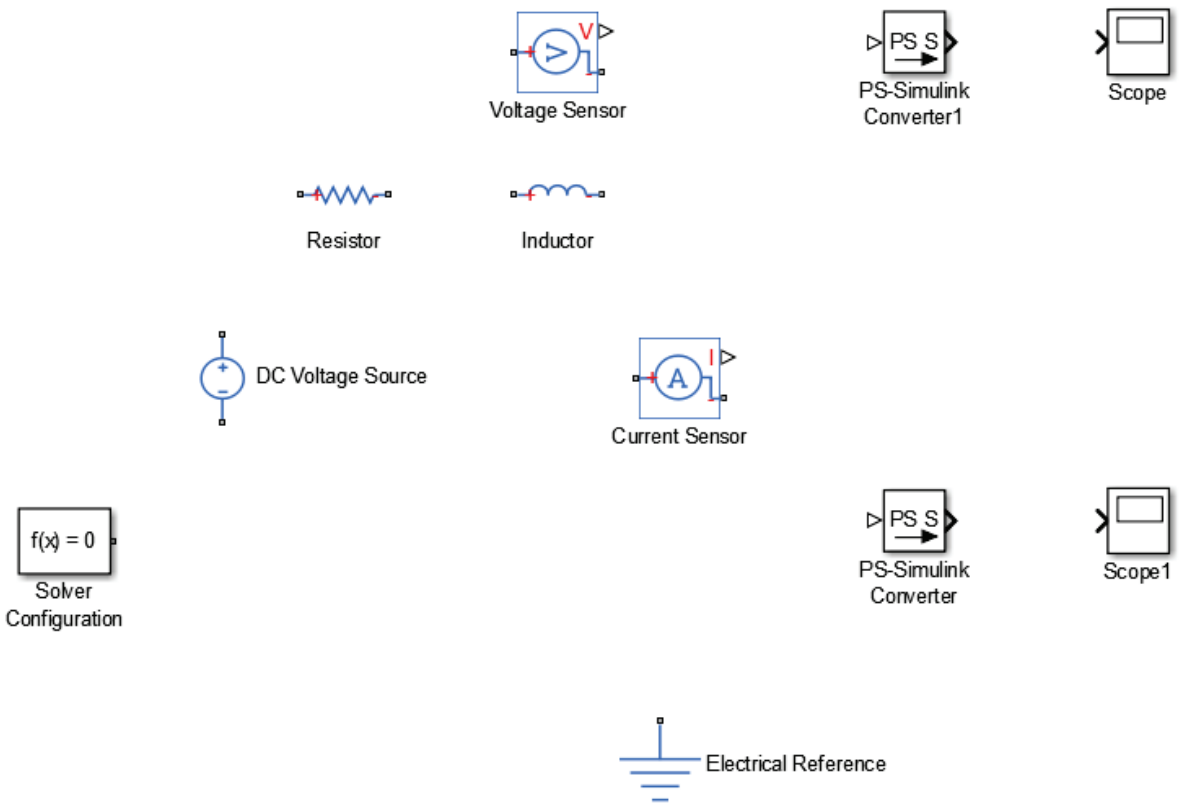
Fonction du composant	Représentation	Bibliothèque
Source de tension continue		Simscape/Fondation Library/Electrical/Electrical Sources
Résistance		Simscape/Fondation Library/Electrical/Electrical Element
Inductance		Simscape/Fondation Library/Electrical/Electrical Element
Référence électrique		Simscape/Fondation Library/Electrical/Electrical Element
Capteur de courant		Simscape/Fondation Library/Electrical/Electrical Sensors
Capteur de tension		Simscape/Fondation Library/Electrical/Electrical Sensors
Conversion d'un signal physique en signal Simulink		Simscape/Utilities
Moniteur		Simulink/Sinks
Solveur		Simscape/Utilities

Figure 87 : les composants nécessaires à la modélisation du circuit R-L avec Simscape

B. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, exécuter la commande **File/New/Model**  pour créer un nouveau fichier.

Glisser/Déposer les différents composants à partir des bibliothèques et les disposer dans la fenêtre de travail comme indiqué ci-dessous. Vous pouvez vous reporter à la Figure 88 pour prendre connaissance des commandes utiles à l'orientation des composants.



Commandes utiles		
Fonctions	Actions	Raccourcis clavier
Rotation d'un composant sens horaire	**Clic droit** sur le composant, **Rotate&Flip/Clockwise**	Ctrl+R
Rotation d'un composant sens antihoraire	**Clic droit** sur le composant, **Rotate&Flip/CounterClockwise**	Shift+Ctrl+R
Inversion gauche/droite d'un composant	**Clic droit** sur le composant, **Rotate&Flip/Flip Bloc/Left Right**	Ctrl+I
Inversion haut/bas d'un composant	**Clic droit** sur le composant, **Rotate&Flip/Flip Bloc/Up Down**	
Copier un composant	**Clic droit** sur le composant à dupliquer, puis **glisser/déposer** en gardant le bouton droit appuyé	
Supprimer un composant	**Clic droit** sur le composant puis sélectionnner **Delete**	Suppr

Figure 88 : commandes utiles

Relier les **composants** pour obtenir le modèle ci-dessous :

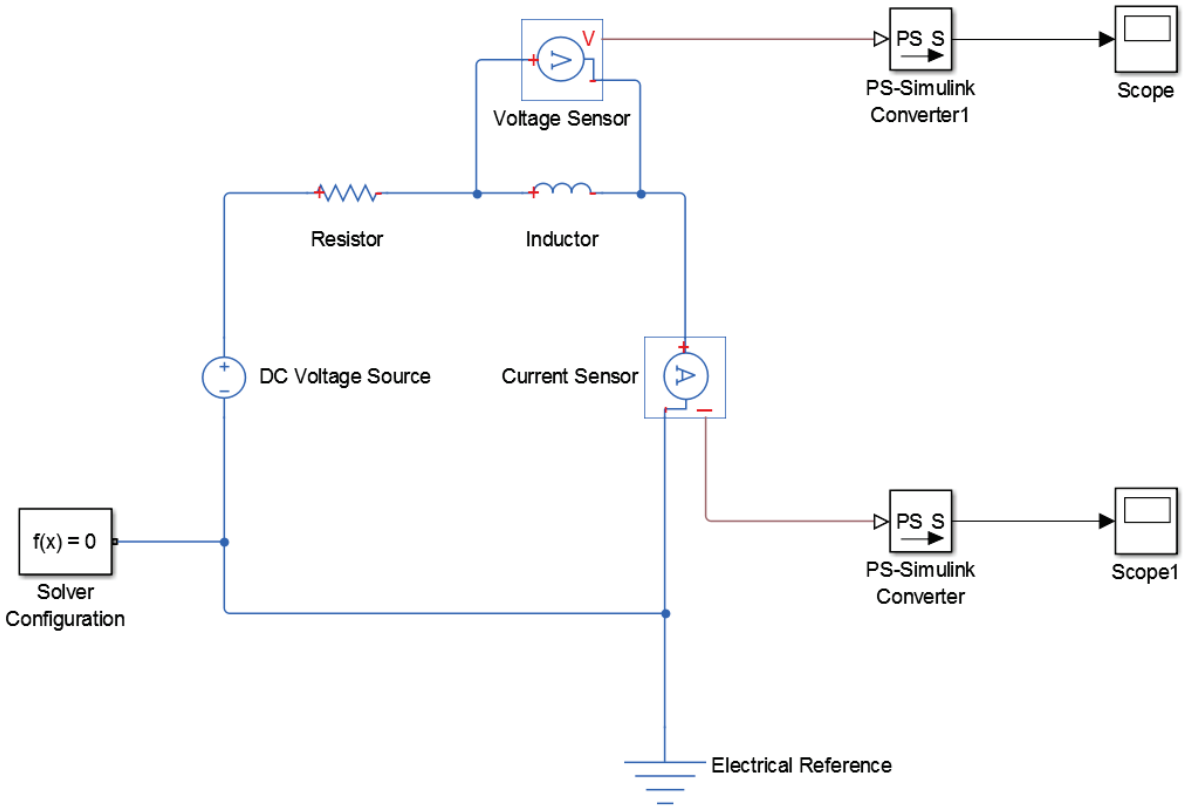


Figure 89 : Modèle Simscape du circuit R-L

Commandes utiles		
Fonctions	Actions	Raccourcis clavier
Relier deux composants	Clic gauche sur le port du premier composant puis se déplacer en maintenant le bouton gauche enfoncé jusqu'au port du second composants	

Figure 90 : commandes utiles

C. Les différents types de ports et de connexions

Avant de paramétrer les blocs et de lancer la simulation, analysons la nature des différentes connexions entre les blocs du modèle. Le modèle présenté utilise différents types de ports et de connexions. Les ports peuvent être classés en trois catégories (Figure 91).

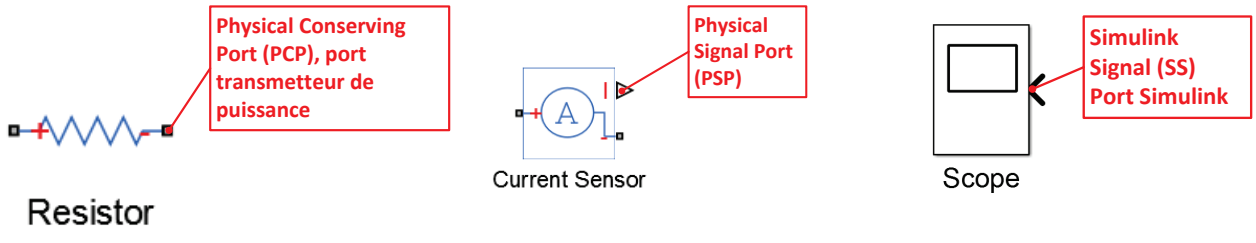


Figure 91 : les différents types de ports de Simscape

- Les ports de type « **Physical Conserving Port** » (**PCP**) qui transmettent la puissance entre deux composants d'un même domaine. Les connexions relatives à ces ports ne sont pas orientées et sont analogues aux liens qui relient les composants dans la réalité et se situent dans le domaine de la modélisation acausale. Dans le cas de notre modèle qui ne fait intervenir que le domaine électrique, ces connexions sont des fils électriques. Ils sont donc traversés par un courant et il est possible de mesurer une différence de potentiel entre deux points appartenant à ces connexions. La mesure du courant dans le circuit se fait par un capteur de courant placé en série dans le circuit. La mesure de la tension aux bornes de la bobine se fait par un capteur de tension placé en parallèle, aux bornes de la bobine.

- Des ports de type « **Physical Signal Port** » (**PSP**) qui transmettent des signaux physiques prélevés à l'aide de capteurs sur le modèle. Ces signaux sont orientés et sont l'image de la grandeur physique prélevée. Ces ports et ces connexions fonctionnent selon le principe de causalité.

- Des ports de type « **Simulink Signal » (SS)**, qui transmettent des signaux numériques orientés exploitables par les blocs de la bibliothèque Simulink. Ces ports et ces connexions fonctionnent selon le principe de causalité.

Le type de connexion peut être identifié par le format du port qui lui est associé.
La construction des modèles nécessite une parfaite compréhension de la nature des informations et des signaux qui parcourt les différentes connexions ainsi que l'identification de la nature des ports des blocs qui interviennent dans la modélisation. Des ports de natures différentes ne peuvent pas être reliés. De plus deux ports de types **Physical Conserving Port** (transmetteur de puissance) ne peuvent être reliés que s'ils appartiennent au même domaine physique. Afin d'apporter plus de lisibilité aux modèles réalisés avec Simscape, chaque domaine physique est représenté avec une couleur différente.

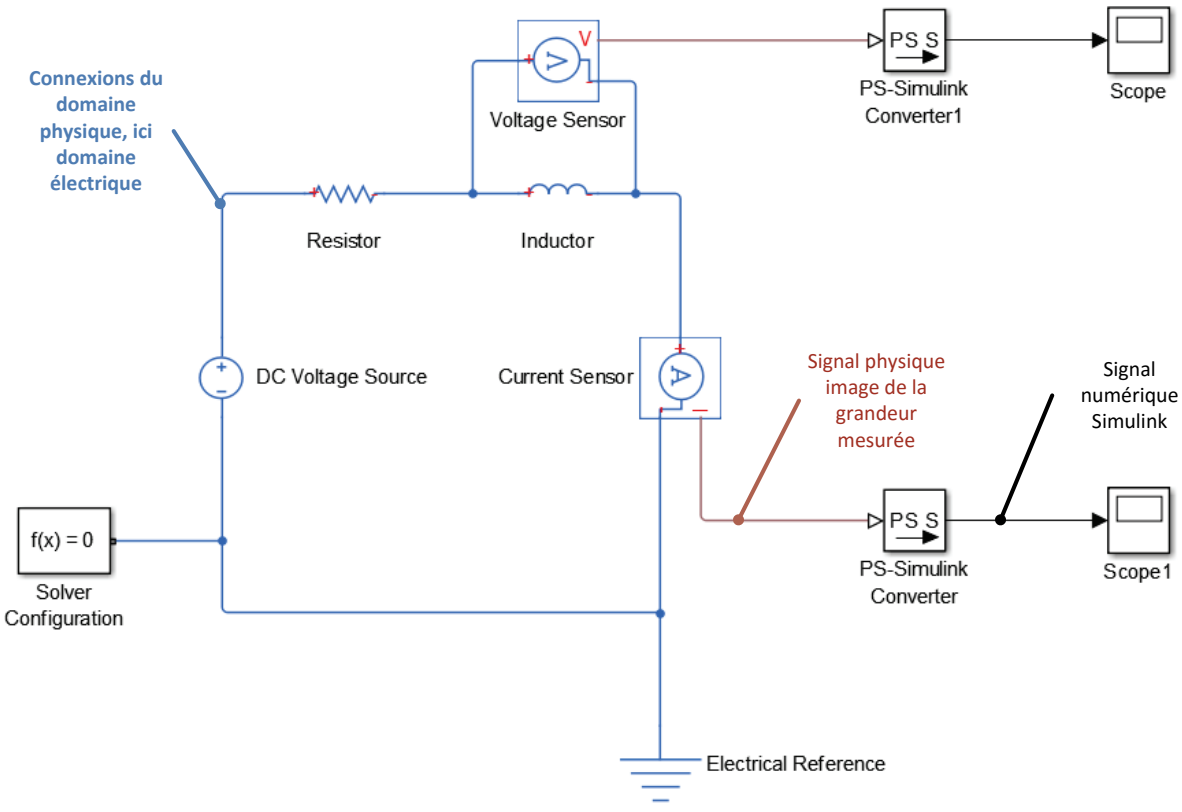
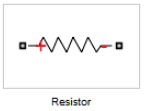


Figure 92 : identification des connexions d'un modèle Simscape

D. Paramétrage des composants

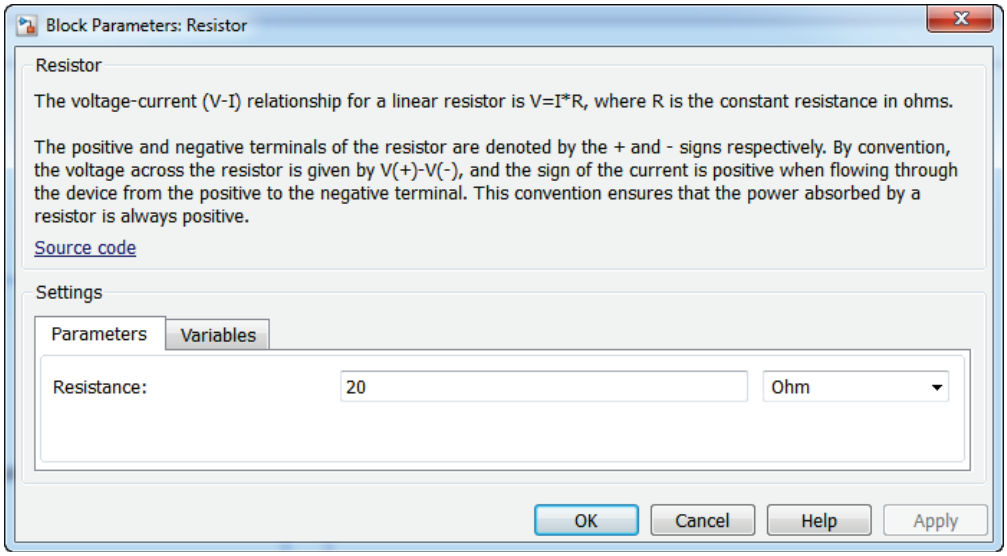
Le paramétrage des composants est accessible par un double-clic sur le bloc.

RESISTOR

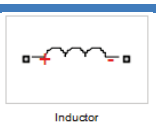


Simscape/Fondation Library/Electrical/Electrical Element

Le paramétrage de ce composant consiste à saisir la valeur de la résistance

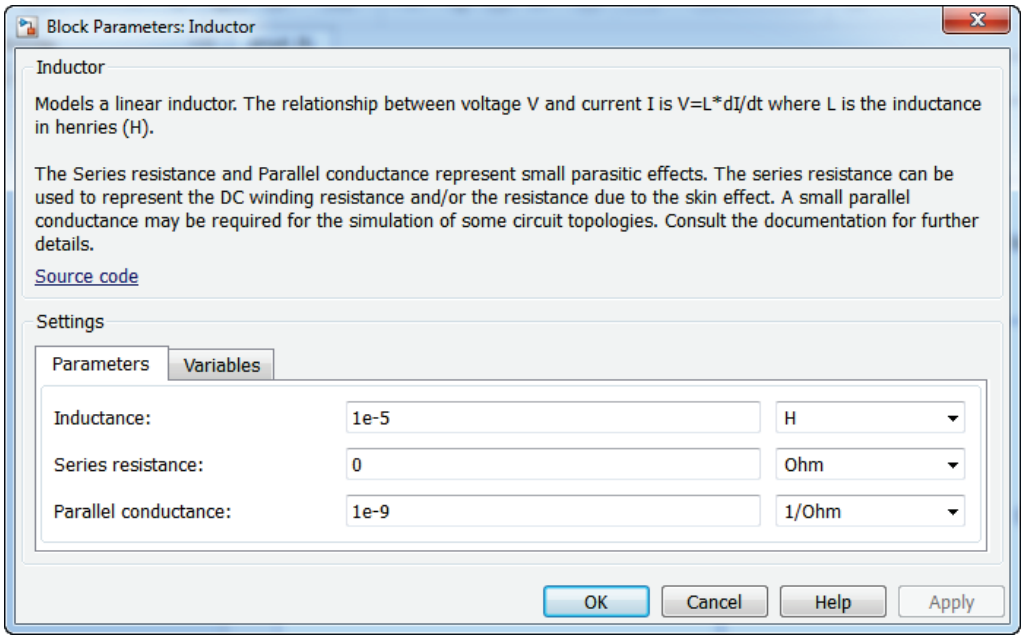


INDUCTOR



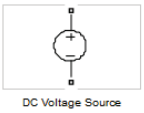
Simscape/Fondation Library/Electrical/Electrical Element

Le paramétrage de ce composant consiste à saisir la valeur de l'inductance.



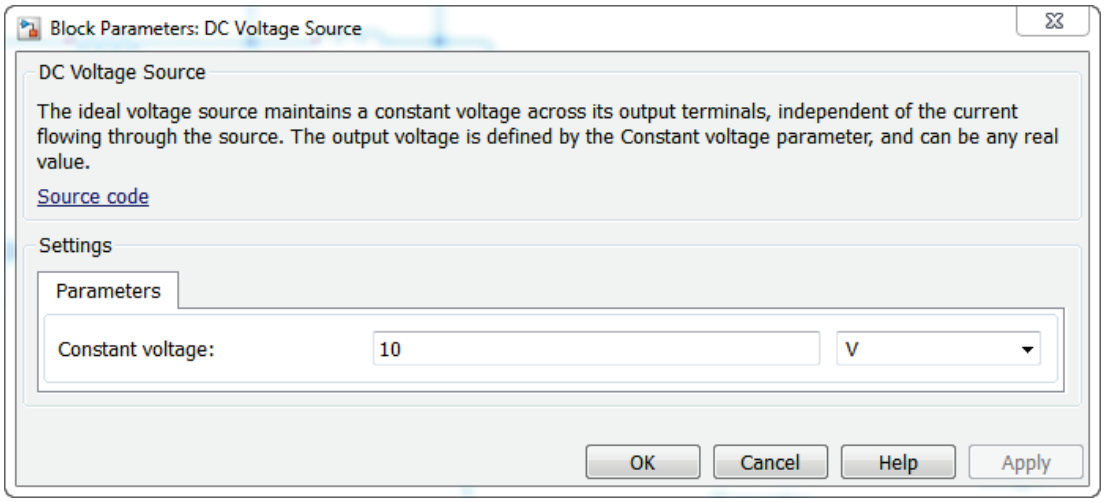
Paramétrage

DC VOLTAGE SOURCE



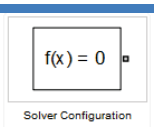
Simscape/Fondation Library/Electrical/Electrical Sources

Le paramétrage de ce composant consiste à saisir la valeur de la tension d'alimentation.



Paramétrage

SOLVER CONFIGURATION

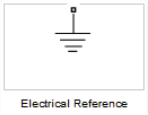


Simscape/Utilities

Aucune modification n'est à apporter au paramétrage de ce bloc pour ce modèle. Un bloc **Solver Configuration** doit obligatoirement être relié au modèle afin que le solveur puisse effectuer les calculs.

Paramétrage

ELECTRICAL REFERENCE



Simscape/Fondation Library/Electrical/Electrical Element

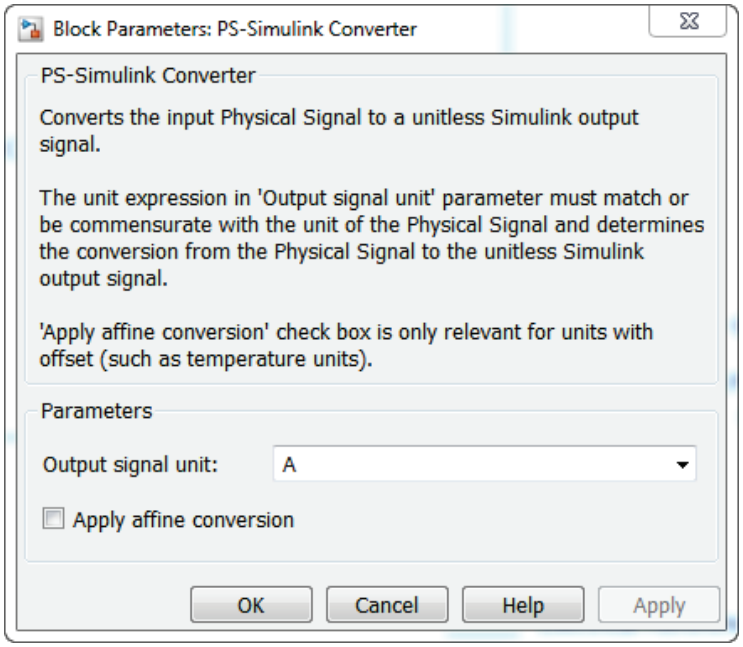
Aucun paramétrage n'est nécessaire pour ce composant. Un bloc référence du domaine physique utilisé doit obligatoirement être présent dans le modèle.

PS-SIMULINK CONVERTER 

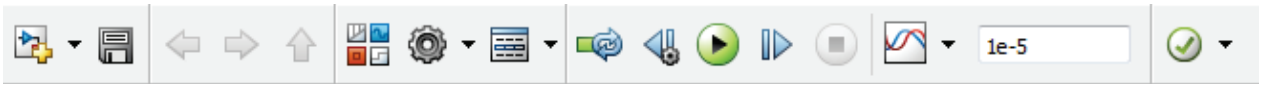
Simscape/Utilities

Ce composant permet de convertir un signal physique image d'une grandeur mesurée en signal **Simulink** afin de pouvoir être affiché dans un Scope. Il est souhaitable de préciser l'unité de la grandeur physique souhaitée pour l'affichage dans Simulink. Indiquer dans les deux blocs correspondants que l'on souhaite relever le courant en ampère (A) et la tension en volts (V). Le menu déroulant propose un choix d'unités, nous verrons qu'il est possible d'étendre les choix possibles d'unités. Si l'unité n'est pas spécifiée par l'utilisateur, l'unité SI du domaine sera choisie par défaut.

La fonctionnalité de ce bloc est très utile et permet de maitriser les unités des grandeurs physiques relevées. Simscape gère automatiquement la cohérence des unités pour la résolution du modèle et l'utilisateur n'a pas à s'en préoccuper.



E. Lancement de la simulation et analyse des résultats



Si nécessaire, le modèle complet est disponible dans le fichier ***circuit_RL.slx***

Spécifier un temps de simulation de **1e-5 seconde** et **lancer** la simulation en cliquant sur 

Double cliquer sur le Scope qui mesure le courant pour obtenir la variation du courant en fonction du temps.

Cliquer sur la mise à l'échelle automatique  si nécessaire pour obtenir la courbe suivante :

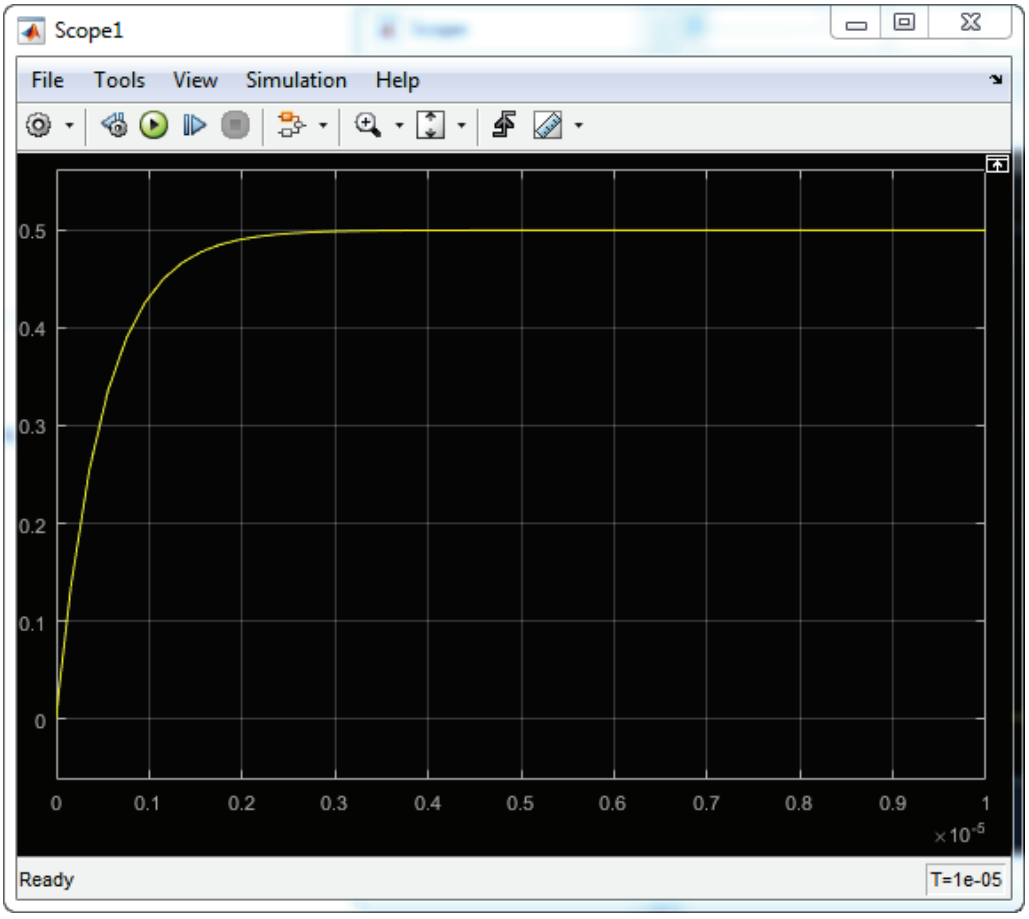


Figure 93 : évolution de l'intensité du courant dans le circuit RL

Se reporter à l'annexe « Paramétrage des scopes » pour la mise en forme des courbes.

On procède de manière identique pour visualiser l'évolution temporelle de la tension aux bornes de la bobine.

Cliquer sur la mise à l'échelle automatique  si nécessaire pour obtenir la courbe suivante :

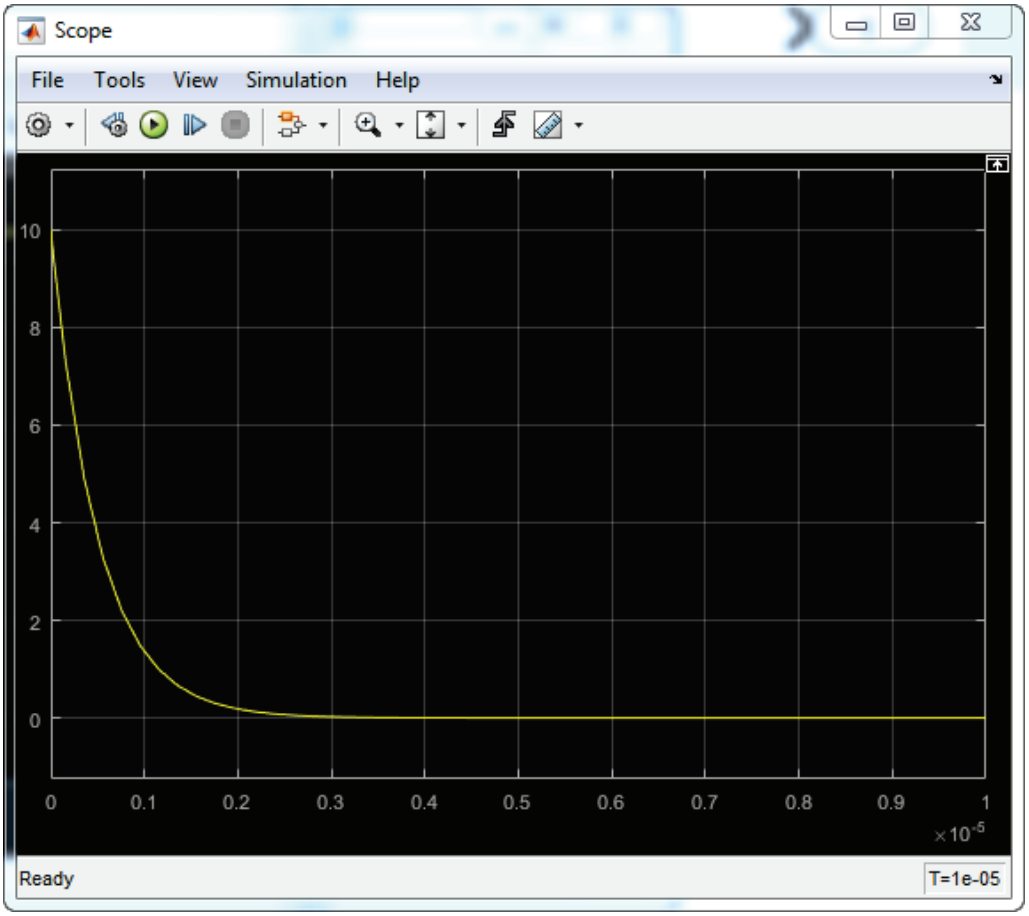


Figure 94 : évolution de la tension aux bornes de la bobine pour le circuit RL

Il est possible de modifier la valeur de l'inductance de la bobine pour voir son influence sur l'établissement du courant dans le circuit en modifiant le paramètre du bloc **inductor** et en relançant la simulation.

On voit que cette approche de modélisation est très intuitive et permet de construire et de modifier très rapidement un modèle. Le modèle ainsi construit possède une structure très proche de celle que l'on peut observer sur le système réel ce qui lui donne une excellente lisibilité.

II. Comparaison avec l'approche causale

A. Equation de comportement du système

Les tensions dans le circuit sont liées par la relation temporelle suivante :

$$U(t) = U_R(t) + U_L(t) = R\,i(t) + L\frac{d\,i(t)}{dt}$$

En appliquant la transformée de Laplace à cette équation en admettant les conditions de Heaviside :

$$U(p) = (R + Lp)\,I(p)$$
$$I(p) = \frac{1}{(R + Lp)}U(p)$$

L'évolution du courant sera donnée par la réponse temporelle à un échelon de tension d'une fonction de transfert du premier ordre.

B. Choix des composants

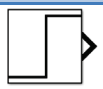
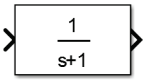
Fonction du composant	Représentation	Bibliothèque
Source échelon	 Step	Simulink/Sources
Fonction de transfert	 1 / s+1 Transfer Fcn	Simulink/Continuous
Moniteur	 Scope	Simulink/Sinks

Figure 95 : les composants nécessaires à la modélisation du circuit R-L avec Simulink

C. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, sélectionner la commande **File/New/Model** pour créer un nouveau fichier.

Glisser/Déposer les différents composants à partir des bibliothèques et les disposer dans la fenêtre de travail comme indiqué ci-dessous.

Relier les composants. Vous pouvez vous reportez à la Figure 97 pour prendre connaissances des commandes utiles pour relier et ordonner les blocs dans **Simulink**.

Le modèle obtenu est le suivant :

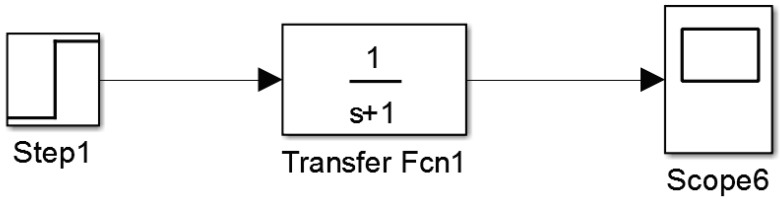


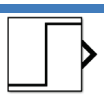
Figure 96 : modèle du circuit R-L réalisé avec Simulink

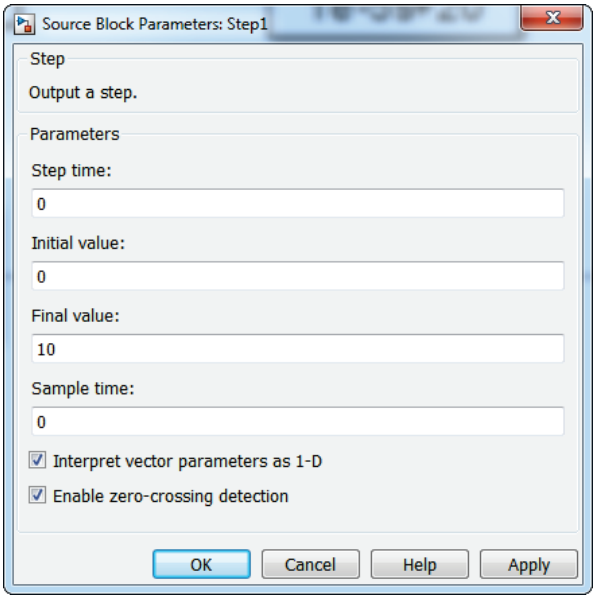
Commandes utiles	
Fonctions	Actions
Connecter deux composants	Sélectionner le premier composant. En maintenant la touche contrôle enfoncée, sélectionner le deuxième bloc. Les blocs sont reliés automatiquement
Aligner les composants	Sélectionner les composants concernés. Puis Clic droit, **Arrange/ Align Middle**
Ordonner le positionnement des composants	Sélectionner les composants concernés. Puis Clic droit, **Arrange** puis choisir l'ordonnancement désiré

Figure 97 : Commandes utiles

D. Paramétrage des composants

Paramétrage

STEP  Simulink/Sources

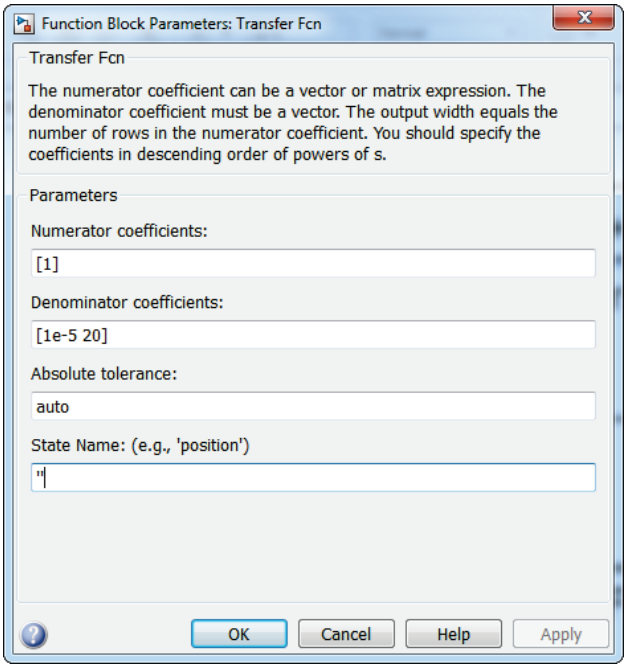


Step time : instant à partir duquel l'échelon passe à la valeur initiale, ici à t=0 s.
Initial value : valeur initiale de l'échelon, ici 0
Final value : valeur finale de l'échelon qui correspond ici à la tension de 10 V que l'on souhaite imposer.
Sample time : toujours laisser cette valeur à 0 pour l'étude des systèmes à temps continu.

TRANSFERT FUNCTION 

Simulink/Continuous

La fonction de transfert à saisir est $H(p)=\frac{1}{R+Lp}=\frac{1}{1e^{-5}p+20}$ avec R=20 Ω et L=0.01 mH



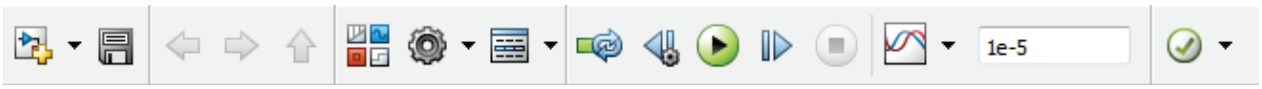
Le paramétrage des fonctions de transfert se fait par la saisie des coefficients du numérateur et du dénominateur en commençant par le coefficient du monôme de plus haut degré.

Exemple :

$G(p)=\frac{3p+1}{12p^2+5p+4}$

Numerator coefficients : [3 1]
Denominator coefficients : [12 5 4]

E. Lancement de la simulation et analyse des résultats



Spécifier un temps de simulation de **1e-5 seconde** et **lancer** la simulation en cliquant sur 

Faire un double-clic sur le Scope qui mesure le courant pour obtenir la variation du courant en fonction du temps.

Cliquer sur la mise à l'échelle automatique pour obtenir la courbe 

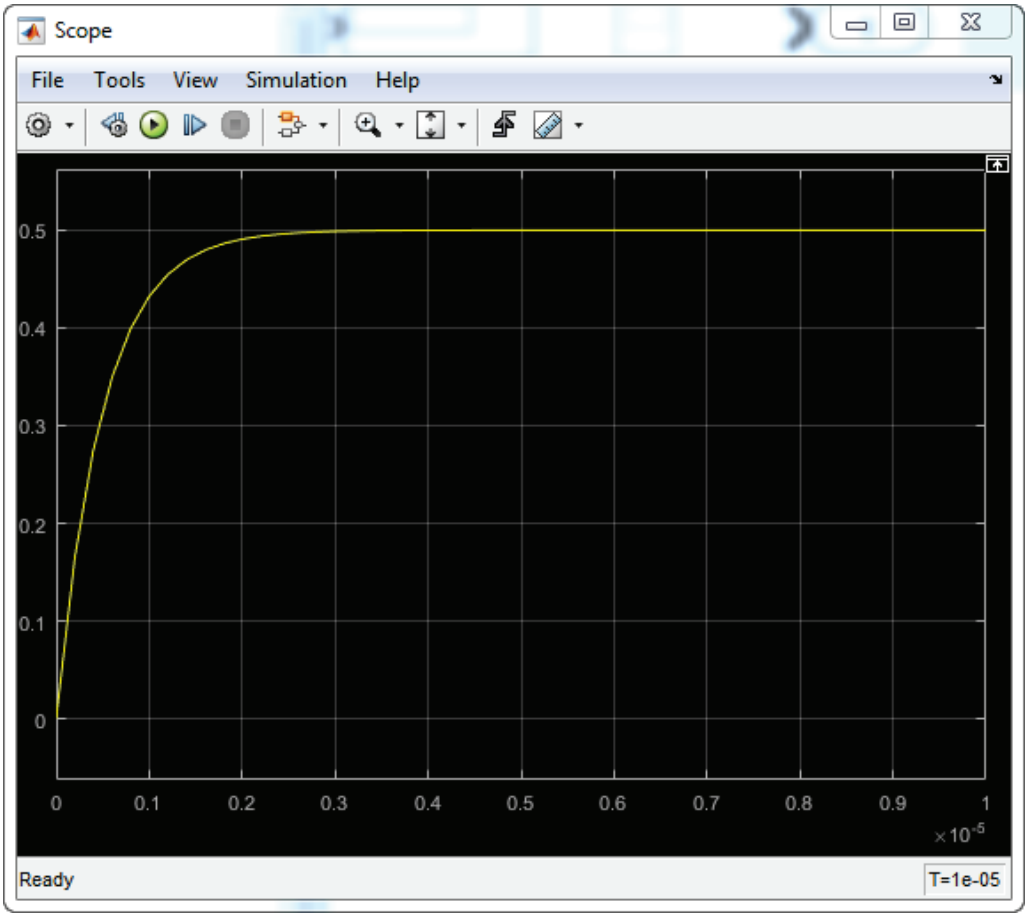


Figure 98 : évolution de l'intensité du courant dans le circuit RL

La courbe donnant l'évolution du courant dans le circuit est la même que pour le modèle acausale (heureusement !). Nous pouvons remarquer que l'évolution de la tension aux bornes de la bobine n'est pas directement accessible, une modification du modèle permettrait cependant d'y avoir accès.

F. Avantage et inconvénients des approches causale et acausale

	Modélisation acausale	Modélisation causale
Avantages	• Très proche de la structure du système réel • Grande facilité à prendre en compte des phénomènes physiques complexes • Facile à modifier, facile à lire • Plus rapide que les autres méthodes de modélisation • Les conditions initiales peuvent être choisies quelconques • Toutes les grandeurs physiques sont accessibles et mesurables au sein du modèle • Pas d'équation à écrire • Les connexions entre les composants transmettent un niveau d'information plus riche • Aucun outil mathématique nécessaire • Reconnaissance immédiate du domaine • Méthode de résolution numérique plus adaptée à la résolution des modèles complexes	• Parfaite maitrise du niveau de modélisation • Structure du modèle plus adaptée à la mise en place d'une démarche de contrôle commande • Maitrise de l'introduction des non linéarités
Inconvénients	• Peu de maitrise concernant le niveau de modélisation des composants utilisés(1) • Les modèles obtenus sont fortement non linéaires ce qui rend la démarche de contrôle commande délicate(2) • Nécessite des connaissances de base dans les domaines physiques abordés	• Les grandeurs physiques sont considérées nulles à t=0 (conditions de Heaviside) • Difficile à lire et à modifier • Nécessite une parfaite connaissance théorique des phénomènes étudiés • Impose l'utilisation d'outils mathématiques avancés (transformée de Laplace) • La modélisation des systèmes complexes entraîne de nombreux bouclages dans le modèle ce qui pose des problèmes de résolution numérique pour les solveurs.

Figure 99 : avantages et inconvénients des approches causales et acausales

(1) Le niveau de modélisation du composant est imposé par le logiciel. La compréhension et l'interprétation des équations utilisées nécessitent une bonne connaissance du domaine pour être interprétées et permettre un paramétrage correct du composant. Il est important de noter que **Simscape** et ses bibliothèques proposent souvent plusieurs niveaux de modélisation pour le même type de composant.

(2) La modélisation des composants intègre par défaut de nombreuses non-linéarités qui reflètent le comportement réel du composant (seuil, saturation, hystérésis…).

Ces deux approches de modélisation coexistent dans la construction d'un modèle multi-physique. Le choix de l'une ou de l'autre des approches se fera en fonction de l'utilisation que l'on souhaite faire du modèle et du niveau de connaissance du domaine physique abordé.

III. Les fondamentaux de la modélisation avec Simscape

A. Notions de domaines physiques

Au sein de l'environnement **Simscape**, plusieurs domaines physiques ou technologiques coexistent. Il est très important d'analyser et d'identifier la nature des flux d'énergies qui transitent au travers des ports d'un composant et d'identifier le ou les domaine(s) physique(s) qui interviennent dans la modélisation.
Le nombre de ports de type **Physical Conserving Port (PCP)** que possède un composant est déterminé par le nombre de flux d'énergie qu'il échange avec l'extérieur. Il peut échanger des flux d'énergie dans un même domaine physique ou assurer une conversion de puissance d'un domaine physique à un autre. La conversion de puissance peut se faire avec un rendement de 100% ou tenir compte de paramètres caractérisant les pertes. Cela dépendra du niveau de modélisation du composant et des phénomènes pris en compte. Le tableau de la Figure 100 présente quelques exemples de composants et détaille les types de ports associés.

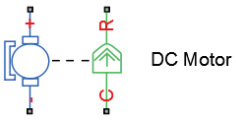
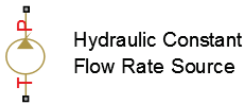
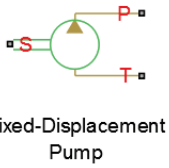
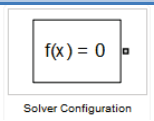
Fonction du composant	Représentation	Types de ports
Résistance		Ce composant possède 2 ports de type **PCP** du domaine **électrique** Paramètre : résistance
Réducteur		Ce composant possède 2 ports de type **PCP** du domaine **mécanique de rotation.** Paramètre : rapport de réduction
Conversion mécanique rotation/translation		Ce composant possède 2 ports de type **PCP**. Port A du domaine **mécanique de rotation** et Port B du domaine **mécanique de translation**. Paramètre : rayon de la roue
Conversion de puissance électrique en puissance mécanique		Ce composant possède 4 ports de type **PCP**. 2 port (+ et -) du domaine **électrique** et 2 ports (C et R) du domaine **mécanique de rotation**. Paramètre : k_t en N.m/A
Source de débit hydraulique		Ce composant possède 2 ports de type **PCP** du domaine hydraulique. Paramètre : débit
Conversion de puissance mécanique en puissance hydraulique		Ce composant possède 3 ports de type **PCP**. 1 port (S) du domaine **mécanique** et 2 ports (T et R) du domaine **hydraulique.** Le port mécanique (S) transmet un couple qui fait tourner le barillet de la pompe générant un transfert de fluide du port T au port P. Paramètre : cylindrée de la pompe (et d'autre paramètres annexes caractérisant le rendement de la conversion de puissance)

Figure 100 : les ports PCP des composants Simscape

B. Les blocs importants de Simscape

Les modèles conçus avec **Simscape** doivent obligatoirement contenir certains blocs indispensables au déroulement de la simulation.

Le bloc Solveur :

Désignation	Représentation	Bibliothèque
Solveur		Simscape/Utilities

Un bloc Solveur doit être relié à une partie du modèle **Simscape**. Si ce bloc n'est pas présent un message d'erreur d'affiche au lancement de la simulation.

Les blocs « Référence » des domaines physiques utilisés :

Si un domaine physique est utilisé par un modèle **Simscape**, le modèle doit obligatoirement contenir une référence du domaine utilisé. Si un modèle exploite plusieurs domaines physiques, une référence de chaque domaine doit être utilisée. Si une référence est absente, un message d'erreur s'affiche au lancement de la simulation.

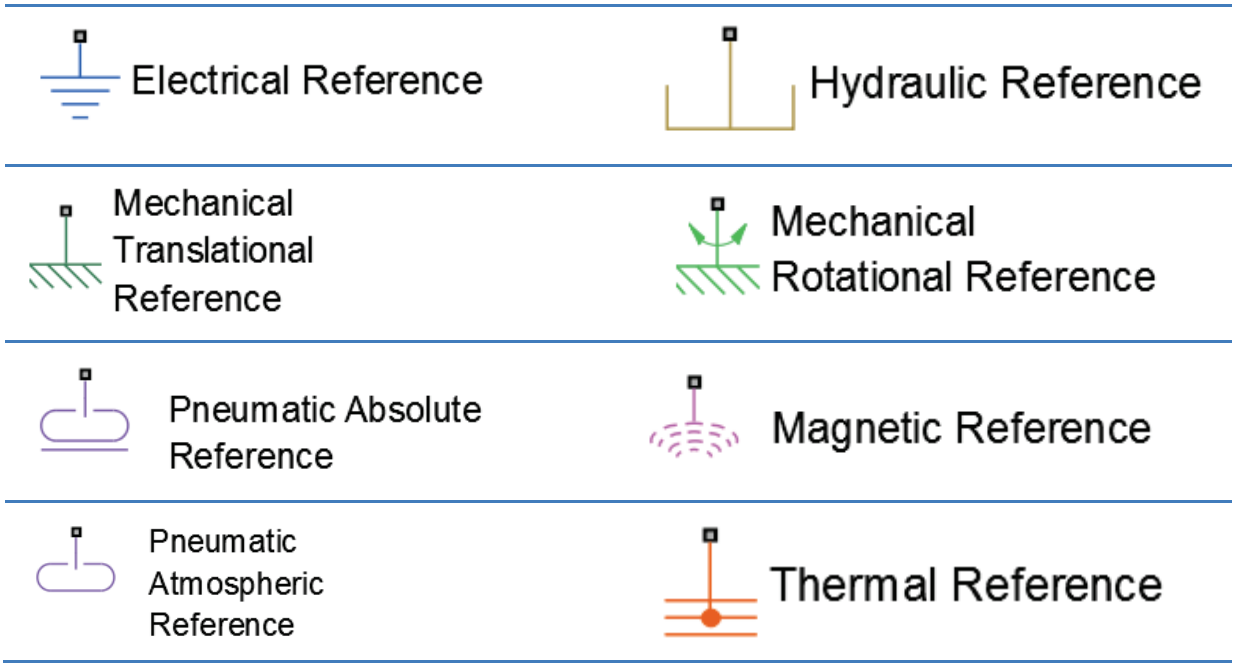


Figure 101 : les références des domaines physiques de Simscape

C. Variables de type « Across » et « Throught » et positionnement des capteurs

Dans **Simscape**, les flux d'énergie sont définis à l'aide de deux variables, l'une de type « **Across** » (parallèle) et l'une de type « **Through** » (série). Ces variables sont appelées variables conjuguées et leur produit caractérise la puissance dans le domaine physique considéré.

- Les variables de type **Through** sont mesurées à l'aide d'un capteur placé en **série** dans le modèle (courant électrique, débit hydraulique…).
- Les variables de type **Across** sont mesurées à l'aide d'un capteur placé en **parallèle** entre deux points du modèle (différence de potentiel entre deux points d'un circuit électrique, différence de pression entre deux points d'un circuit hydraulique…).

Il est donc fondamental de connaître la nature (Across ou Through) de la grandeur physique que l'on souhaite mesurer sur une connexion du modèle dans la mesure où cette information déterminera l'implantation du capteur. Une erreur pourra entrainer un message d'erreur lors du lancement de la simulation ou un comportement erroné du modèle.

Domaine Physique	Variable(s) Across	Variable(s) Through
Electrique	Tension	Courant
Hydraulique	Pression	Débit
Mécanique de translation	Vitesse linéaire	Force
Mécanique de rotation	Vitesse angulaire	Moment
Pneumatique	Pression et temperature	Débit massique et flux d'entropie
Thermique	Température	Flux thermique et flux d'entropie

Figure 102 : les variables Across et through de Simscape

D. L'orientation des composants

Dans **Simscape**, les composants ont une convention d'orientation. Le sens positif d'un composant est toujours défini d'un port vers l'autre. Cette information est donnée dans la fenêtre de paramétrage du composant.

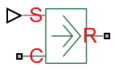
Il faudra prendre en compte la polarité des composants dans les cas suivants :

- Utilisation de composants actifs
- Implantation des capteurs
- Utilisation de composants dont la dynamique est orientée

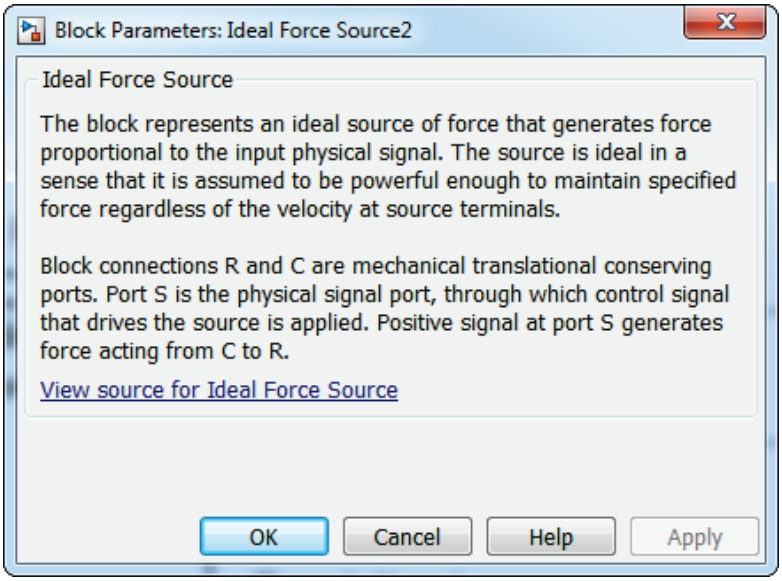
1. Utilisation de composants actifs

Lors de l'utilisation des composants actifs (sources de vitesse, de force, de débit, de pression…), il faut tenir compte de l'orientation du composant.

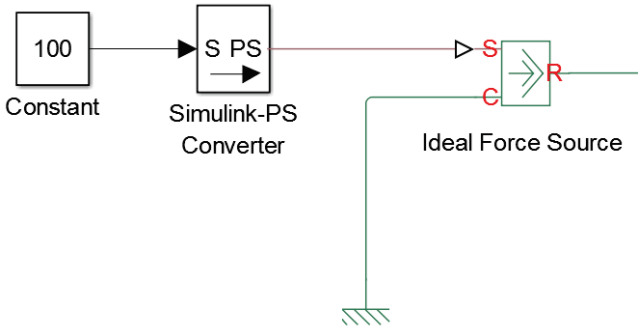
Exemple de l'utilisation d'une source d'effort.

Désignation	Représentation	Bibliothèque
Source d'effort	 Ideal Force Source	Simscape/Fondation Library/Mechanical/Mechanical Sources

Fenêtre de paramétrage:



Source d'effort



Ce modèle permet de simuler une source de force de 100 N.
Le paramétrage du bloc nous indique qu'une valeur positive du signal qui arrive sur le port S génère un effort qui s'exerce positivement de C vers R.

Sur le modèle de la Figure 103, une force de 100 N, dirigée de C vers R va agir sur l'ensemble ressort amortisseur.

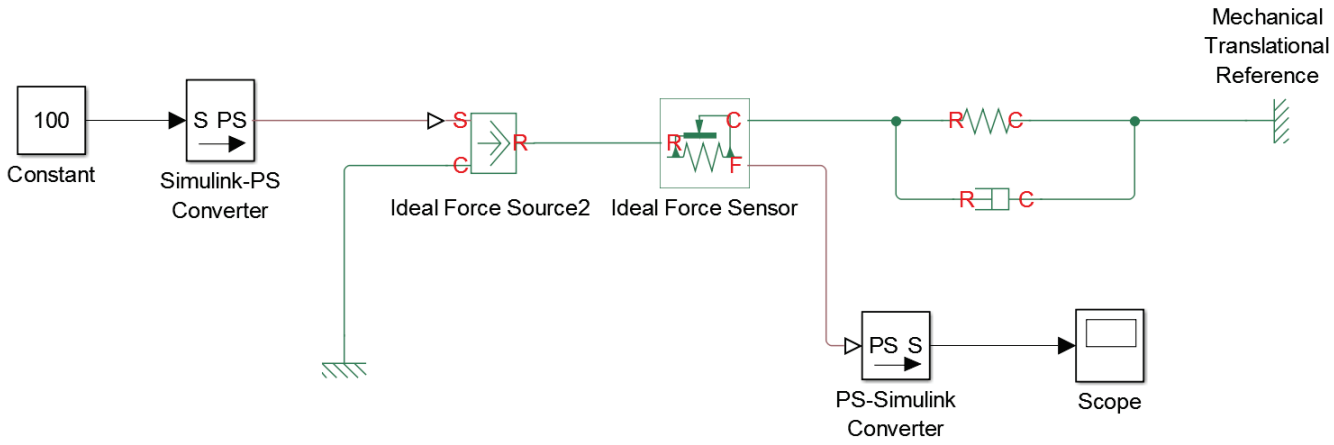


Figure 103 : source de force orientée positivement

Sur le modèle de la Figure 104, les ports C et R ont été inversés, et la force va s'appliquer à l'opposé de la configuration de la Figure 103.

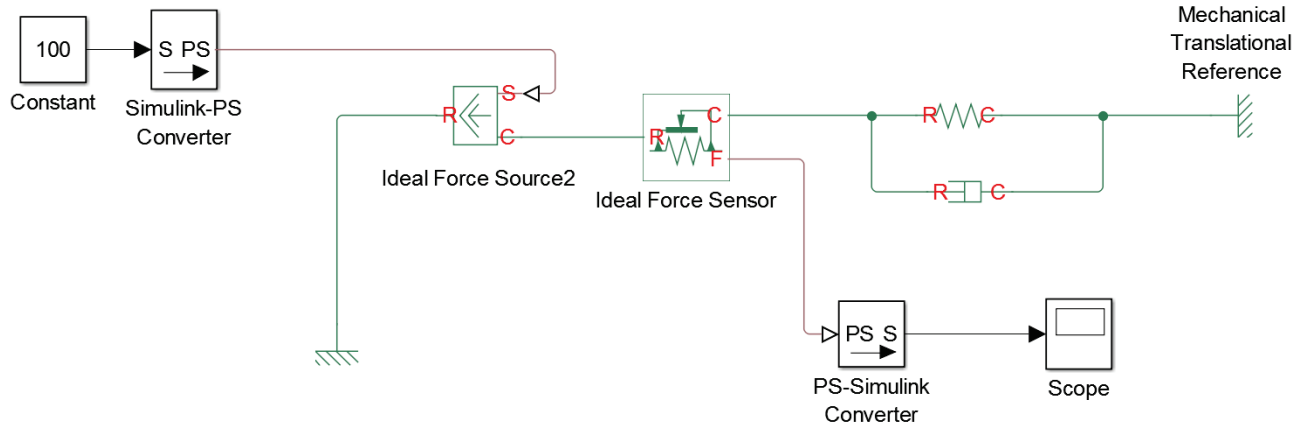


Figure 104 : source de force orientée négativement

Ce raisonnement peut être généralisé à tout type de source d'énergie.

Ouvrir le fichier **« orientation_source_force.slx ».**

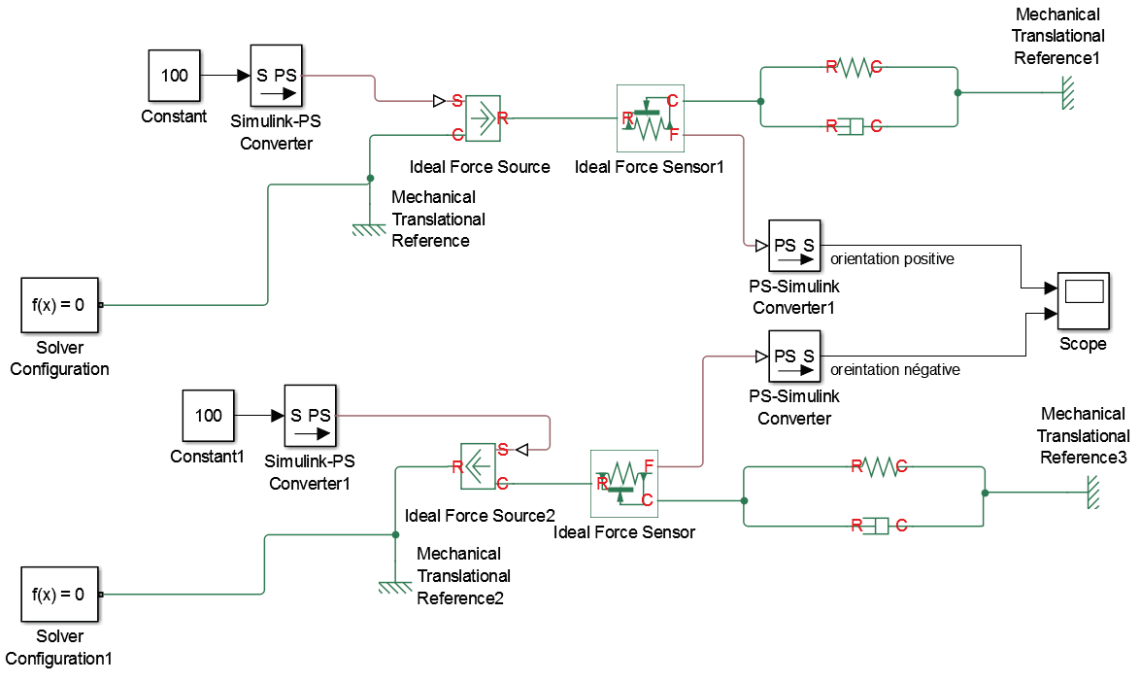


Figure 105 : orientation d'une source de force

Ce fichier contient deux modélisations identiques d'une source de force qui agit sur un ensemble ressort+amortisseur. Un capteur d'effort placé en série mesure l'effort (variable de type Through) exercé par la source d'effort sur l'ensemble ressort+amortisseur. Dans le modèle correspondant au bas de la Figure 105, l'orientation de la source de force a été inversée.

Lancer la simulation et observer la mesure de l'effort à l'aide du scope pour les deux modélisations. Cliquer sur la mise à l'échelle automatique du scope si nécessaire 

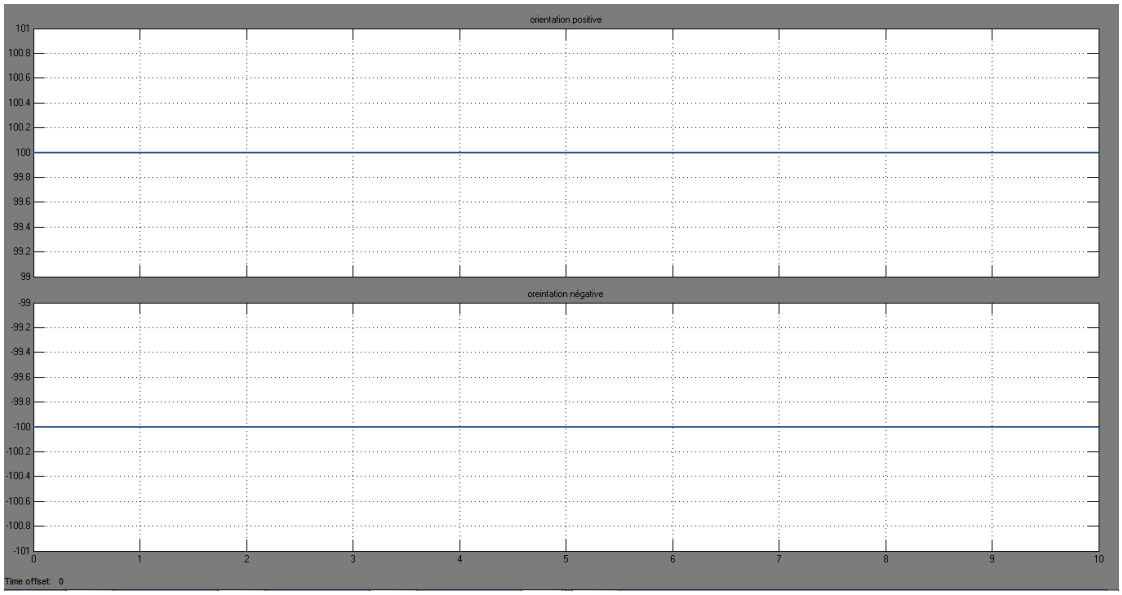


Figure 106 : visualisation de l'influence de l'orientation de la source de force

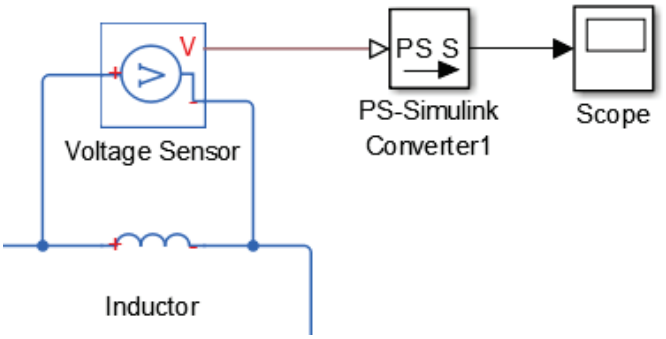
On observe que l'orientation de la source a une influence directe sur le sens de l'effort exercé sur l'ensemble ressort+amortisseur.

2. Implantation et orientation des capteurs

Lors de l'implantation d'un capteur. Le signe de la mesure dépend de l'orientation du capteur.

Exemple de mesure de grandeurs électriques :

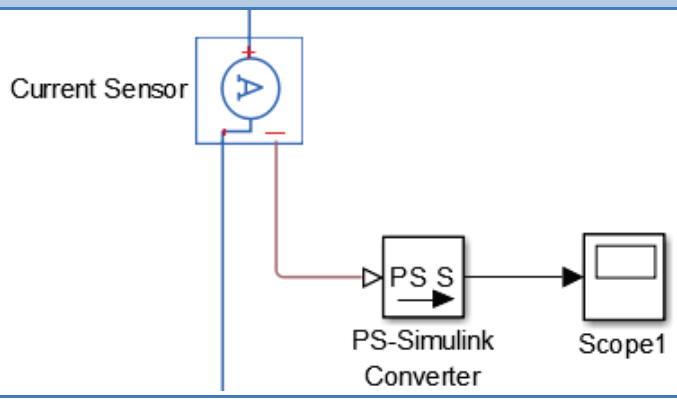
Mesure d'une tension (variable Across)



Le capteur est placé en parallèle.

Le résultat de la mesure est $V^+ - V^-$ et sera positif si le potentiel relié à la borne + du capteur est supérieur au potentiel relié à la borne – du capteur. Si on inverse les bornes du capteur, le résultat de la mesure sera de signe contraire.

Mesure d'un courant (variable Through)



Le capteur est placé en série.
Le résultat de la mesure sera positif si le courant circule de la borne + vers la borne – du capteur. Si on inverse les bornes du capteur, le résultat de la mesure sera de signe contraire.

Figure 107 : exemple de mesure de grandeurs électriques

Ce raisonnement peut être généralisé à tout type de capteur et tout type de variable de type « Across » et « Through ».

Ouvrir le fichier « **implantation_des_capteurs.slx** »

Ce fichier contient la modélisation d'un circuit RL. La tension aux bornes de la bobine et le courant qui traverse le circuit sont mesurés à l'aide de deux capteurs dont la polarité a été inversée.

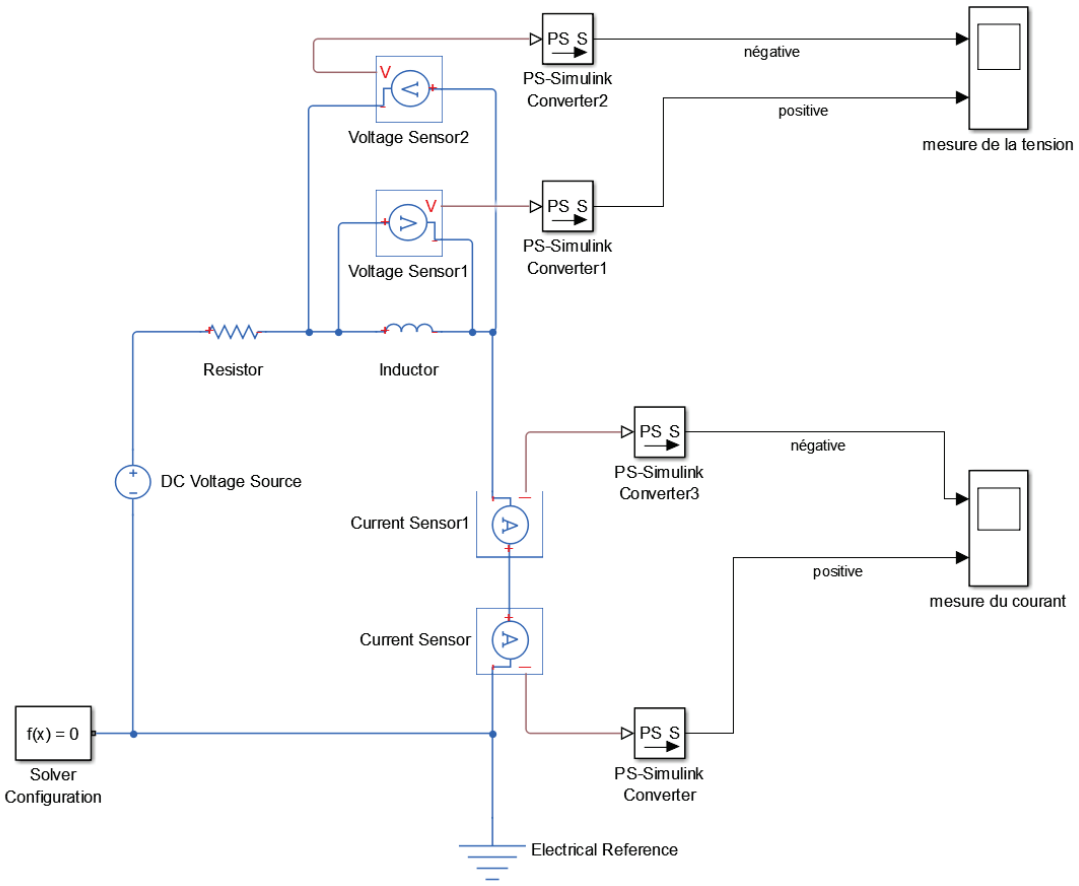


Figure 108 : implantation des capteurs

Lancer la simulation et observer à l'aide des scopes l'influence de l'orientation des capteurs sur le signe de la mesure obtenu.

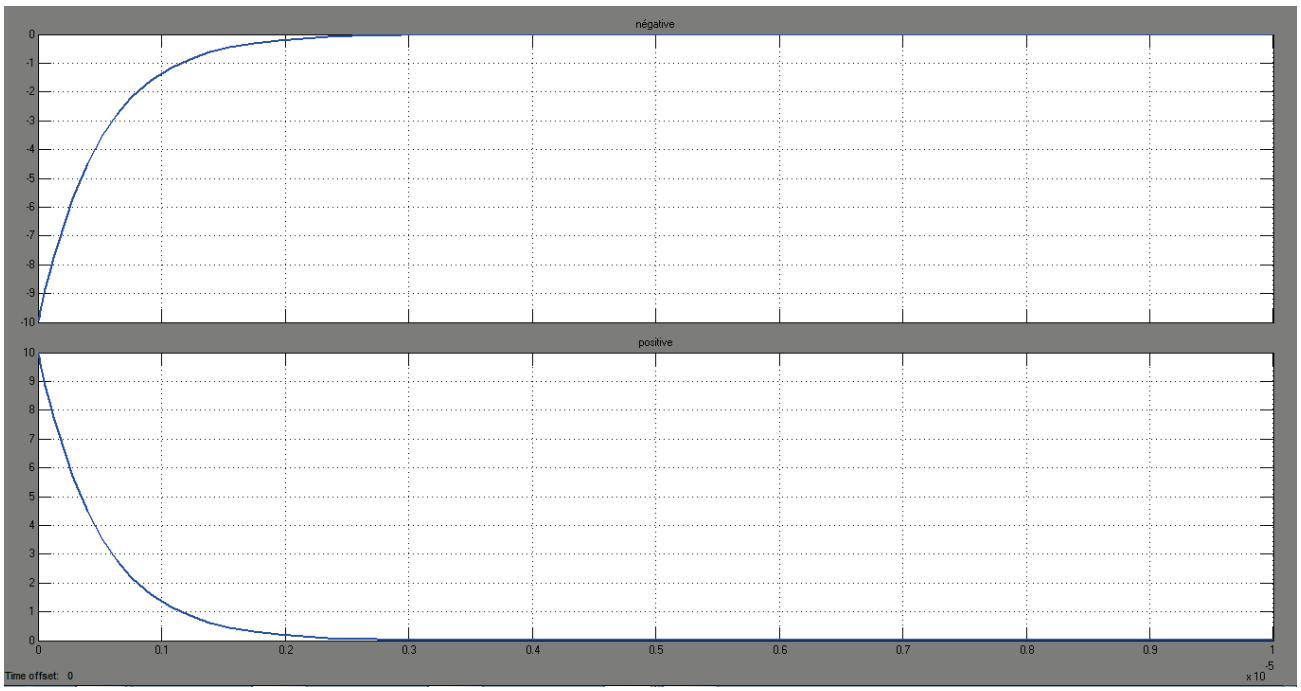


Figure 109 : visualisation de la tension aux bornes de la bobine pour les deux orientations des capteurs

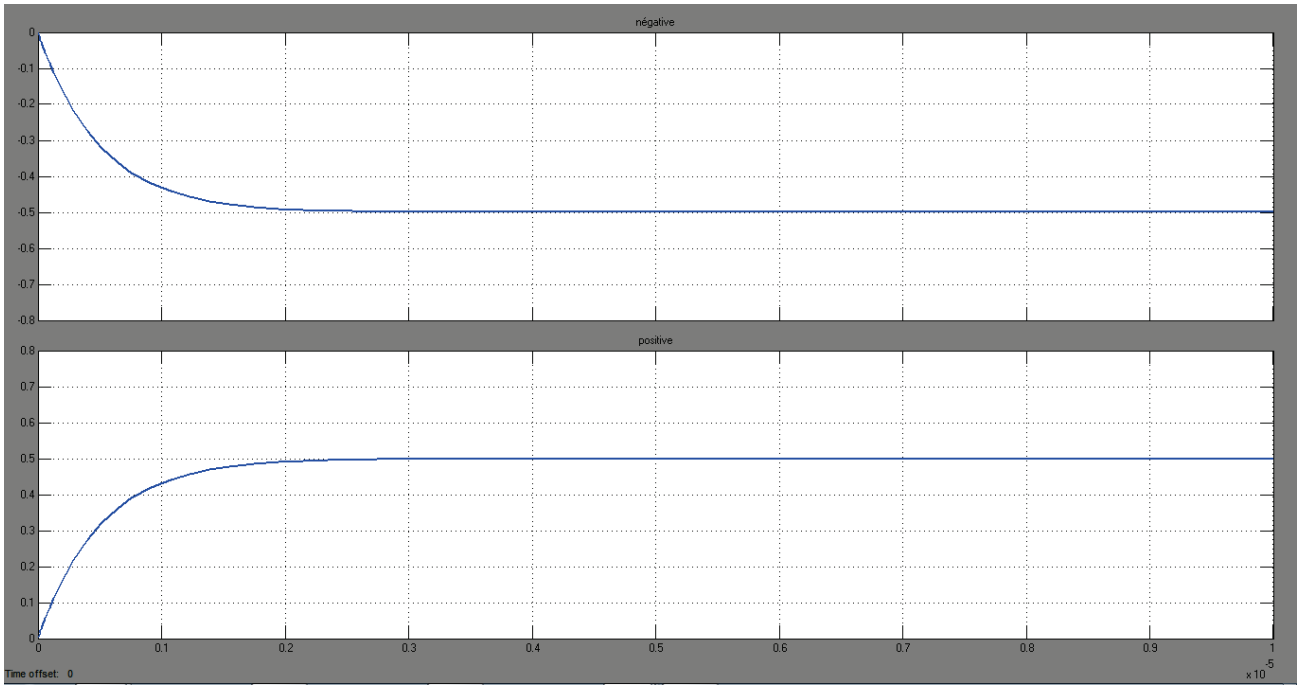


Figure 110 : visualisation de l'intensité du courant dans le circuit en fonction des deux orientations des capteurs

Les résultats obtenus mettent en évidence l'influence de l'orientation des capteurs sur le signe de la mesure obtenue.

3. Utilisation de composants dont la dynamique est orientée

Si on utilise un composant dont la dynamique est orientée (diode, réducteur de vitesse…), le composant doit être orienté de la même manière que sur le système réel afin d'obtenir un résultat cohérent avec le système que l'on souhaite modéliser. Si l'orientation est mauvaise, aucun message d'erreur ne permettra de détecter cette erreur et le système modélisé ne sera plus conforme au système réel.

4. Utilisation de composants passifs

Lors de l'utilisation de composants passifs (résistance, ressort, amortisseurs, canalisation…), dont la dynamique n'est pas orientée, l'orientation du composant n'a pas d'influence sur le comportement du modèle et sur l'observation des variables « Across » et « Through » relevées à l'aide des capteurs.

5. Choix du solveur

Le choix du solveur permet de définir les méthodes de résolution numériques qui seront mise en œuvre pour que le logiciel puisse mener la résolution numérique du modèle dans les meilleures conditions. Lorsque les modèles **Simscape** deviennent complexes, il est préférable de choisir le solveur le mieux adapté à la résolution de ce type de problème.

Pour cela dans la fenêtre de votre modèle **cliquer** sur **Simulation/Model Configuration Parameters** ou sur l'icône  pour afficher la fenêtre de paramétrage du solveur (Figure 111).

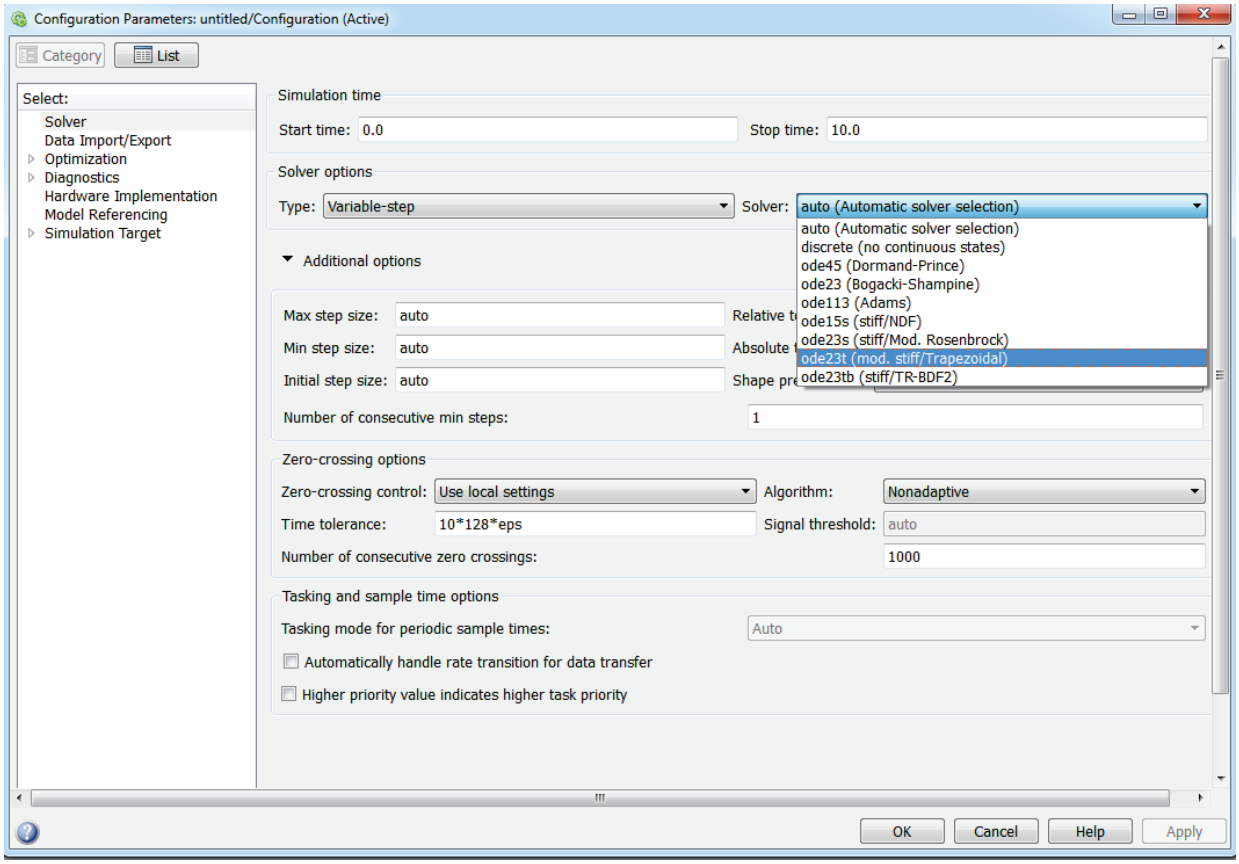


Figure 111 : fenêtre de paramétrage du solveur

Par défaut le choix du solveur est automatique. Dans le cas général cette option permet de choisir le solveur le plus adapté.

Les deux solveurs **ode15s** ou **ode23t** sont les mieux adaptés à la résolution des modèles **Simscape et** multiphysiques. Il est fortement conseillé de ne travailler qu'avec ces deux solveurs pour les modèles **Simscape**. Dans le cas général, si le choix automatique du solveur ne donne pas de résultats satisfaisants, on pourra choisir le solveur **ode15s**. Si ce solveur ne parvient toujours pas à résoudre le modèle ou nécessite un temps de calcul trop long, on utilisera **ode23t**.

Il est également possible dans cette fenêtre de spécifier un pas de calcul minimum et/ou maximum pour choisir le nombre de points et la résolution du calcul. On renseignera alors les champs **Max step size** et **Min step size**.

6. Les problèmes que peut rencontrer le solveur

Lors du lancement de la simulation, le solveur va résoudre numériquement le modèle afin de déterminer l'évolution en fonction du temps de toutes les variables. Ce processus mathématique très complexe sera géré automatiquement par le logiciel. Cependant des problèmes peuvent entraîner un arrêt du processus de calcul entraînant un message d'erreur indiquant que le solveur ne parvient pas au terme de la résolution.
Les solveurs utilisés par MATLAB étant extrêmement robustes, dans la majorité des cas les problèmes viennent d'une erreur dans la modélisation. Il faudra reprendre le modèle et corriger les éventuelles erreurs. Lors de la réalisation d'un modèle complexe, il faudra simuler séparément chaque partie du modèle pour identifier les parties du modèle qui provoquent les erreurs.

Voici quelques pistes pour permettre de résoudre les problèmes les plus courants :

- Il manque le bloc « Solveur » ;
- Il manque une référence d'un domaine physique utilisé ;
- Il manque des composants ou les composants ne sont pas paramétrés avec des valeurs cohérentes. Les grandeurs calculées par le solveur n'ont plus de sens physique (vitesse infinie, pression infinie, température inférieure au zéro absolu…). Dans ce cas le solveur arrêtera le calcul en indiquant les problèmes qu'il rencontre. Il faudra revoir le paramétrage ou le positionnement des composants.

IV. Exemples de modélisation multi-domaine

Cette partie propose, au travers d'exemples, un aperçu des domaines physiques les plus courants qui peuvent être abordés avec **Simscape**.
Chaque exemple propose de construire un modèle et d'exploiter les résultats de la simulation. Au fur et à mesure de la construction des modèles, des notions importantes sont introduites permettant d'acquérir les compétences nécessaires à la conception de modèle multi-physiques avec **Simscape**. Il est donc conseillé de parcourir l'ensemble des exemples proposés dans cette partie.

A. Domaine électromécanique – Axe linéaire

Le système modélisé est un axe linéaire. L'actionneur est un motoréducteur à courant continu commandé par une source de tension variable. Le mouvement de rotation est transformé en mouvement de translation par un système poulies/courroie. L'objectif est d'évaluer les réponses en vitesse et en position de l'axe pour estimer le temps de réponse et la rapidité ou tout autre type de performances. Il sera également possible de mesurer des paramètres électriques comme le courant moteur.

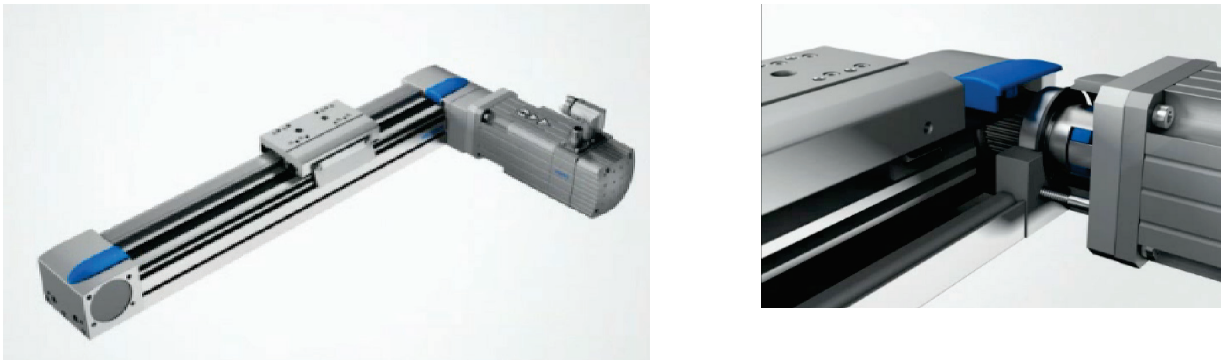


Figure 112 : photo de l'axe linéaire

Le système étudié fait intervenir 3 domaines physiques :

- Domaine électrique
- Domaine mécanique de rotation
- Domaine mécanique de translation

Certains composants appartiennent à un domaine physique particulier et d'autres composants permettront de faire la conversion de l'un à l'autre des domaines physiques. Ce sera le cas du composant « **DC Motor** » qui fera la conversion du domaine électrique au domaine mécanique de rotation et du composant « **Wheel and Axle** » qui fera la conversion du domaine mécanique de rotation au domaine mécanique de translation.

1. Choix des composants

Seuls sont référencés dans le tableau de la Figure 113, les composants qui n'ont pas encore été utilisés auparavant dans le document.

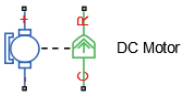
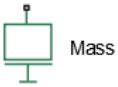
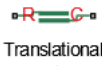
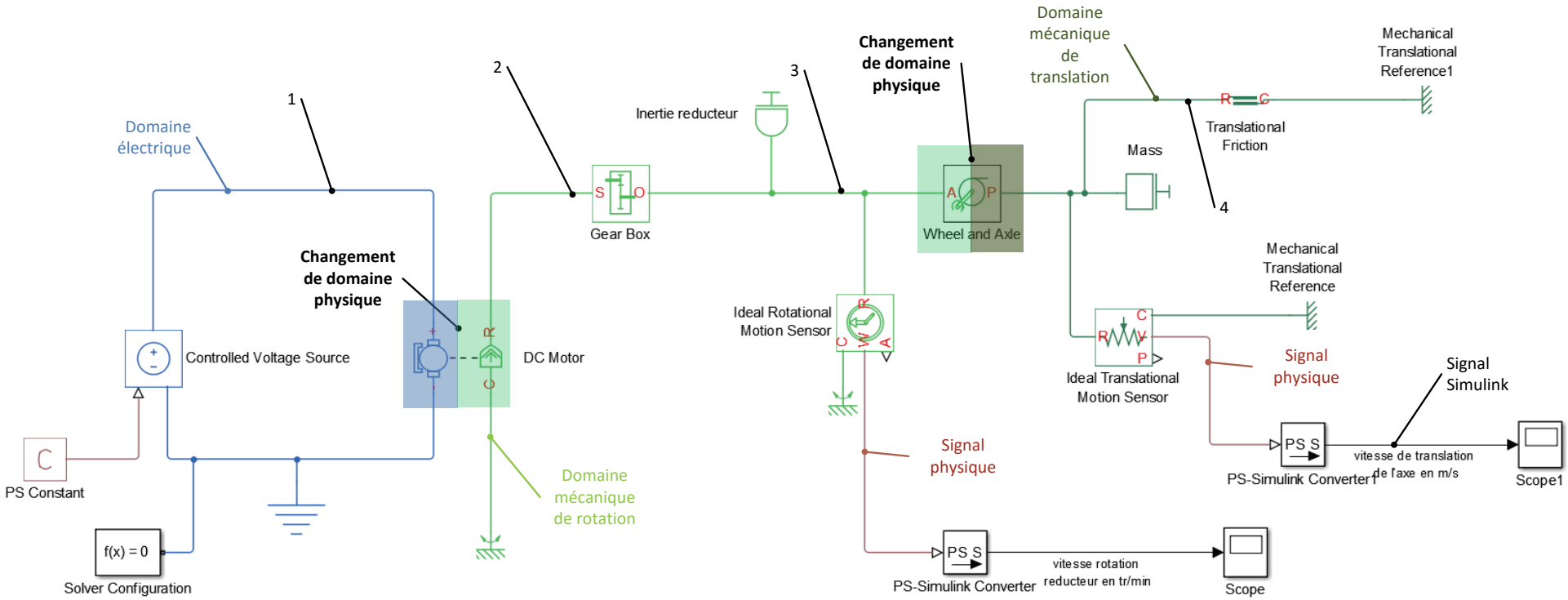
Fonction du composant	Représentation	Bibliothèque
Source de tension variable		Simscape/Fondation Library/Electrical/Electrical Sources
Source de signal physique		Simscape/Fondation Library/Physical Signales/Sources
Moteur à courant continu		Simscape/SimElectronics/Actuators and drivers/ Rotational Actuators
Réducteur		Simscape/Fondation Library/Mechanical/Mechanisms
Inertie		Simscape/Fondation Library/Mechanical/Rotational Elements
Conversion mouvement de rotation en mouvement de translation		Simscape/Fondation Library/Mechanical/Mechanisms
Masse		Simscape/Fondation Library/Mechanical/Translational Elements
Frottement sec et visqueux en translation		Simscape/Fondation Library/Mechanical/Translational Elements
Capteur de position ou de vitesse de translation		Simscape/Fondation Library/Mechanical/Mechanical Sensors
Capteur d'angle ou de vitesse de rotation		Simscape/Fondation Library/Mechanical/ Mechanical Sensors

Figure 113 : les composants nécéssaires à la modélisation de l'axe linéaire avec Simscape

Le modèle **Simscape** de l'axe linéaire est présenté Figure 114 et permet de visualiser les différents domaines physiques qui interviennent dans la modélisation ainsi que les composants qui réalisent les conversions entre les domaines physiques. Une correspondance est donnée entre les connexions des différents domaines physiques et les éléments réels associés.



Correspondance physique des connexions:
1: fil électrique (domaine électrique)
2: arbre sortie moteur (domaine mécanique de rotation)
3: arbre sortie réducteur (domaine mécanique de rotation)
4: courroie (domaine mécanique de translation)

Figure 114 : visualisation des domaines physiques intervenant dans la modélisation de l'axe linéaire avec Simscape

2. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, exécuter la commande **File/New/Model** pour créer un nouveau fichier.

Glisser/Déposer les différents blocs à partir des bibliothèques, les disposer dans la fenêtre de travail et les relier pour obtenir la configuration de la Figure 115.

La Figure 116 donne quelques commandes utiles à la mise en forme d'un modèle.

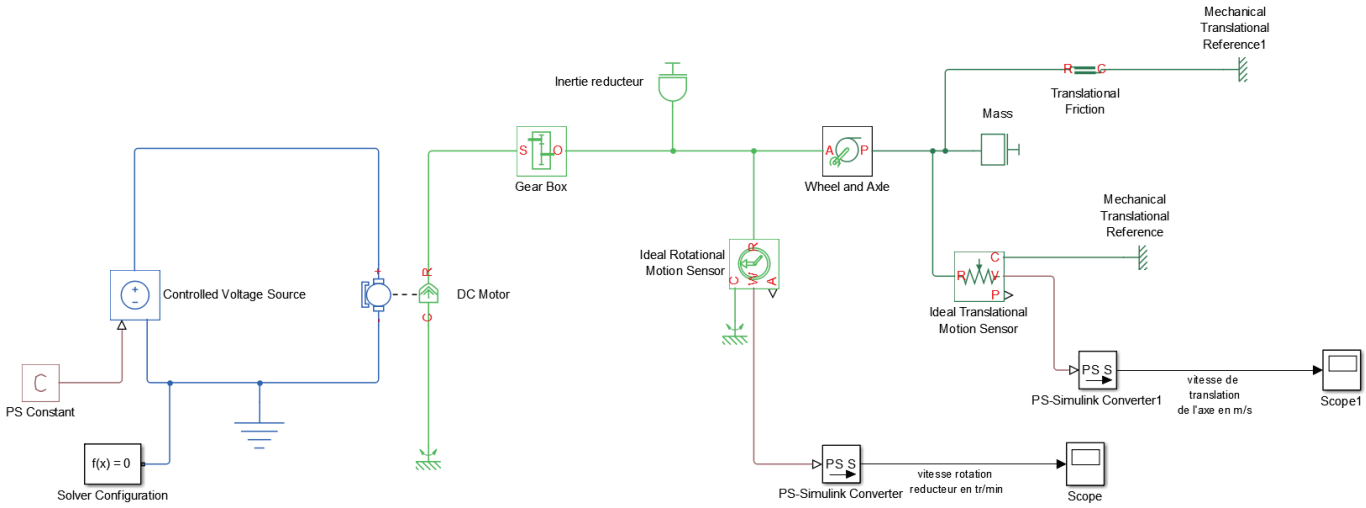


Figure 115 : modèle Simscape de l'axe linéaire

Commandes utiles	
Fonctions	Actions
Donner un nom à un signal Simulink	Clic droit directement sur la connexion du signal Simulink, sélectionner **Properties,** puis compléter le champ **Signal Name** dans la fenêtre. Le nom du signal va apparaitre dans la légende du scope.
Renommer un bloc	Double-clic directement sur le nom du composant puis taper le nouveau nom directement (attention deux composants ne peuvent pas avoir le même nom)
Supprimer le nom d'un composant	Clic droit sur le composant, sélectionner **Format** puis désélectionner **Show Block Name**

Figure 116 : commandes utiles

3. Paramétrage des composants

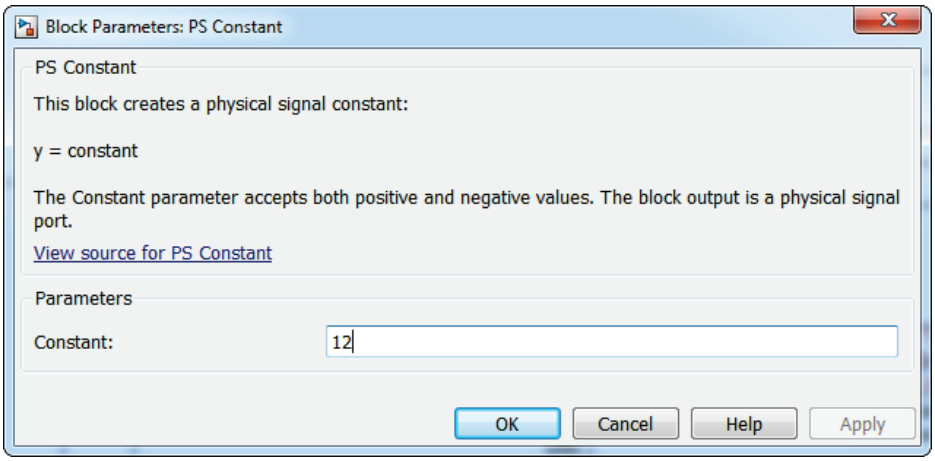
Effectuer le paramétrage des différents blocs conformément aux informations données ci-dessous.

Paramétrage

PS Constant

Simscape/Fondation Library/Physical Signales/Sources

La valeur indiquée dans ce composant permet de définir la valeur de la tension de commande de la source de tension variable.
Le paramétrage de ce bloc consiste à saisir la valeur du signal physique, ici 12 V.

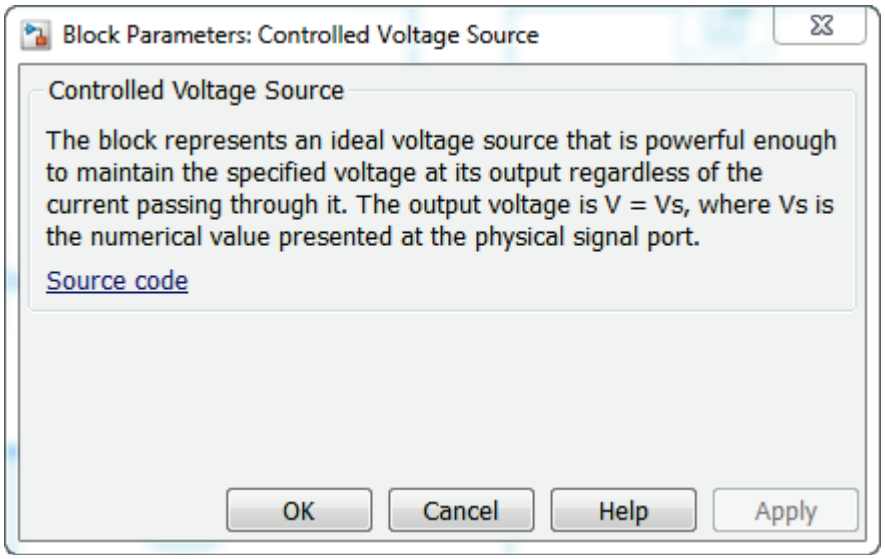


Paramétrage

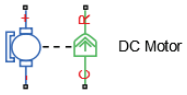
CONTROL VOLTAGE SOURCE

Simscape/Fondation Library/Electrical/Electrical Element

Ce composant ne nécessite aucun paramétrage.
Le port correspondant au signal physique, indique à la source de tension variable la valeur de la tension imposée au borne de la source.



DC MOTOR

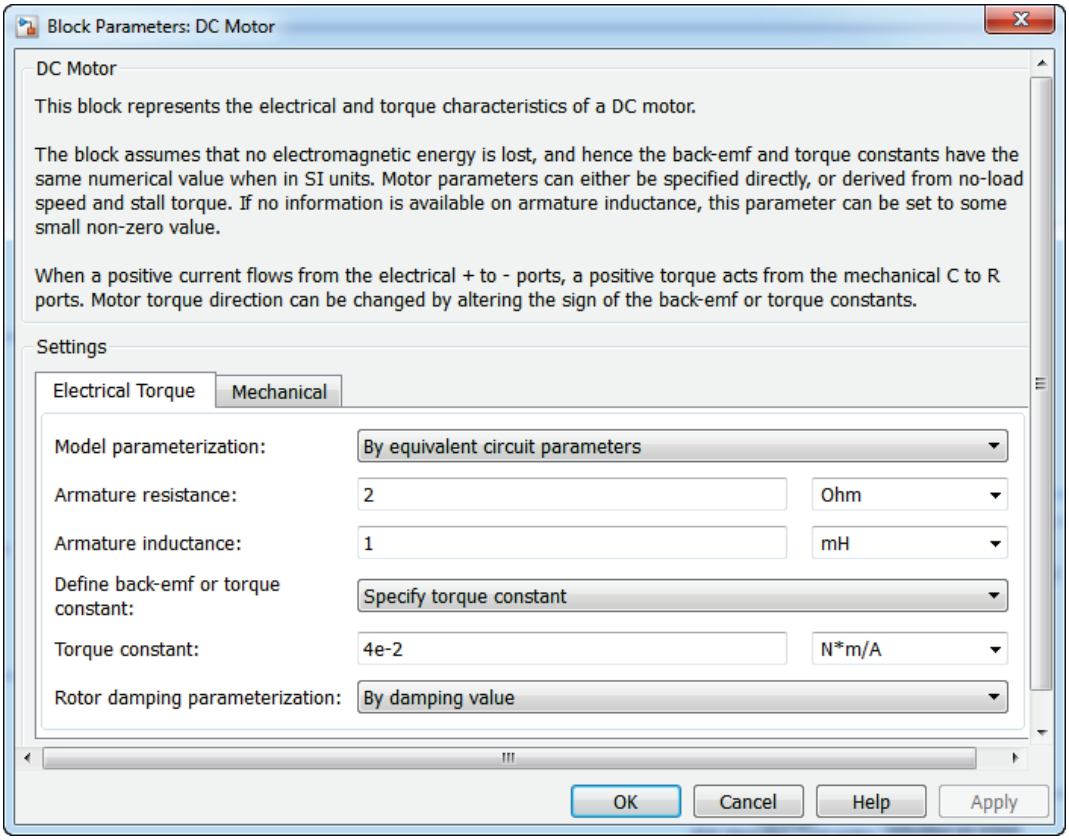


Simscape/SimElectronics/Actuators and drivers/
Rotational Actuators

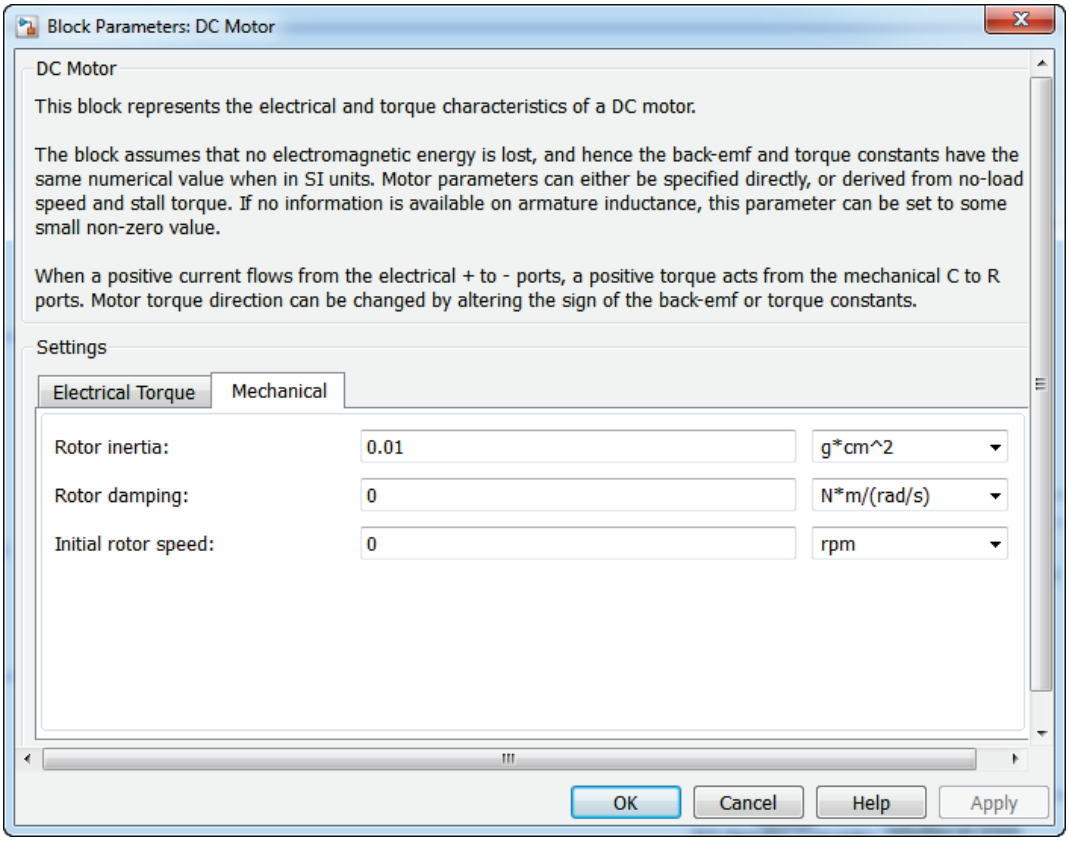
Ce composant modélise un moteur à courant continu. Il possède 4 ports de type **PCP**, 2 du domaine électrique (**+** et **-**) et 2 du domaine mécanique de rotation (**C** et **R**). Le PCP du domaine mécanique **C** (**C**age, partie fixe) sera relié à une référence mécanique de rotation que l'on pourra associée au bâti. L'autre port mécanique de type **R** (**R**otor, partie mobile) sera associé à l'arbre de sortie du moteur.

Ce bloc permet de disposer d'un moteur à courant continu déjà modélisé et de permettre un paramétrage à partir des données constructeur. Deux onglets permettent de saisir les paramètres électriques et les paramètres mécaniques.

Onglet **Electrical Torque**



Onglet **Mechanical**



Par défaut le **Model Parameterization** de l'onglet **Electrical Torque** est réglé sur **By equivalent circuit parameters**. En fonction des informations disponibles dans le fiche constructeur du moteur, il sera possible de modifier le type de paramètres à saisir en choisissant **By stall Torque & and no-load speed** ou **By rated power, rated speed & noload speed.**

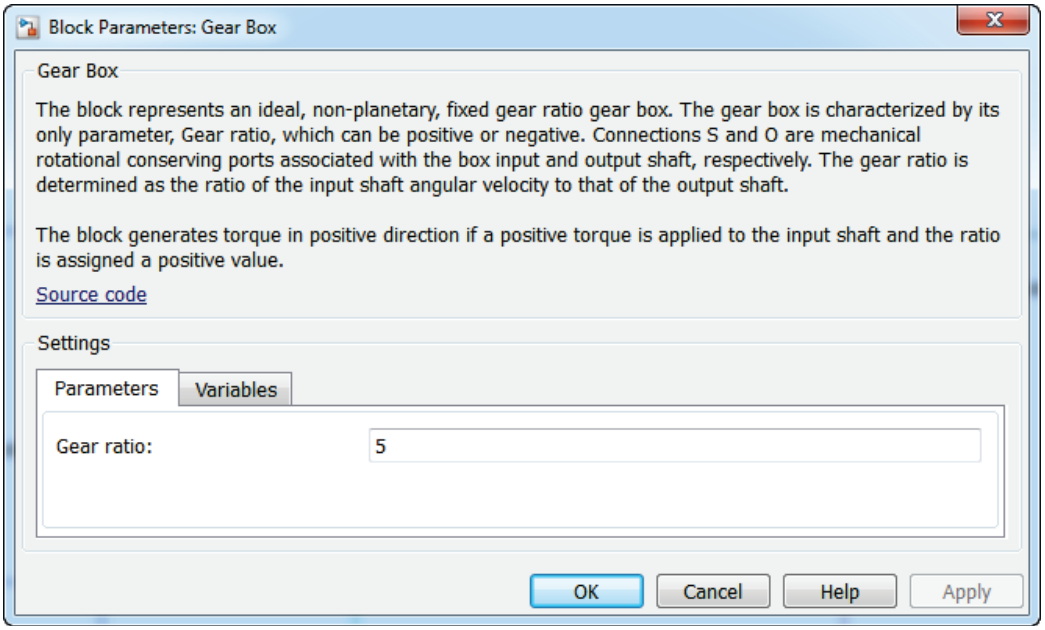
Paramétrage

GEAR BOX

Simscape/Fondation
Library/Mechanical/Mechanisms

Ce composant permet de modéliser le comportement d'un réducteur parfait.

Par convention le Gear ratio est défini tel que $Gear\ ratio = \frac{\omega_{entrée}}{\omega_{sortie}}$ ce qui signifie que pour un réducteur de rapport de réduction $r = \frac{1}{5} = \frac{\omega_{sortie}}{\omega_{entrée}}$ le Gear ratio à spécifier sera égal à 5.

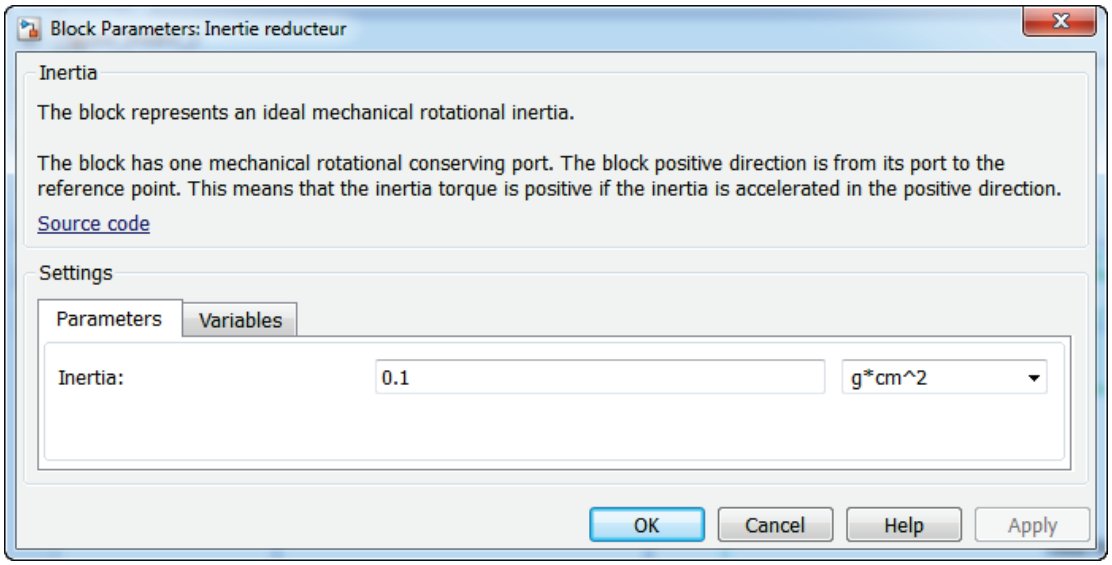


Paramétrage

INERTIA

Simscape/Fondation
Library/Mechanical/Rotational Elements

Ce composant permet de modéliser une inertie associée à une connexion du domaine mécanique de rotation.



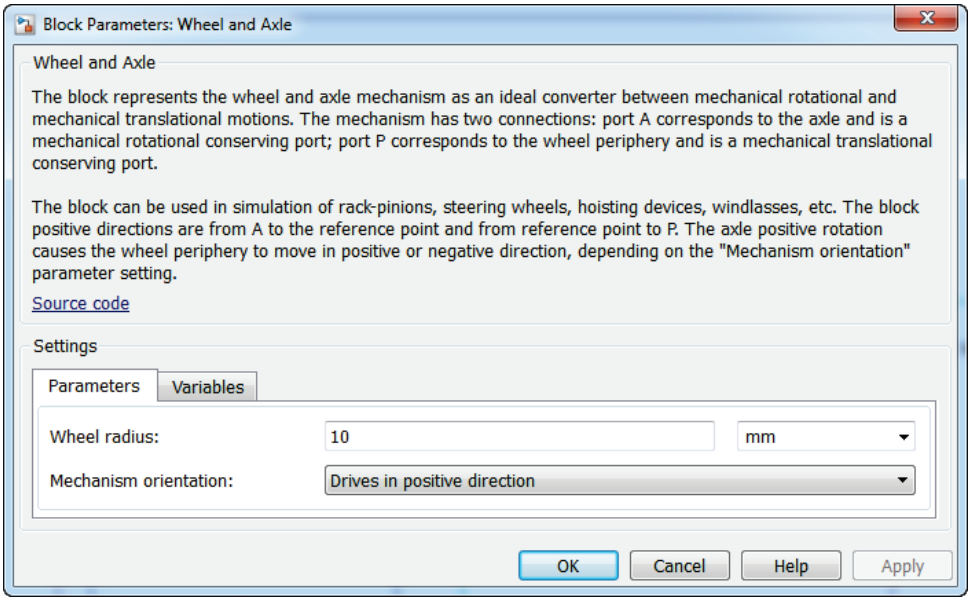
Il est également possible de spécifier une vitesse initiale pour cette inertie. La modélisation acausale permet de choisir des conditions initiales non nulles (par opposition à l'approche causale par fonction de transfert ou les grandeurs physiques doivent être nulles à t=0).

Paramétrage

WHEEL AND AXLE

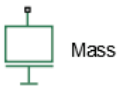
Simscape/Fondation
Library/Mechanical/Mechanisms

Ce composant permet de modéliser une conversion de puissance parfaite entre un mouvement de rotation et un mouvement de translation.
Le paramètre à renseigner est le **rayon** de la poulie.
Il est également possible de spécifier une convention de signe pour réaliser la conversion.



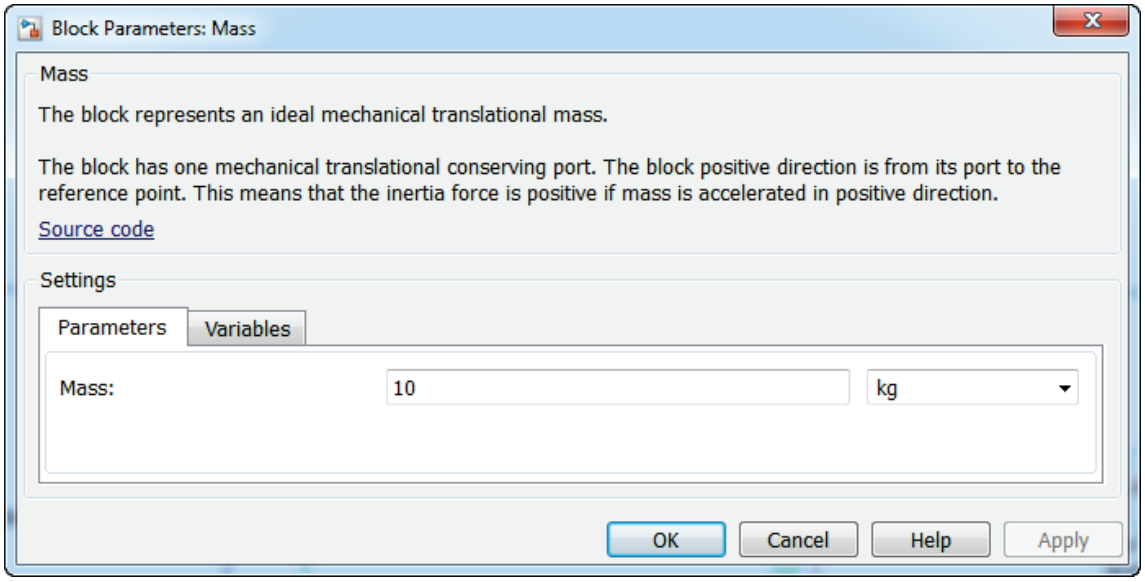
Paramétrage

MASS



Simscape/Fondation
Library/Mechanical/Translational Elements

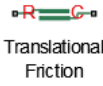
Ce composant permet de modéliser une masse associée à une connexion du domaine mécanique de translation.
Ici, la masse correspond à la masse du chariot entraîner en translation par la poulie.



Il est également possible de spécifier une vitesse initiale pour cette masse.

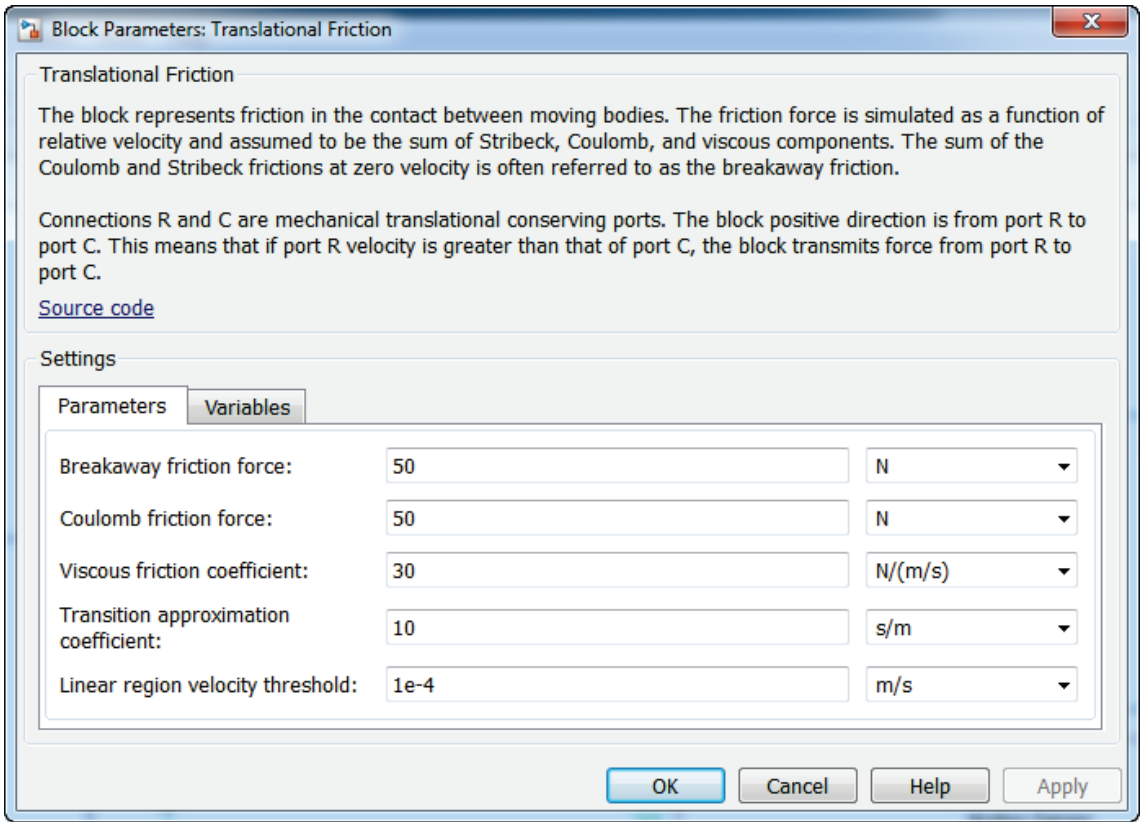
Paramétrage

TRANSLATIONAL FRICTION



Simscape/Fondation
Library/Mechanical/Translational Elements

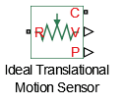
Ce composant permet de modéliser une force de frottement (sec et visqueux) qui s'exerce sur un élément du domaine mécanique de translation. Le port **C** sera relié au bâti et le port **R** à la connexion qui subit l'effort de frottement. L'effort sera exercé par le bâti sur l'élément du domaine mécanique de translation auquel, le port **R** est relié. L'effort s'exercera toujours en s'opposant au déplacement. Ici, il s'agira d'un effort exercé par le bâti sur le chariot qui jouera le rôle de perturbation pour le système.



Breakaway friction force : force limite d'adhérence (force à partir de laquelle le mouvement commence)
Coulomb friction force : force de frottement (qui s'exerce pendant le mouvement)
Viscous friction coefficient : coefficient de frottement visqueux

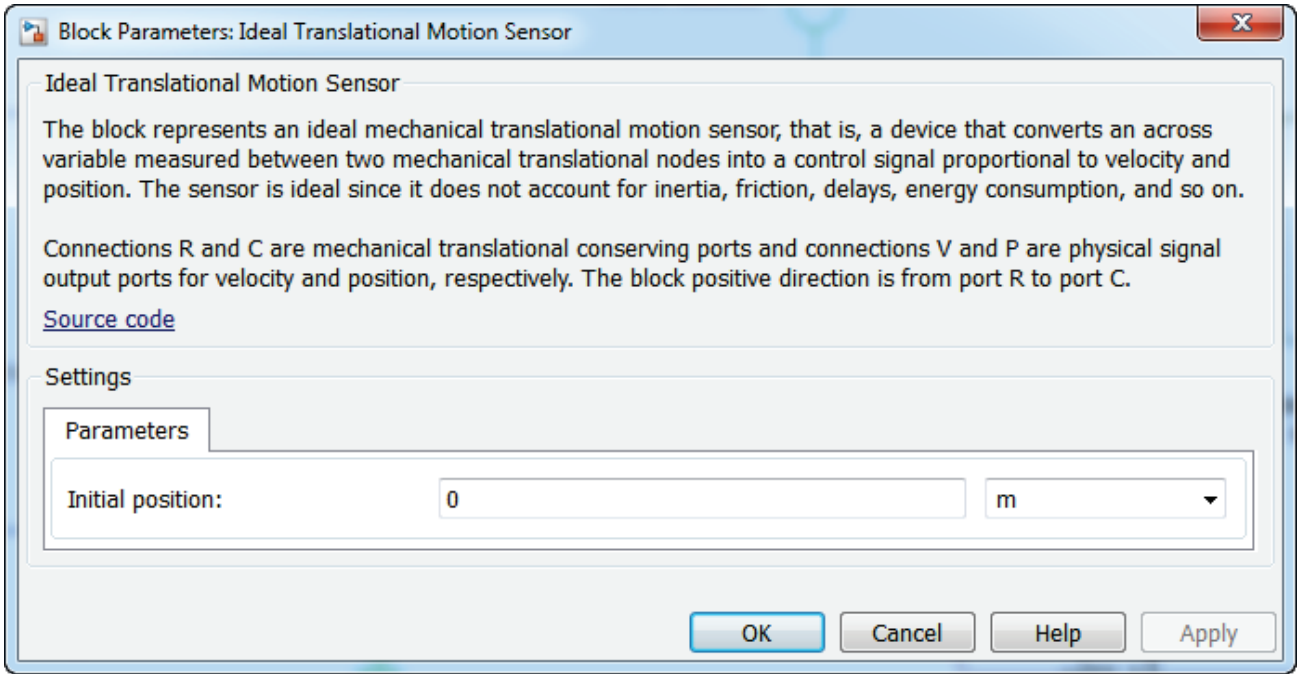
Paramétrage

IDEAL TRANSLATIONAL MOTION SENSOR



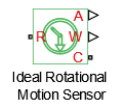
Simscape/Fondation
Library/Mechanical/Mechanical Sensors

Ce capteur permet de mesurer une vitesse de translation (variable de type « Across ») ou une position en translation. Il se monte donc en parallèle. Le port **R** est relié à la connexion dont on souhaite connaître la vitesse par rapport au port **C** que l'on relie à la référence choisie. Le signal physique que l'on peut prélever est soit la vitesse de translation (port **V**) soit la position (port **P**).
Ici, on souhaite connaître la vitesse de déplacement du chariot par rapport au bâti.
Il est également possible de spécifier la valeur initiale de la mesure en début de simulation (0 par défaut).



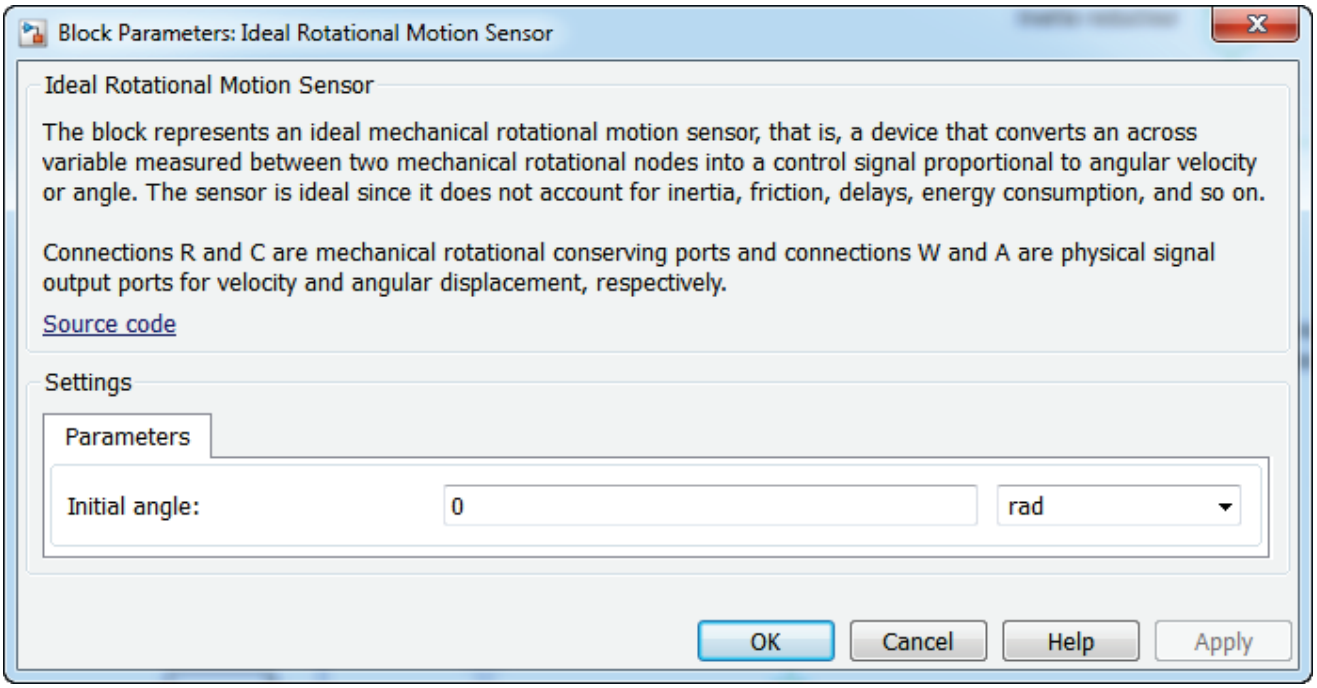
Paramétrage

IDEAL ROTATIONAL MOTION SENSOR

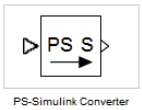


Simscape/Fondation
Library/Mechanical/Rotational Sensors

Ce capteur permet de mesurer une vitesse de rotation (variable de type « Across ») ou un angle. Il se monte donc en parallèle. Le port **R** est relié à la connexion dont on souhaite connaître la vitesse par rapport au port **C** que l'on relie à la référence choisie. Le signal physique que l'on peut prélever est soit la vitesse de rotation (port **W**) soit l'angle (port **A**).
Ici, on souhaite connaître la vitesse de déplacement du chariot par rapport au bâti.
Il est également possible de spécifier la valeur initiale de la mesure en début de simulation (0 par défaut).



Paramétrage

PS-SIMULINK CONVERTER 

Simscape/Utilities

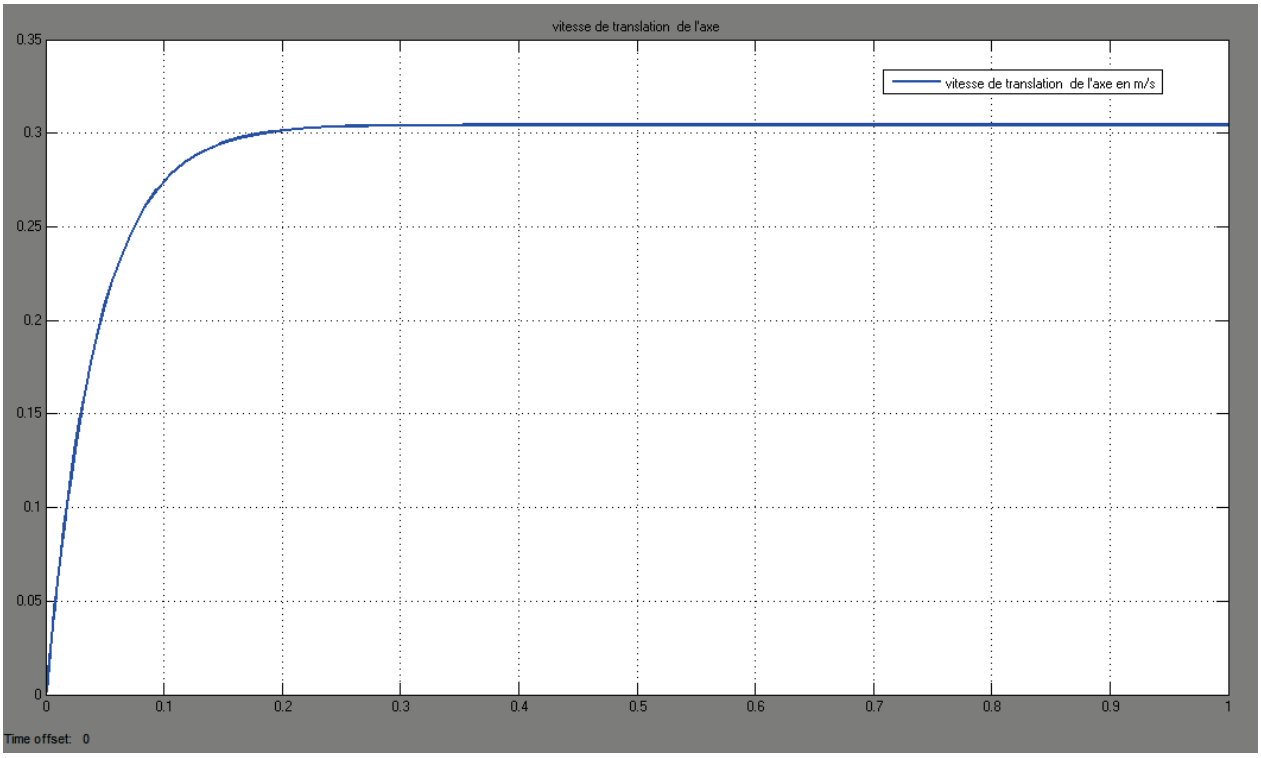
Ce composant permet de convertir un signal physique image d'une grandeur mesurée en signal Simulink afin de pouvoir être affiché dans un Scope. Spécifier que la vitesse de translation du chariot est relevée en m/s et la vitesse de rotation de l'arbre en tr/min (rpm pour Simscape)

Le fichier contenant le modèle paramétré est disponible sous le nom **axe_lineaire_0.slx**

4. Simulation du modèle en boucle ouverte

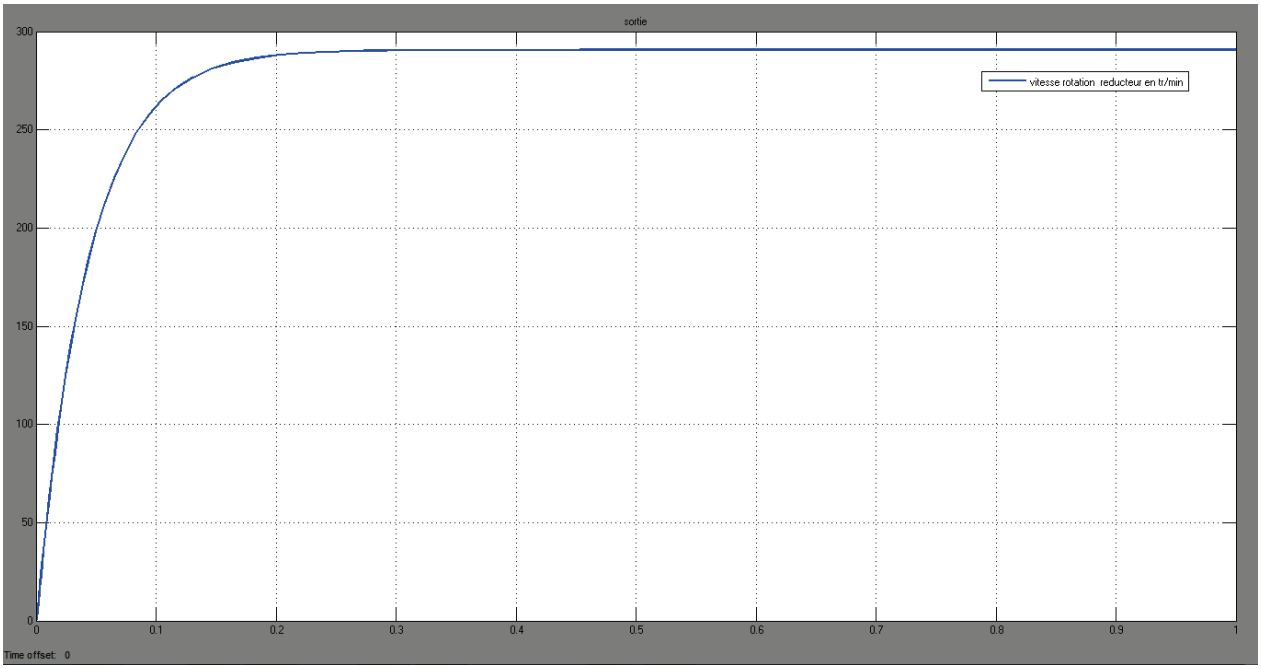
Lancer la simulation en spécifiant un temps de simulation de **1s**.

Observer la vitesse de translation en double-cliquant sur le scope.



La vitesse de translation du chariot se stabilise autour de 0.3 m/s.

Observer la vitesse de rotation de l'arbre de sortie du réducteur en double-cliquant sur le scope.



La vitesse de rotation de l'arbre de sortie de réducteur se stabilise autour de 290 tr/min.

5. Utilisation du Data-logger de Simscape

Lors de la conception d'un modèle, il est très utile d'avoir accès à l'ensemble des variables qui interviennent dans le modèle. La première solution peut consister à implanter des capteurs partout où il est nécessaire d'observer des variables. Cette méthode se montrera vite longue et fastidieuse et rendra le modèle très peu lisible.
Pour répondre à ce besoin de conception, **Simscape** offre la possibilité de stocker l'évolution de toutes les variables qui évoluent au sein du modèle. Il sera alors possible à l'aide du Data-logger de **Simscape** de visualiser l'évolution de toutes les variables.

La fonction qui réalise la visualisation des données doit être accessible dans le « path » de MATLAB. Cette fonction se nomme « **ssc_explore.m** » et reçoit comme argument la variable qui contient toutes les données de la simulation.

Le dossier « **Simscape_logger** » qui contient cette fonction doit être dans le « path » de **MATLAB** afin de rendre la fonction « **ssc_explore.m** » accessible depuis la fenêtre de commande. Le dossier « Simscape_logger » est déjà inclus dans le dossier « Modelisation_multi_physique_modeles/ Chapitre_2_Simscape » qui a déjà été placé dans le path de **MATLAB**.

Il faut maintenant spécifier le nom de la variable qui stockera toutes les données de la simulation avant d'utiliser la fonction « **ssc_explore.m** ».

Cliquer sur **Simulation/Model Configuration Parameters**, puis choisir l'onglet **Simscape** dans la partie gauche de la fenêtre.

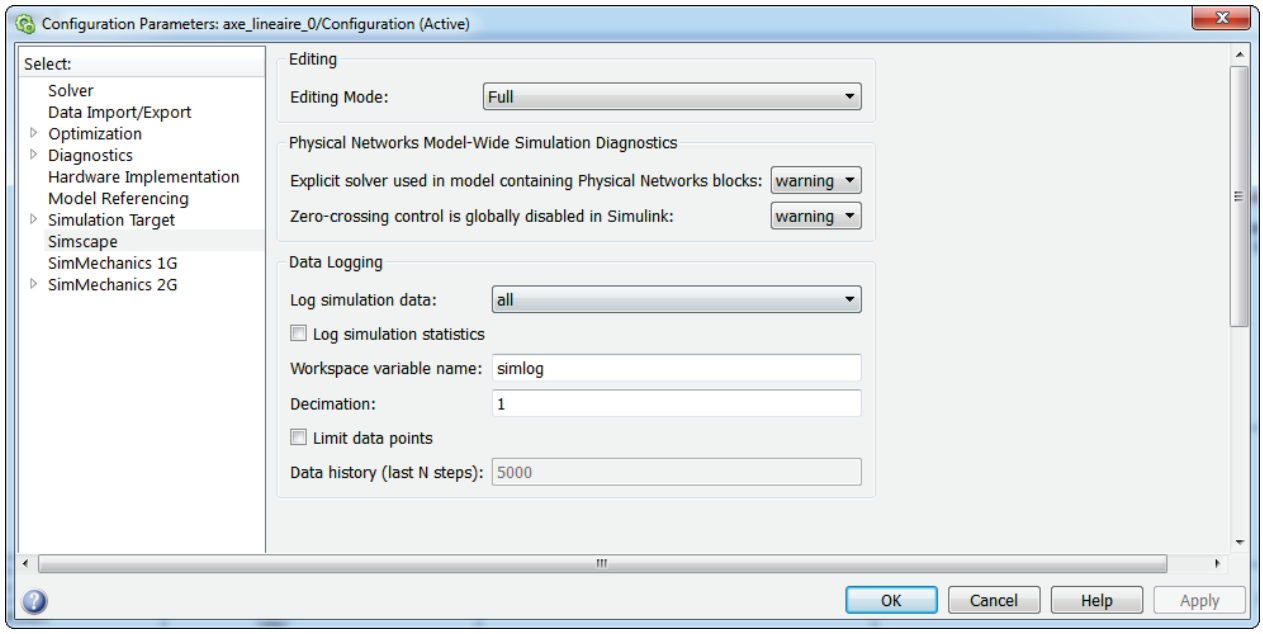


Figure 117 : spécification des variables Simscape à prendre en compte dans le logger

Choisir **all** dans le menu **Log simulation data.**

Choisir le nom de la variable qui stockera les données de la simulation dans le champ **Workspace Variable Name** (ici **simlog**, mais ce nom peut être modifié)

Désélectionner **Limit data points** afin d'être certain de récupérer toutes les données de la simulation.

Relancer la simulation.

Retourner dans la fenêtre de commande de MATLAB et taper **ssc_explore(simlog)**

Le logger de **Simscape** s'affiche et il est possible d'observer toutes les données de la simulation.

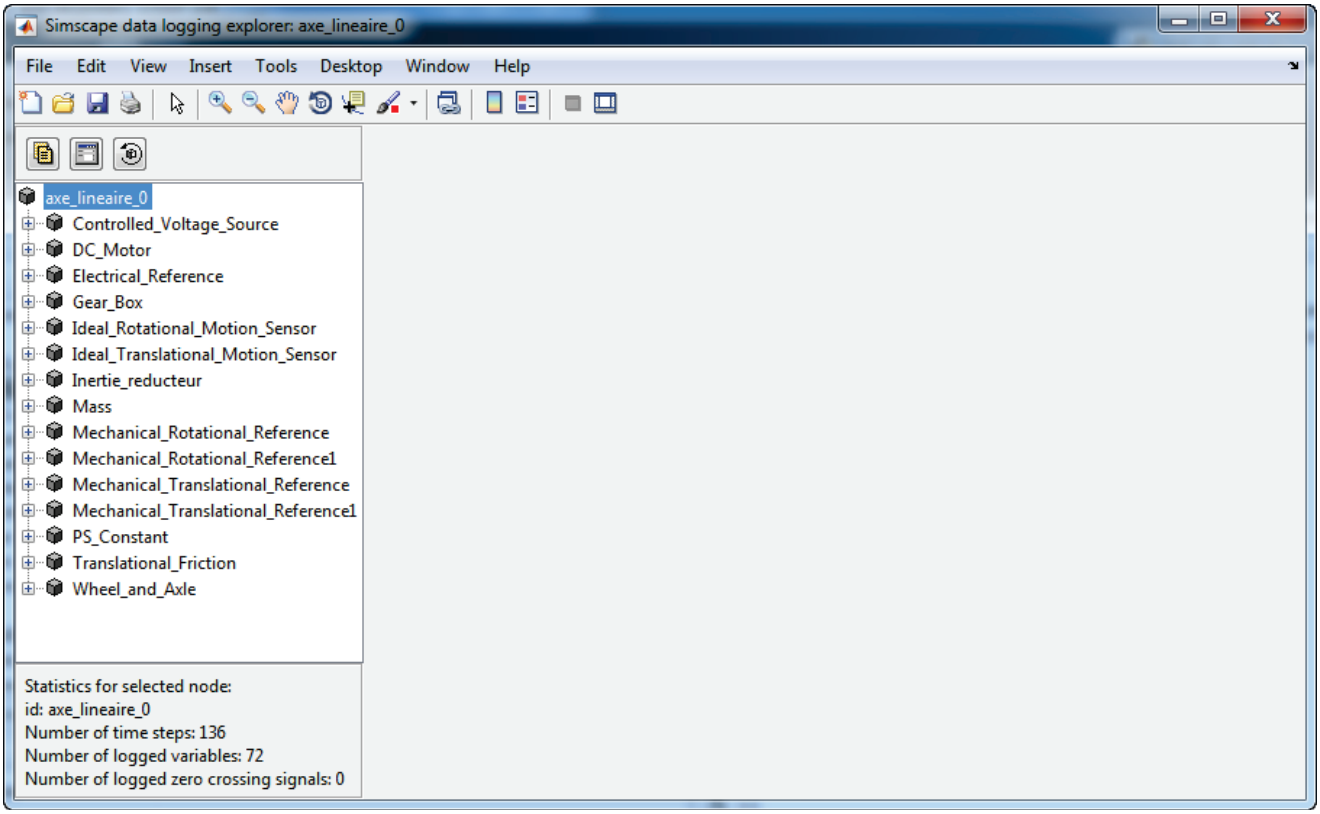
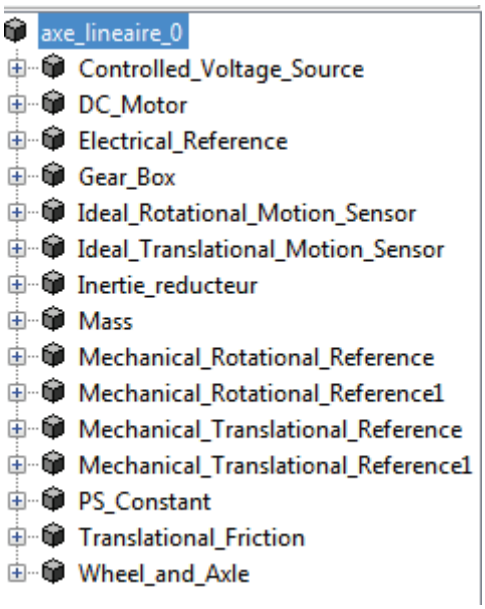


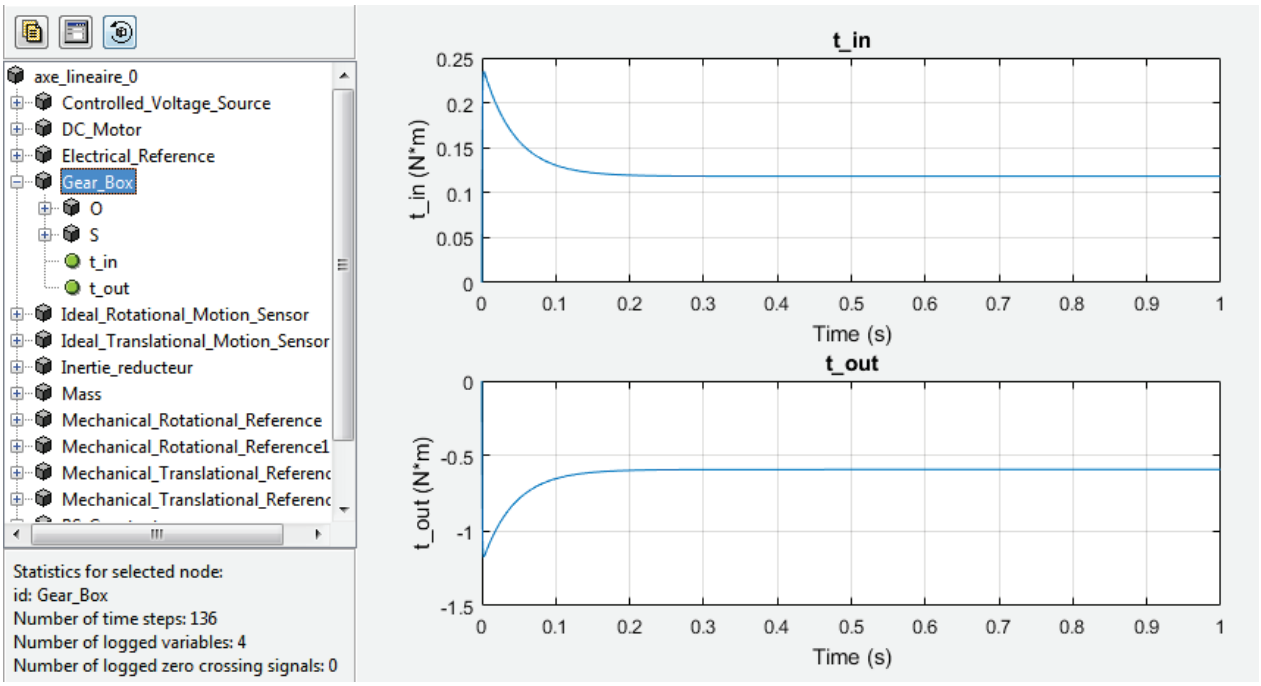
Figure 118 : fenêtre du logger de Simscape

Dans la partie gauche de la fenêtre apparaît le nom des composants du modèle.

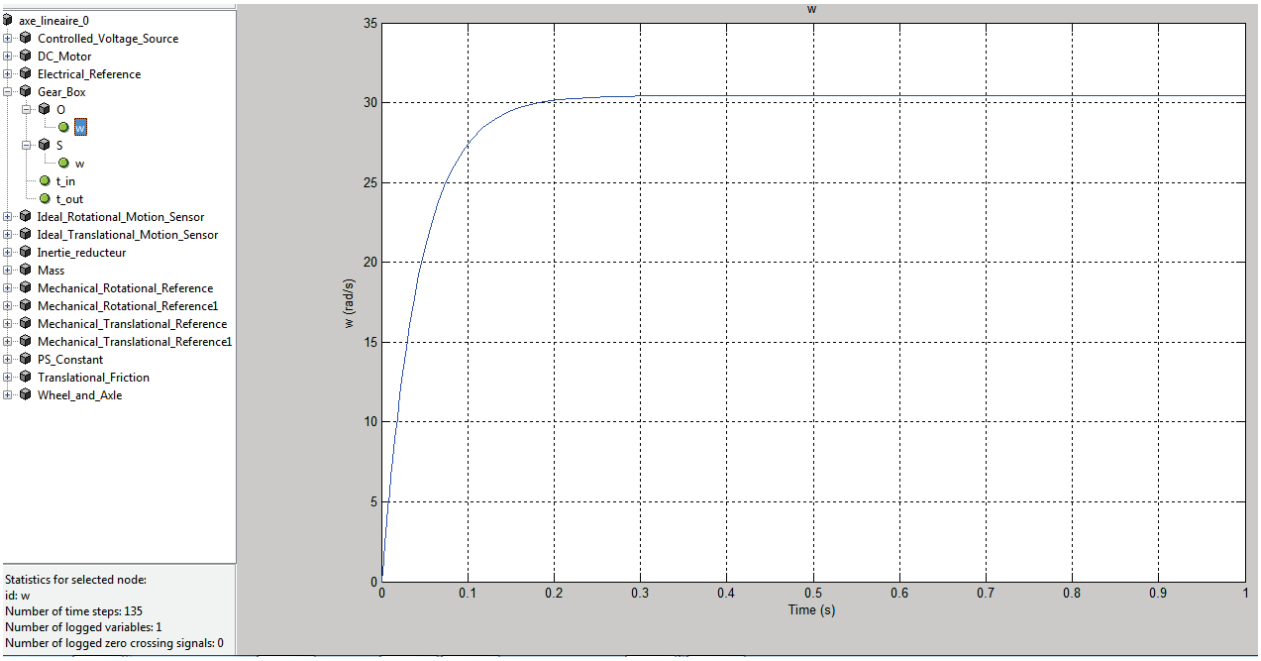


Choisir par exemple le composant **Gear_Box** et développer l'arborescence pour voir apparaître toutes les variables du composant. L'arborescence fait apparaître les deux ports **O** et **S** du composant.

Deux courbes apparaissent montrant l'évolution du **Moment** en N.m (variable de type « Through ») à l'entrée **S** et à la sortie **O** du composant.



Il est également possible d'accéder à la vitesse de rotation (variable de type « Across ») au niveau de l'entrée **O** et de la sortie **S** du composant.



Le principe du Data-loger est de pouvoir observer les variables de types « Across » et « Through » pour tous les composants du modèle. Pour chaque composant, les ports de type **PCP** apparaissent dans l'arborescence. Il suffit de cliquer sur la variable que l'on souhaite observer pour visualiser la courbe.

Le logger de **Simscape** devient vite indispensable et permet un gain de temps considérable lors de la conception des modèles.

6. Création de sous-systèmes

Afin de simplifier la compréhension et l'exploration du modèle, il est possible de créer des sous-systèmes. Cette démarche deviendra vite indispensable lors de la conception de modèles complexes.
Le principe consiste à sélectionner un groupe de composants pour créer un sous-système.

Nous allons créer un sous-système comprenant le moteur et son alimentation.

Sélectionner comme indiqué sur la Figure 119 les composants qui composeront le sous-système puis faire glisser la souris sur l'icône **Create Subsystem** en bas à droite de la sélection.

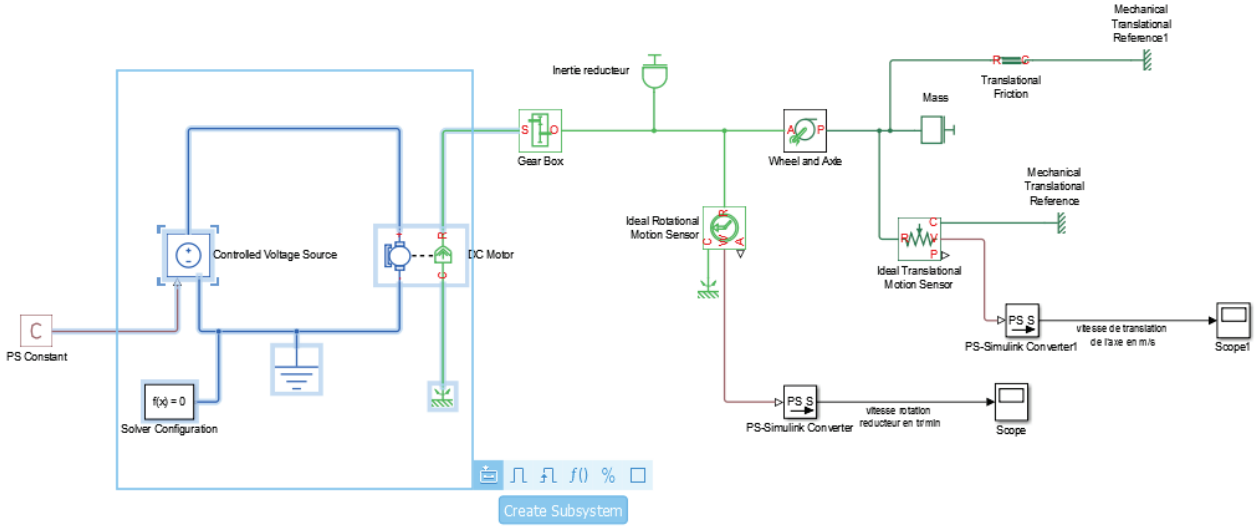


Figure 119 : création d'un sous-système

Il est également possible de faire un **Clic-droit** sur la sélection et de choisir **Create Subsystem from Selection** dans le menu contextuel ou d'utiliser le raccourci clavier **Ctrl+G.**

Redimensionner éventuellement le sous-système pour voir apparaître les ports **Conn1** et **Conn2** du composant.

Commandes utiles	
Fonctions	Actions
Redimensionner un composant	**Sélectionner** le composant avec le bouton gauche de la souris, puis **tirer** sur les poignées qui apparaissent sur le contour du composant pour le redimensionner

Figure 120 : commandes utiles

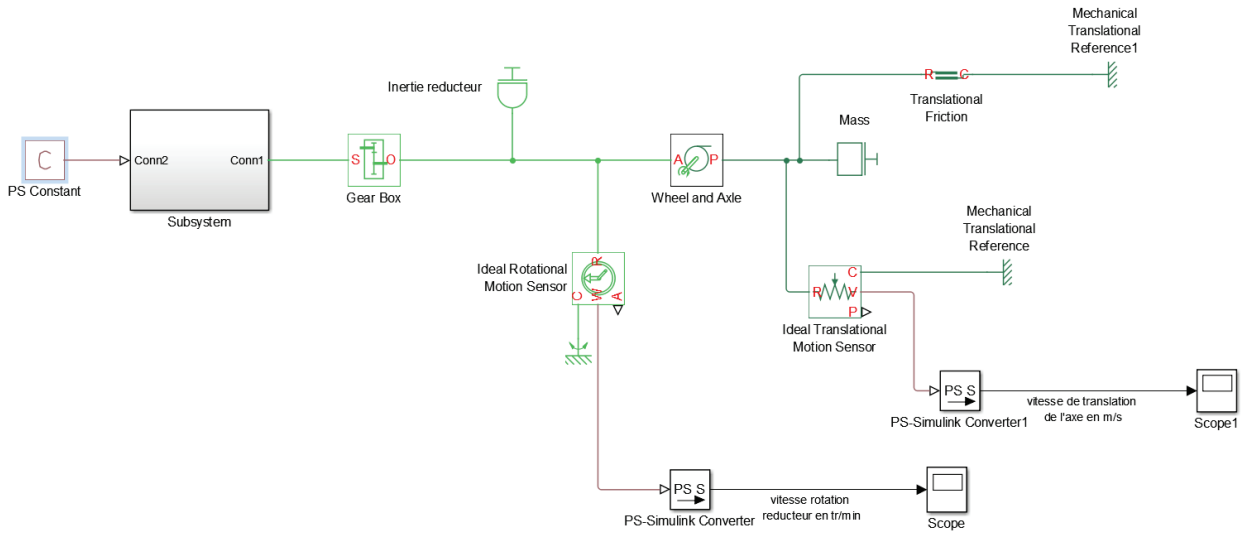


Figure 121 : création d'un sous-système

Pour information, il est également possible de revenir à la forme précédente du modèle en « étendant » le sous-système. Pour cela faire un Clic droit sur le sous-système et choisir la commande **Subsystem & Model Reference/Expand Subsystem.**

Double-cliquer sur le nom du sous-système situé sous le cadre du sous-système pour le remplacer par **«Moteur »**

Double-cliquer ensuite sur le sous-système pour voir les composants qui le constituent.

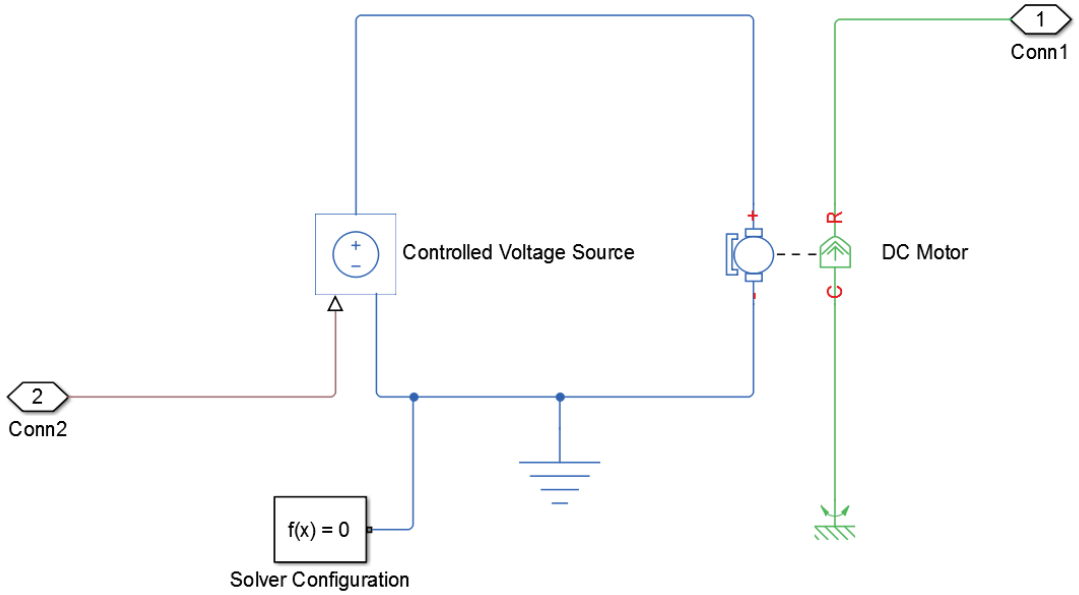


Figure 122: visualisation du contenu du sous-système

On peut observer que les connexions avec l'extérieur apparaissent sous la forme de ports **Conn1** et **Conn2**.

Il est également possible de renommer ces ports.

Double-cliquer directement sur le nom du port **Conn2** et le remplacer par **« Tension de commande** ».

Faire de même en remplaçant le nom **Conn1** par « **Couple moteur** » (Figure 123)

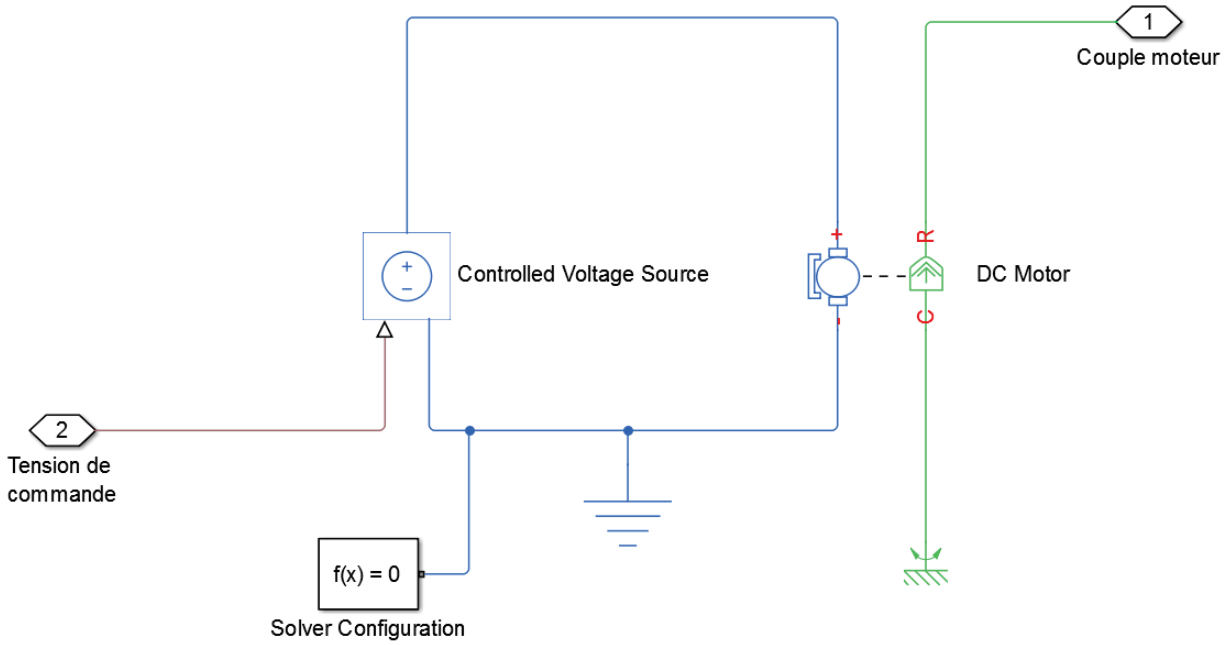


Figure 123 : renommer les ports d'un sous-système

Revenir au modèle global en cliquant sur le nom du modèle global dans la partie gauche de la fenêtre. Le Model Browser (situé à gauche de la fenêtre) permet de naviguer facilement d'un sous-système au système global (Figure 124).

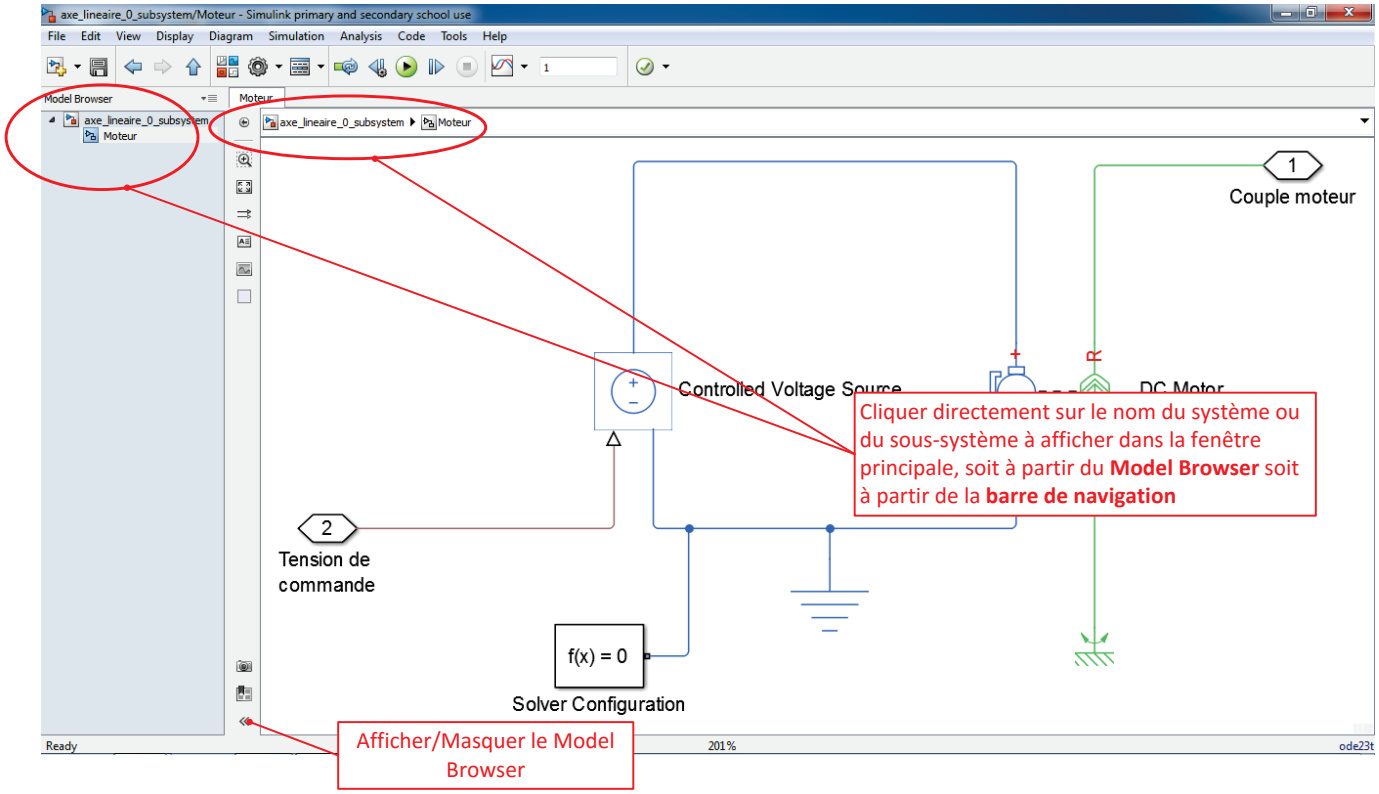


Figure 124 : naviguer dans les sous-systèmes d'un modèle

Vous devez obtenir la configuration de la Figure 125.

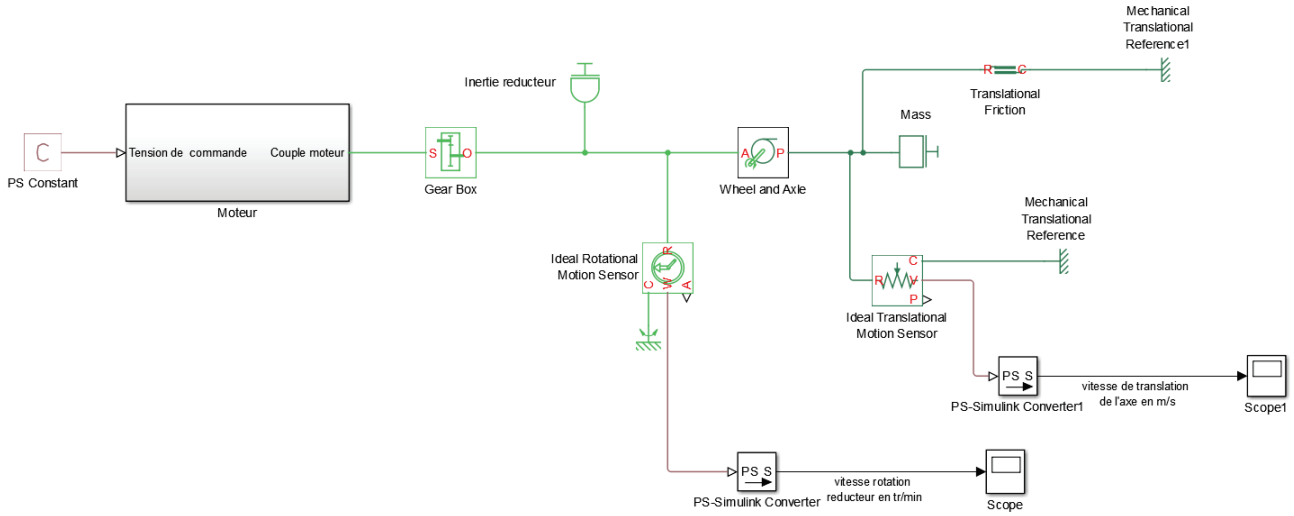


Figure 125 : modèle Simscape de l'axe linéaire avec sous-système « moteur »

Dans l'objectif de créer un asservissement de position, nous allons supprimer le capteur de vitesse de rotation de l'arbre moteur et transformer la mesure de la vitesse linéaire de l'axe en mesure de position.

Supprimez le capteur de vitesse de rotation de l'arbre moteur pour obtenir le schéma de la Figure 126.

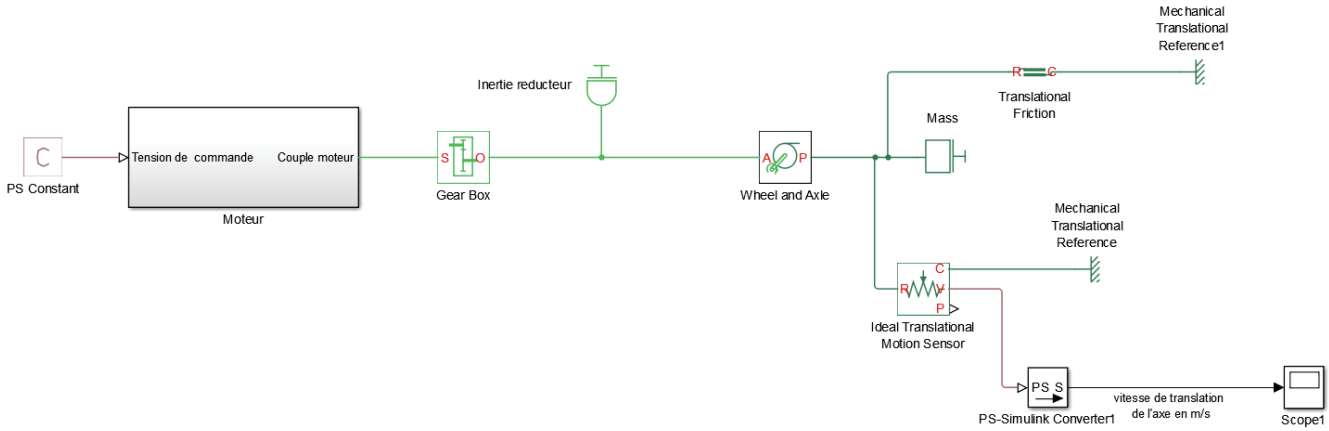


Figure 126 : modèle Simscape de l'axe linéaire sans le capteur de vitesse

Modifier la grandeur physique prélevée sur **l'Ideal Translational Motion Sensor** afin de relever la position linéaire du chariot (Figure 126).

Il faudra changer l'unité dans le bloc **PS-Simulink Converter** et choisir mètre (**m**).

On modifiera également le nom du signal pour obtenir la configuration de la Figure 127.

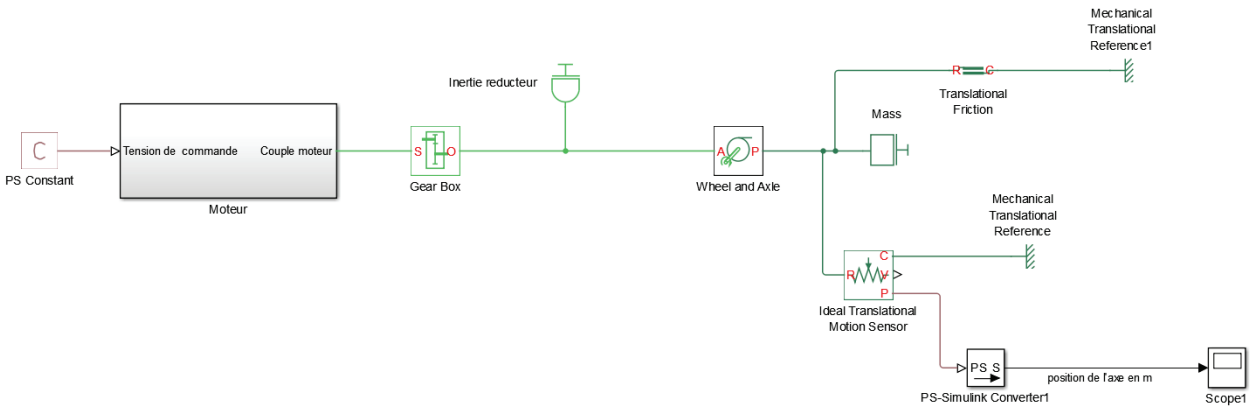


Figure 127 : modèle Simscape de l'axe linéaire avec le nom des signaux

Créer un sous-système « Axe » en sélectionnant les composants conformément à la Figure 128.

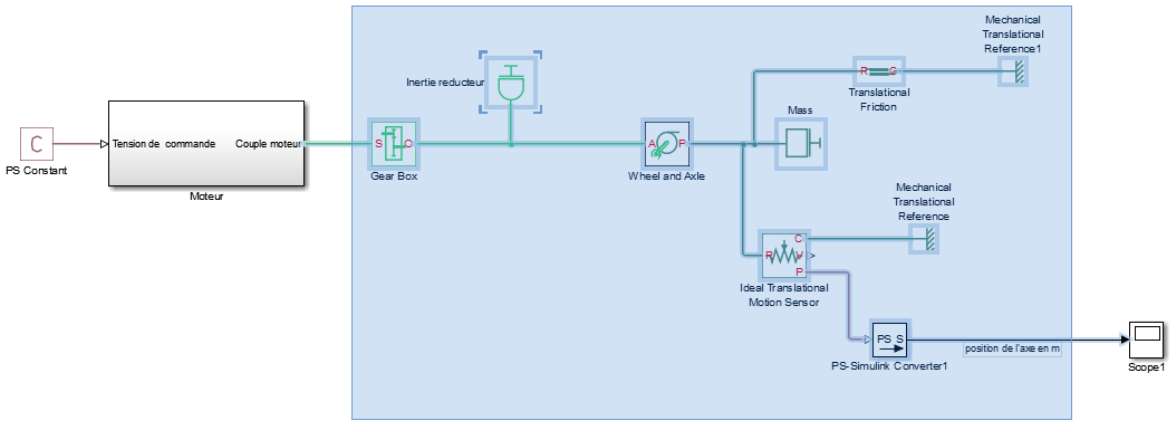


Figure 128 : modèle Simscape de l'axe linéaire, création du sous-système « Axe »

Vous devez obtenir le modèle suivant après redimensionnement du sous-système :

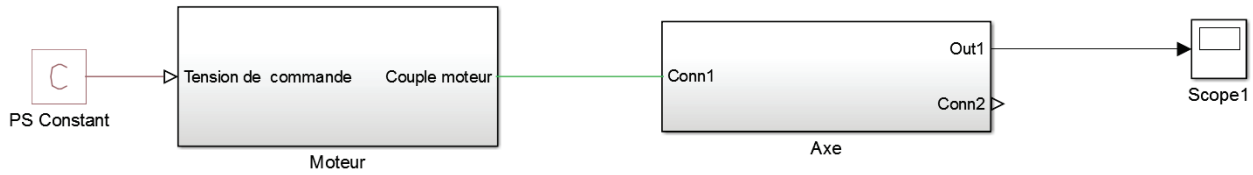


Figure 129 : modèle Simscape de l'axe linéaire avec sous-système « moteur » et « Axe »

On remarque qu'un port supplémentaire **Conn1** a été créé.

Double cliquer sur le sous-système pour l'explorer.

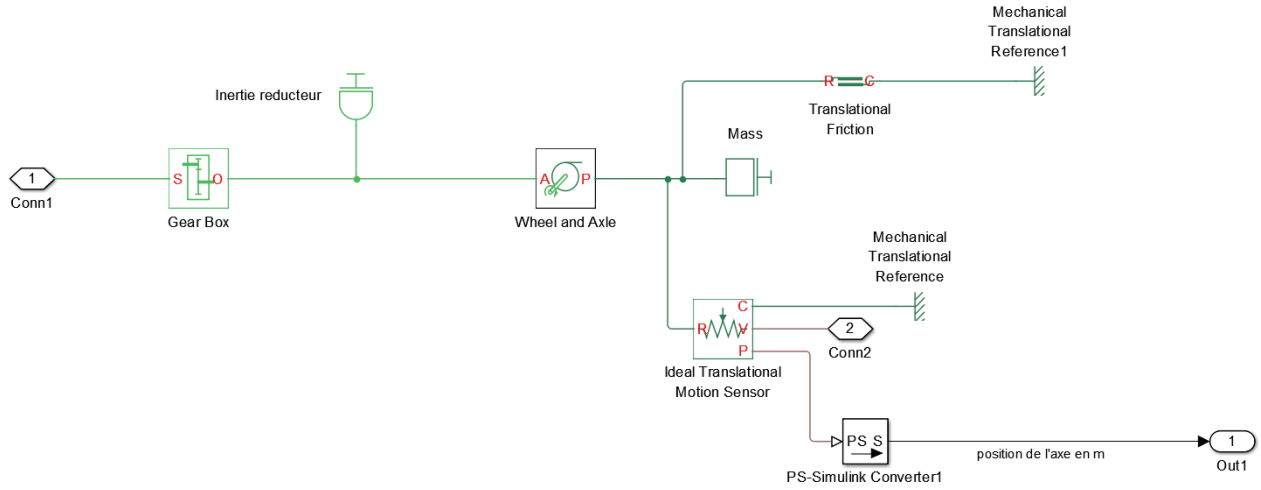


Figure 130 : suppression du port d'un sous-système

Le port **Conn2** a été créé par défaut sur la sortie en vitesse du capteur qui n'était pas utilisée. Lors de la création d'un sous-système, tout port qui n'est pas connecté à l'intérieur du sous-système apparaîtra sous la forme d'un port de communication avec l'extérieur.

Supprimer ce port.

On remarque que le format du port **Conn1** et le format du port **Out1** sont différents. Le port **Conn1** est un port de type **PCP** de **Simscape** (transmetteur de puissance, modélisation acausale) et le port **Out1** est un signal **Simulink** (signal numérique orienté, modélisation causale). Le logiciel permet de faire visuellement la distinction entre ces deux types de ports.

Renommer les ports afin d'obtenir le modèle de l'axe linéaire de la Figure 131

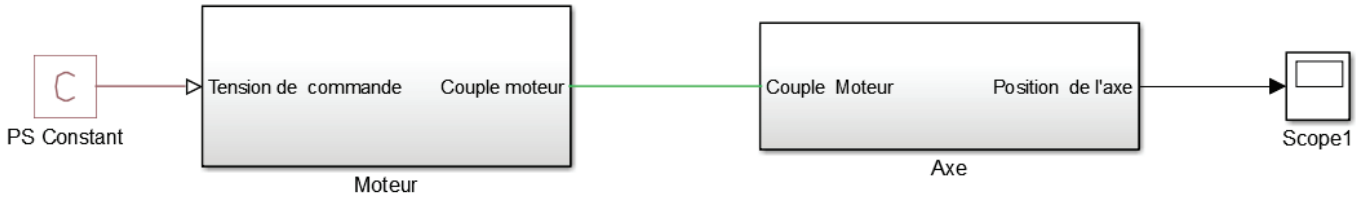


Figure 131 : modèle Simscape terminé de l'axe linéaire

Si nécessaire le modèle correspondant est disponible dans le fichier ***axe_lineaire_1.slx.***

Il est possible d'ajouter une image sur un sous-système. Pour cela cliquer droit sur le sous-système et sélectionner **Mask/Create Mask**

Taper ensuite dans la fenêtre **Mask Editor** (Figure 132) la commande *image('axe.jpg').* Le fichier image correspondant doit impérativement être dans le « path » de **MATLAB** pour s'afficher sur le sous-système.

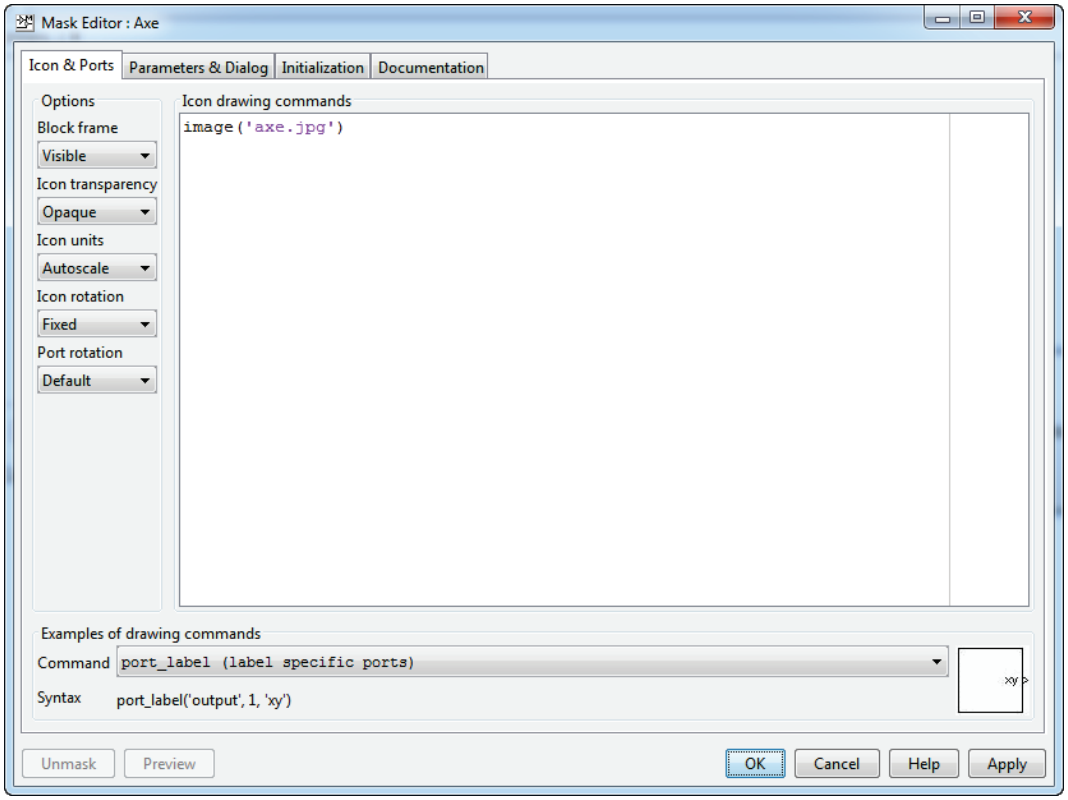


Figure 132 : fenêtre de paramétrage du mask d'un sous-système

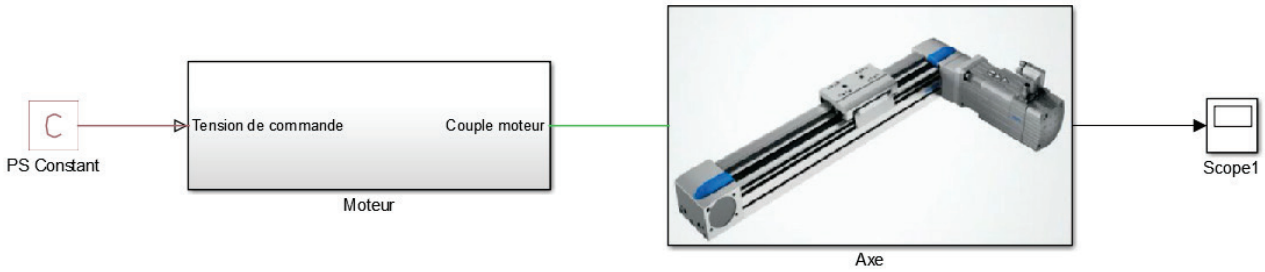


Figure 133 : affichage d'une image sur un sous-système

Redimensionnez éventuellement la taille du sous-système pour que l'image retrouve ses proportions initiales (Figure 133).

7. Modélisation de l'asservissement en position de l'axe

Le modèle ainsi obtenu permet de visualiser la réponse en boucle ouverte de la position du chariot pour une valeur donnée de la tension moteur.

Pour une tension de 12V, la réponse en position de l'axe est la suivante :

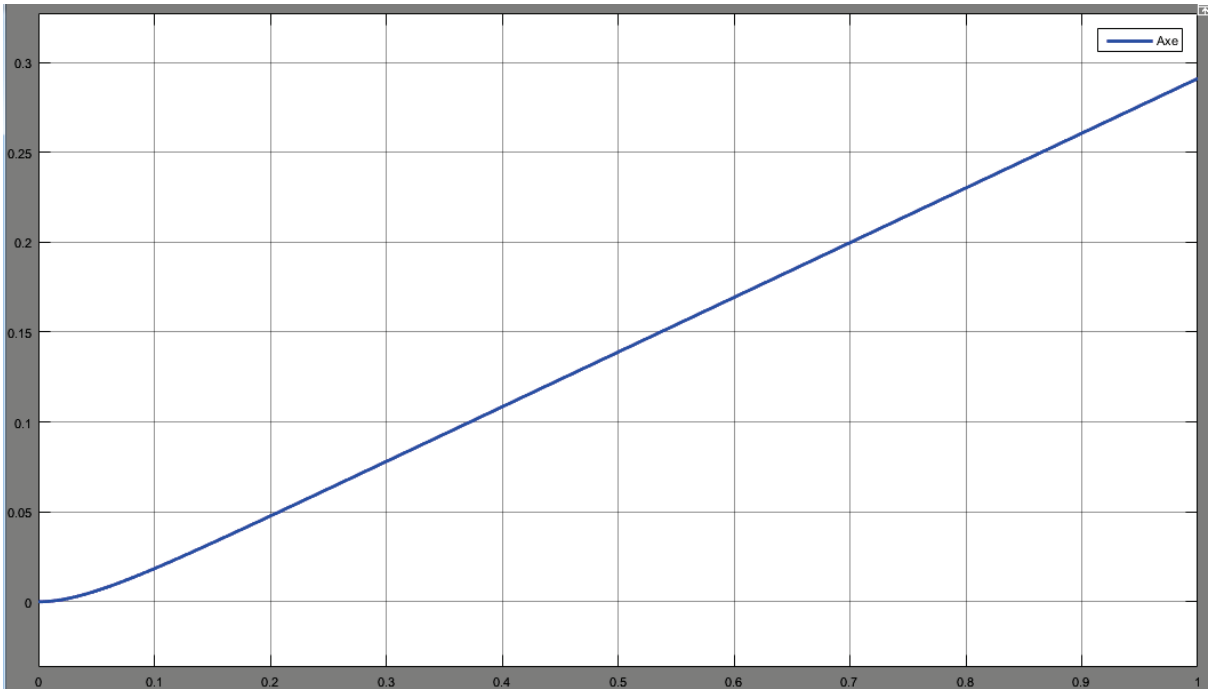


Figure 134 : réponse en position de l'axe linéaire non asservi

On peut maintenant réaliser le modèle de l'asservissement en position de l'axe représenté sur la Figure 135. Les nouveaux composants ajoutés sont décrits Figure 136.

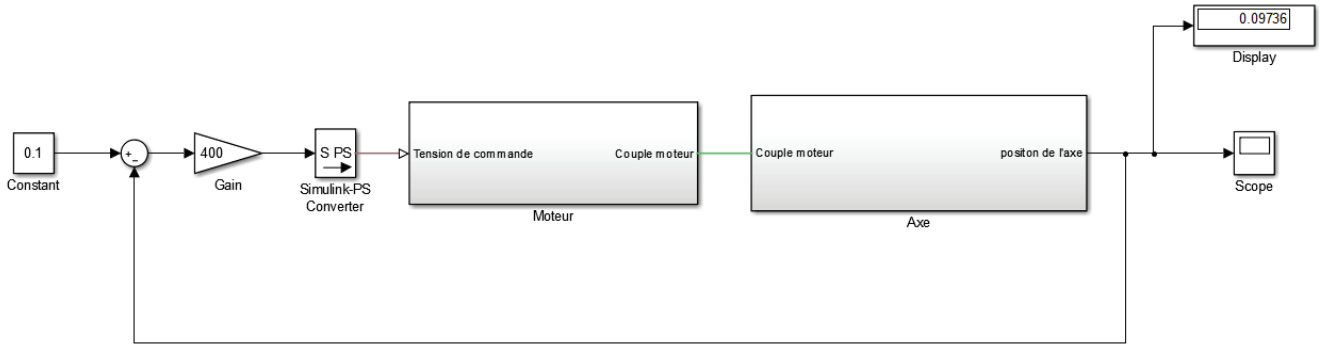


Figure 135 : modèle de l'asservissement en position linéaire de l'axe

On impose une consigne de 0.1 m. Cette consigne est comparée à la mesure en position réelle de l'axe. Un correcteur proportionnel, transforme l'écart en position en consigne de tension pour le moteur.

Seuls sont référencés dans le tableau de la Figure 136, les composants qui n'ont pas encore été utilisés auparavant dans le document.

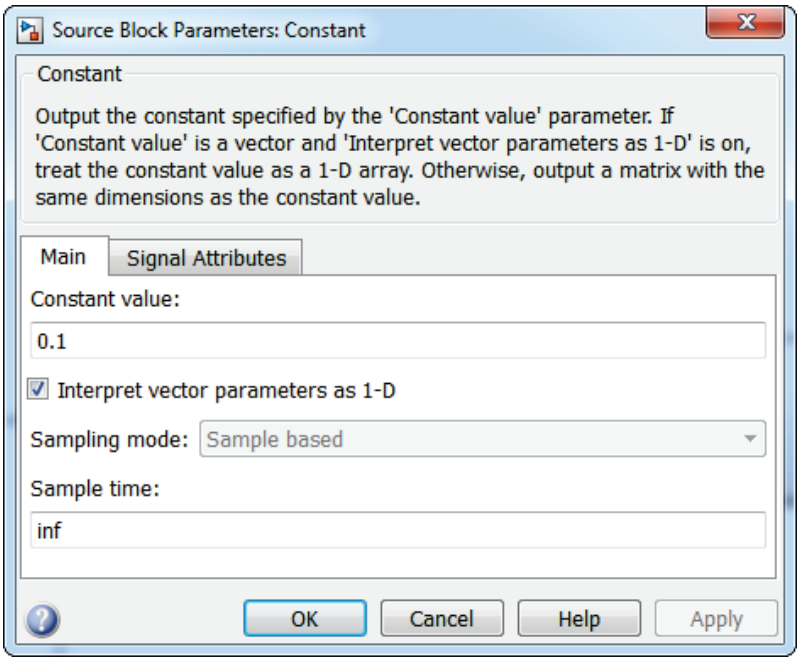
Fonction du composant	Représentation	Bibliothèque
Constante		Simulink/Sources ou Simulink/Commonly used Blocks
Gain		Simulink/Math Operations ou Simulink/Commonly used Blocks
Sommateur		Simulink/Math Operations ou Simulink/Commonly used Blocks
Afficheur		Simulink/Sinks

Figure 136 : composants à ajouter au modèle

Paramétrage

CONSTANT		Simulink/Sources ou Simulink/Commonly used Blocks

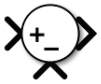
Ce bloc génère en sortie un signal numérique constant qui correspond ici à la consigne de position de 0.1 m.



Paramétrage

SUM  Simulink/Math Operations ou Simulink/Commonly used Blocks

Ce composant permet de réaliser un somme ou une différence de deux signaux. Le paramétrage correspond à la succesion de signes que l'on souhaite obtenir lors de l'opération. Indiquer dans le champs **List of Signs** la disposition des signes.

- `|-+` pour un sommateur de type 
- `-+|` pour un sommateur de type 

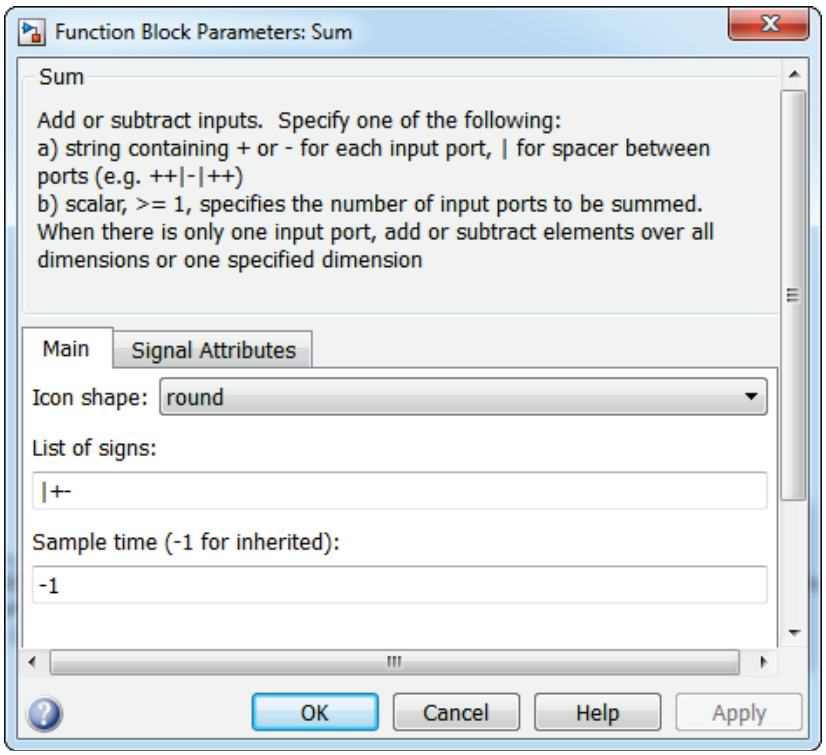
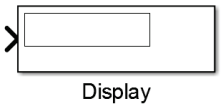


Figure 137 : paramétrage du bloc sommateur

Paramétrage

DISPLAY 

Simulink/Sinks

Ce bloc permet d'afficher une valeur en régime permanent pour visualiser ici la précision de la réponse en position.

Modifier le modèle ou ouvrir le fichier « **axe_lineaire_asservi.slx »**

Lancer la simulation en spécifiant un temps de simulation de **1 s**.

Visualiser la réponse en position à l'aide du scope.

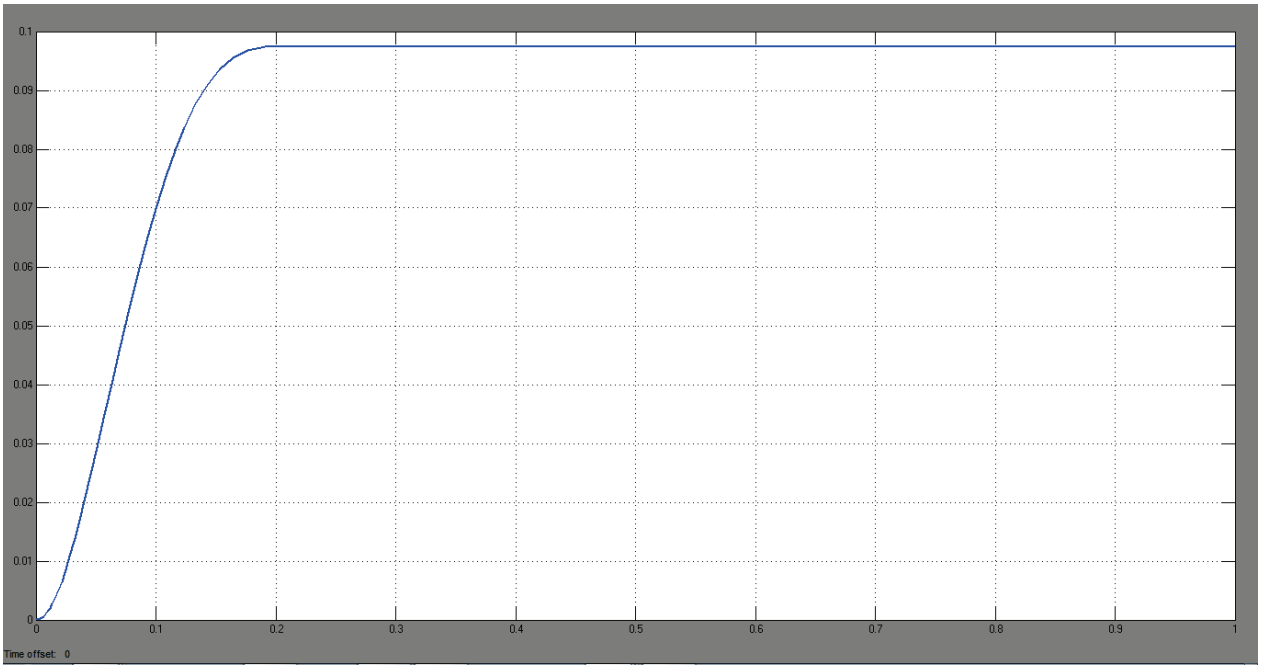


Figure 138 : réponse en position de l'axe asservi

On remarque que la position finale atteinte par l'axe est de 0.09736 m, ce qui nous permet d'évaluer que l'écart statique est de l'ordre de 3%. Il est possible d'améliorer les performances de l'asservissement en utilisant un correcteur PID.

Remplacer le bloc gain qui représente le correcteur proportionnel par le bloc PID.

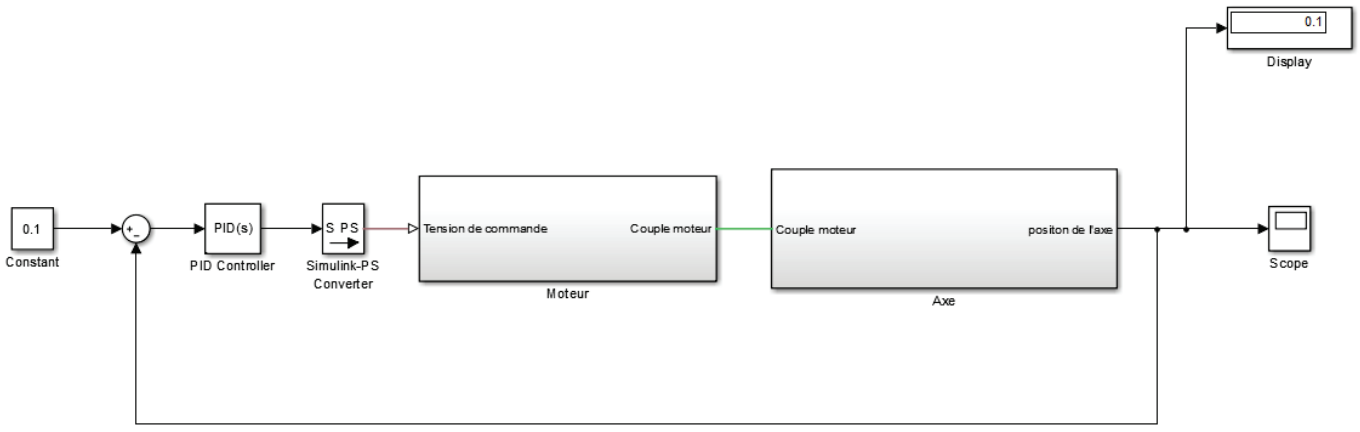
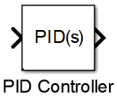


Figure 139 : modèle de l'asservissement en position de l'axe linéaire

PID Controller 

Simulink/Continues

Ce bloc permet de paramétrer un correcteur PID. Nous nous contenterons ici de mettre un gain proportionnel de 400 et un gain intégral de 140 pour annuler l'erreur statique. Nous reviendrons sur l'utilisation de ce bloc qui possède de nombreuses fonctionnalités très intéressantes pour le contrôle commande des systèmes.

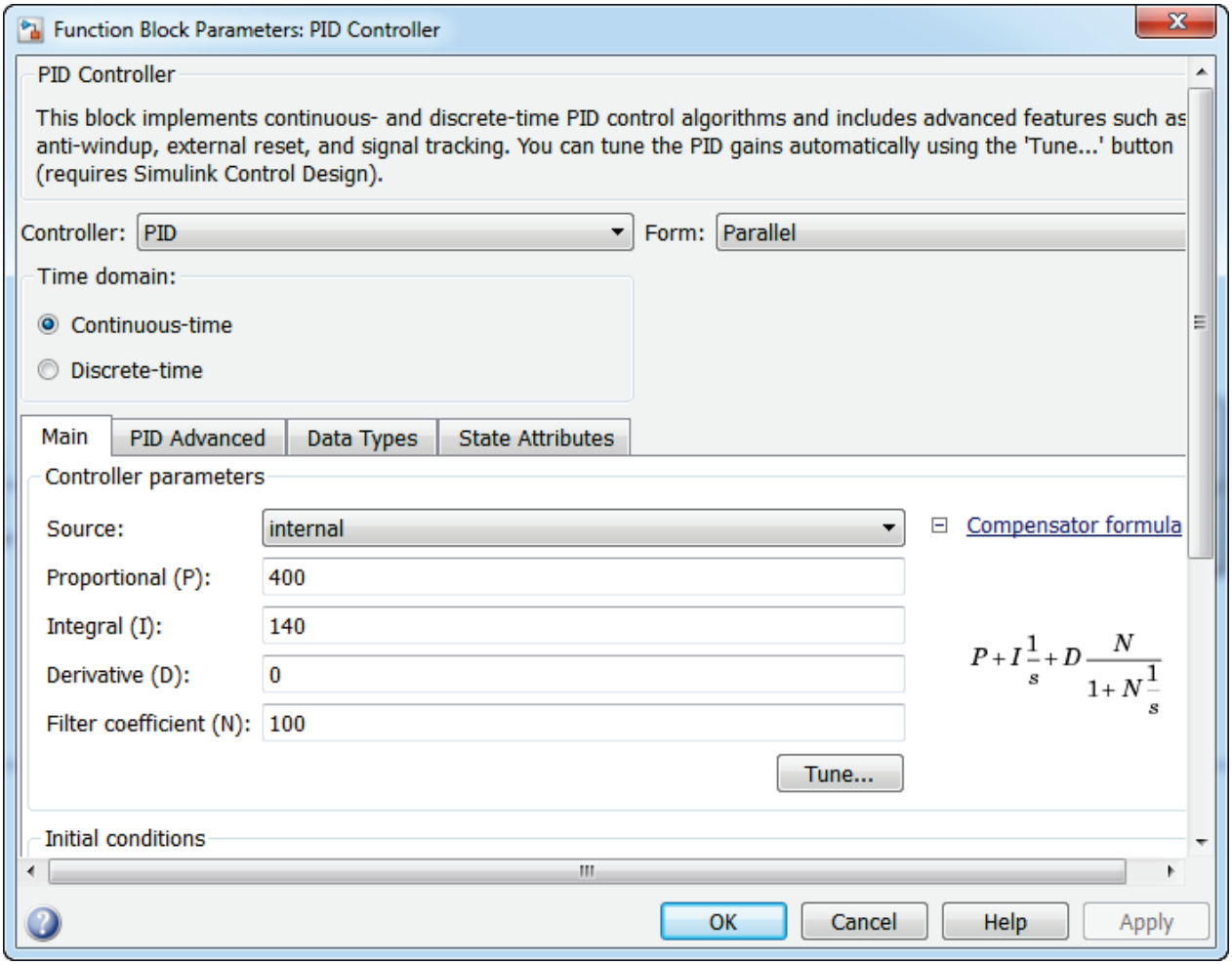


Figure 140 : paramétrage du bloc PID

Lancer la simulation.

Visualiser la réponse en position avec correction proportionnelle et intégrale.

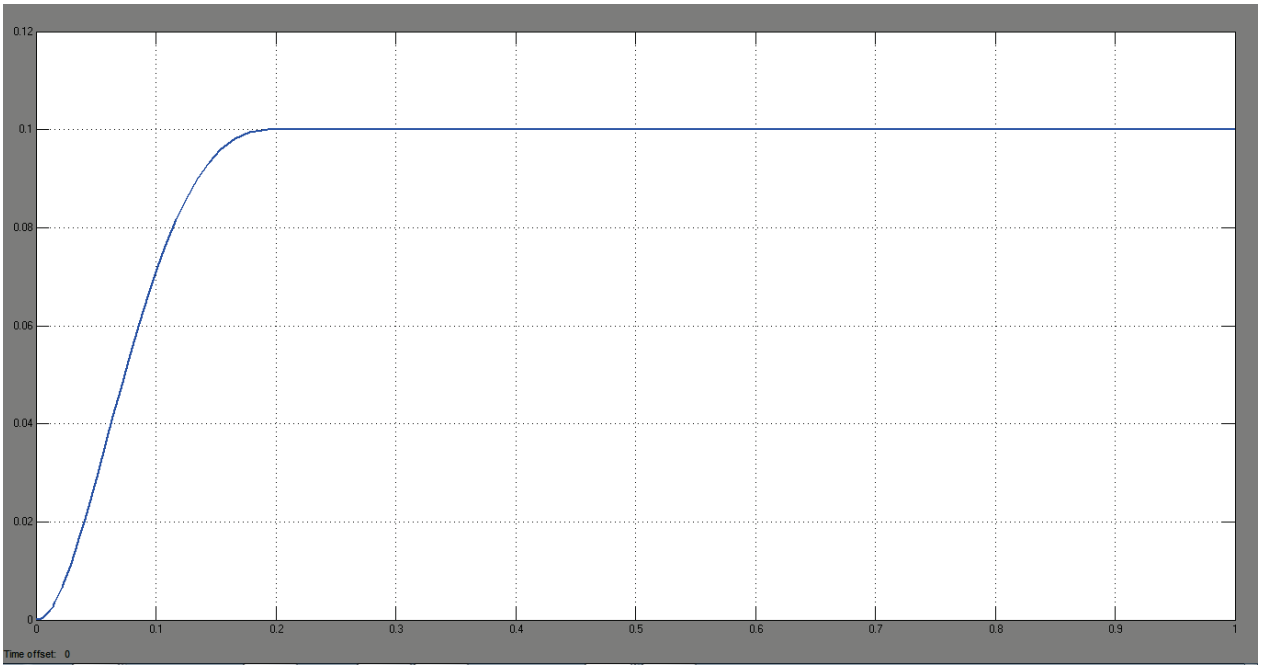


Figure 141 : réponse corrigé de l'asservissement en position de l'axe linéaire

On constate l'annulation de l'erreur statique en raison de la présence d'un terme intégrateur dans le PID.

La perturbation engendrée par le frottement sec est compensée par le correcteur PID pour une consigne de 0.1 m.

B. Domaines hydraulique-mécanique – vérin hydraulique simple effet

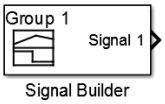
Le système modélisé est un vérin hydraulique simple effet commandé par un distributeur proportionnel 3/2 alimenté par une source de pression constante. Le déplacement de la tige entraîne le déplacement d'une masse qui agit sur un ensemble ressort+amortisseur. L'objectif est d'estimer l'évolution de la vitesse et de la position de la tige en fonction du temps.

Le système étudié fait intervenir 2 domaines physiques :

- Domaine hydraulique
- Domaine mécanique

1. Choix des composants

Seuls sont référencés dans le tableau de la Figure 142 les composants qui n'ont pas encore été utilisés auparavant dans le document.

Fonction du composant	Représentation	Bibliothèque
Source de signal paramétrable		Simulink/Sources

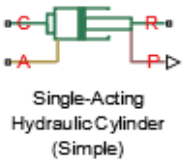
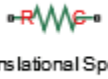
Source de pression hydraulique		Simscape/Fondation Library/Hydraulic/Sources
Distributeur proportionnel 3/2		Simscape/SimHydraulics/Valves/Directional Valves
Référence hydraulique		Simscape/Fondation Library/Hydraulic/Hydraulic elements
Propriétés du fluide		Simscape/SimHydraulics/Hydraulic Utillities
Vérin hydraulique simple effet		Simscape/SimHydraulics/Hydraulic Cylinders
Ressort en translation		Simscape/Fondation Library/ Mechanical/Translational Elements
amortisseur en translation		Simscape/Fondation Library/ Mechanical/Translational Elements

Figure 142 : les composants nécéssaires à la modélisation dela commande d'un vérin hydraulique

Le modèle Simscape du vérin hydraulique simple effet est représenté sur la Figure 143 et permet de visualiser les différents domaines physiques qui interviennent dans le modèle. La correspondance entre les connexions du domaine physique et les éléments réels est donnée.

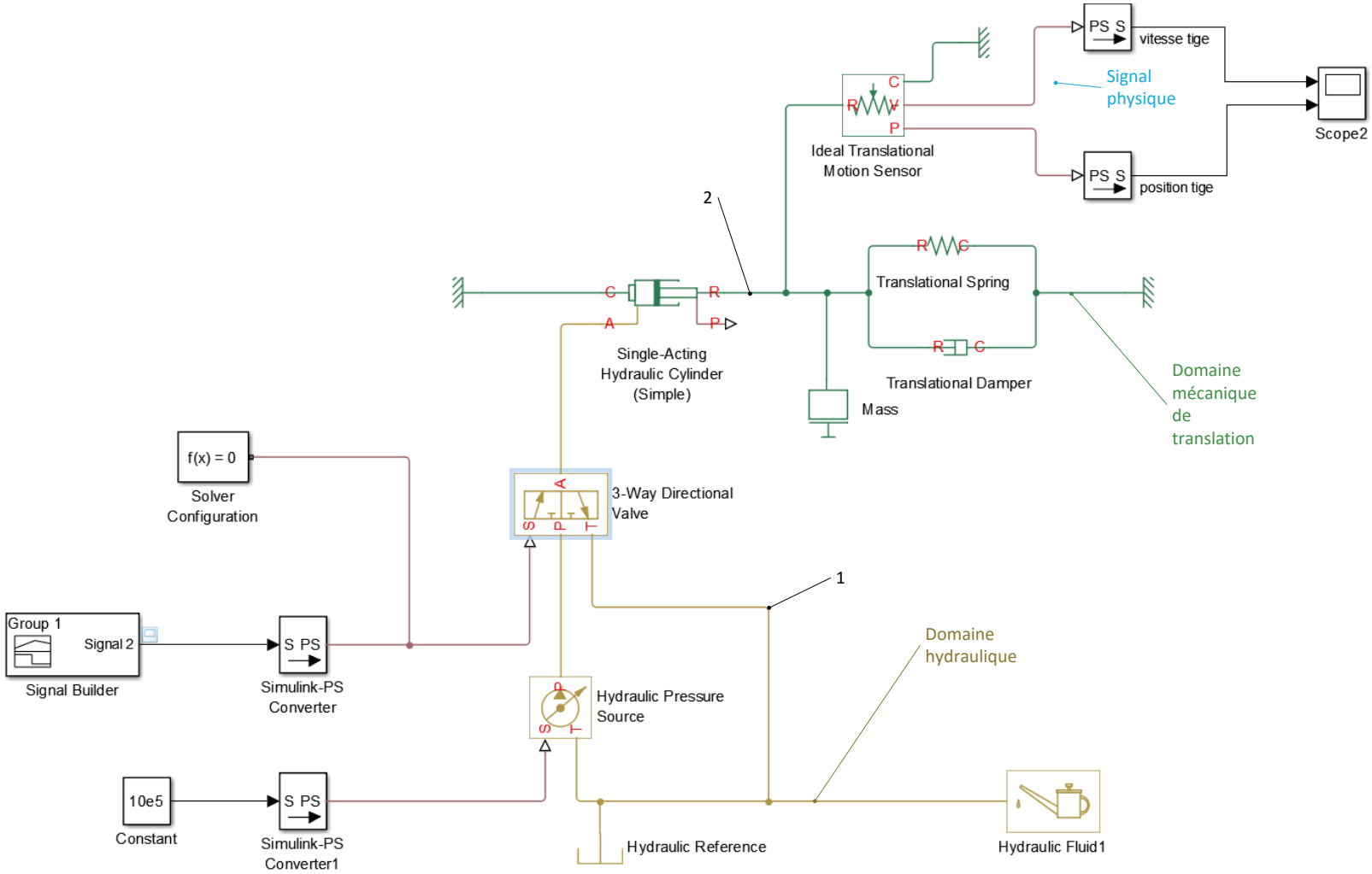


Figure 143 : visualisation des domaines physiques intervenant dans la modélisation de la commande du vérin hydraulique

1. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, exécuter la commande **File/New/Model** pour créer un nouveau fichier.

Glisser/Déposer les différents blocs à partir des bibliothèques, les disposer dans la fenêtre de travail et les relier comme pour obtenir la configuration de la Figure 144.

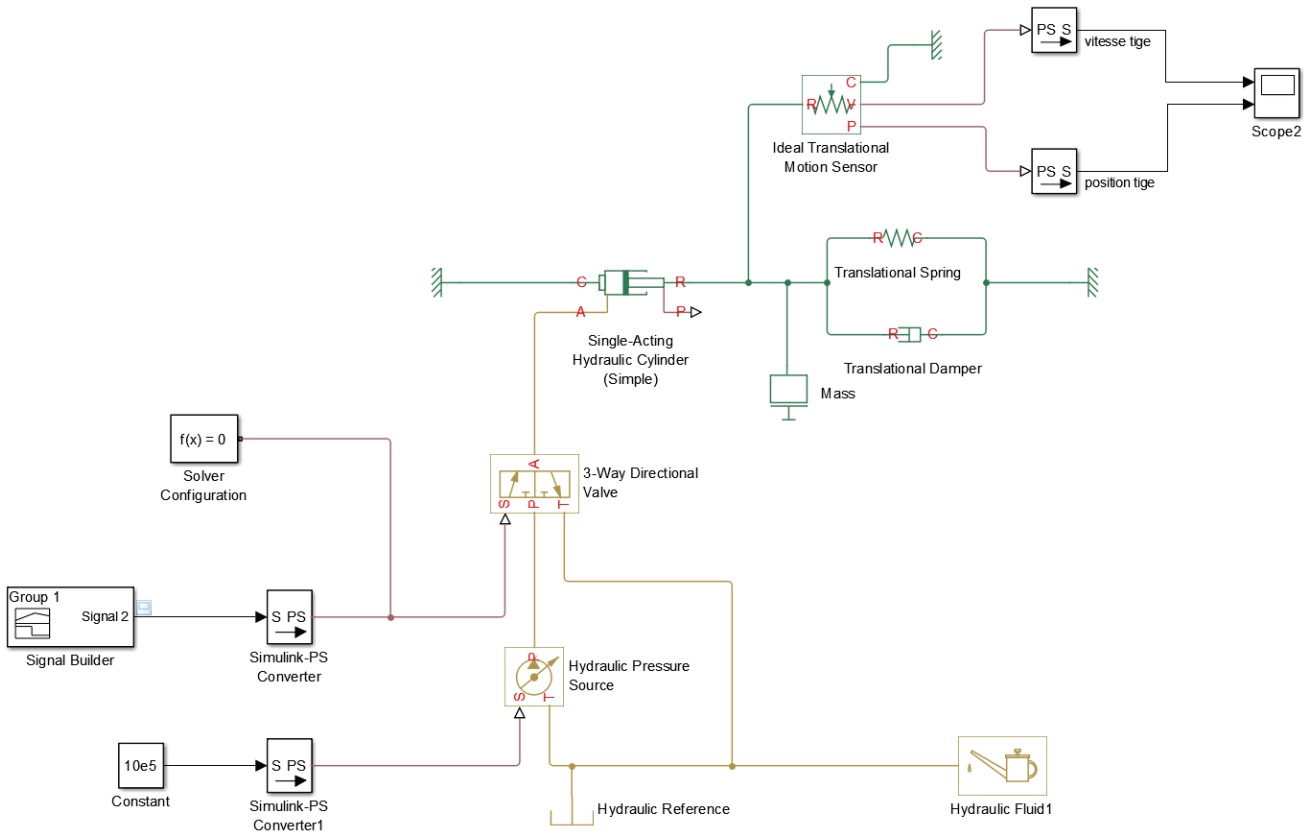
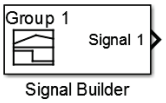


Figure 144 : modèle Simscape de la commande du vérin hydraulique

2. Paramétrage des composants

Effectuer le paramétrage des différents blocs conformément aux informations données ci-dessous.

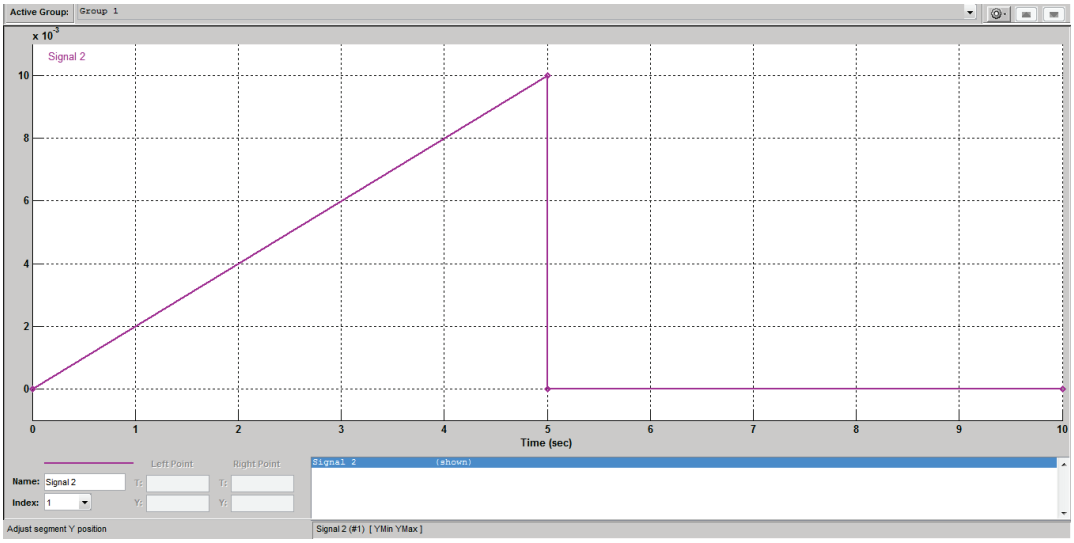
Paramétrage

Signal Builder 

Simulink/Sources

Ce bloc permet de créer un signal de commande quelconque. Nous souhaitons ici commander le distributeur de manière linéaire de la valeur 0 à la valeur 0.01 correspondant à l'ouverture maximale de l'orifice d'alimentation de l'actionneur. A partir de t=5 s, le signal de commande est ramené à 0.

Se reporter à l'annexe « **utilisation du signal builder** » pour paramétrer le signal.

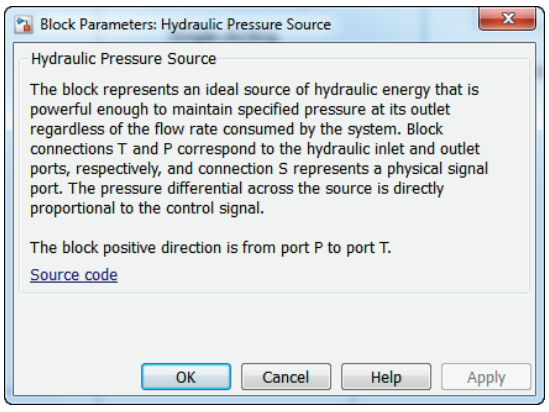


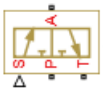
Paramétrage

Hydraulic Pressure Source 

Simscape/Fondation Library/Hydraulic/Sources

Ce composant permet de modéliser une source de pression parfaite. La pression sera constante quelles que soient les conditions de fonctionnement. Il possède 2 ports (**T** et **P**) de type **PCP** du domaine hydraulique et un port de type signal physique qui indique la valeur de la pression. Ici la pression sera réglée à 10^6 Pa. Le bloc « **Constant** » alimente cette source de pression par l'intermédiaire d'un bloc **S-PS** dont l'unité sera réglée sur **Pa**.

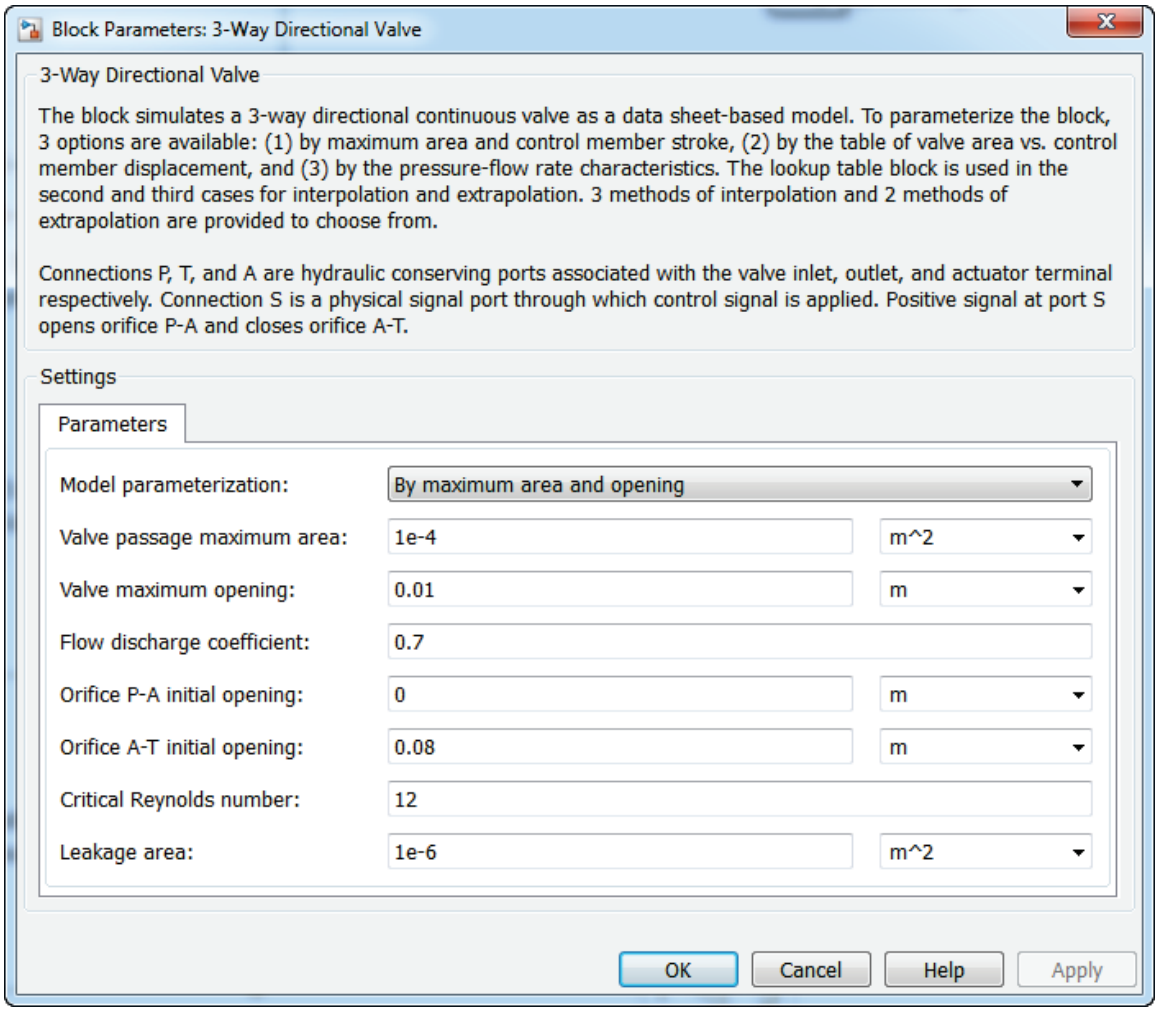


3 – Way Directional Valve  Simscape/SimHydraulics/Hydraulic Cylinders

Ce composant permet de modéliser un distributeur hydraulique 3/2 à commande proportionnelle. Il possède 3 ports de type **PCP** (**T**, **P** et **A**) du domaine hydraulique et un port **S** de type signal physique qui permet de commander le distributeur.

- Port A : sortie vers actionneur
- Port P : alimentation
- Port T : échappement

Le paramétrage peut se faire de différentes manières en fonction des données constructeur disponibles (onglet **Model Parameterization**). Ici le paramétrage est fait à partir de **By maximum area and opening**.



Valve passage maximum area : indique la section maximale de passage du fluide

Valve maximum opening : indique la valeur de la commande qui arrive sur le port S et qui donnera l'ouverture maximale de la valve.

Flow discharge coefficient : coefficient empirique qui dépend des formes de la servo-valve et qui est donné dans les données constructeur (valeur par défaut 0.7)

Orifice P-A initial opening (h_{PA0}) : ouverture initiale de l'orifice P-A

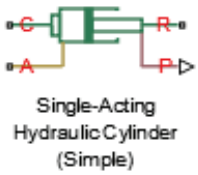
Orifice A-T initial opening (h_{AT0}) : ouverture initiale de l'orifice A-T

Si on appelle x la commande qui arrive sur le port S, les ouvertures respectives des orifices seront :

$$h_{PA} = h_{PA0} + x$$
$$h_{AT} = h_{AT0} - x$$

Paramétrage

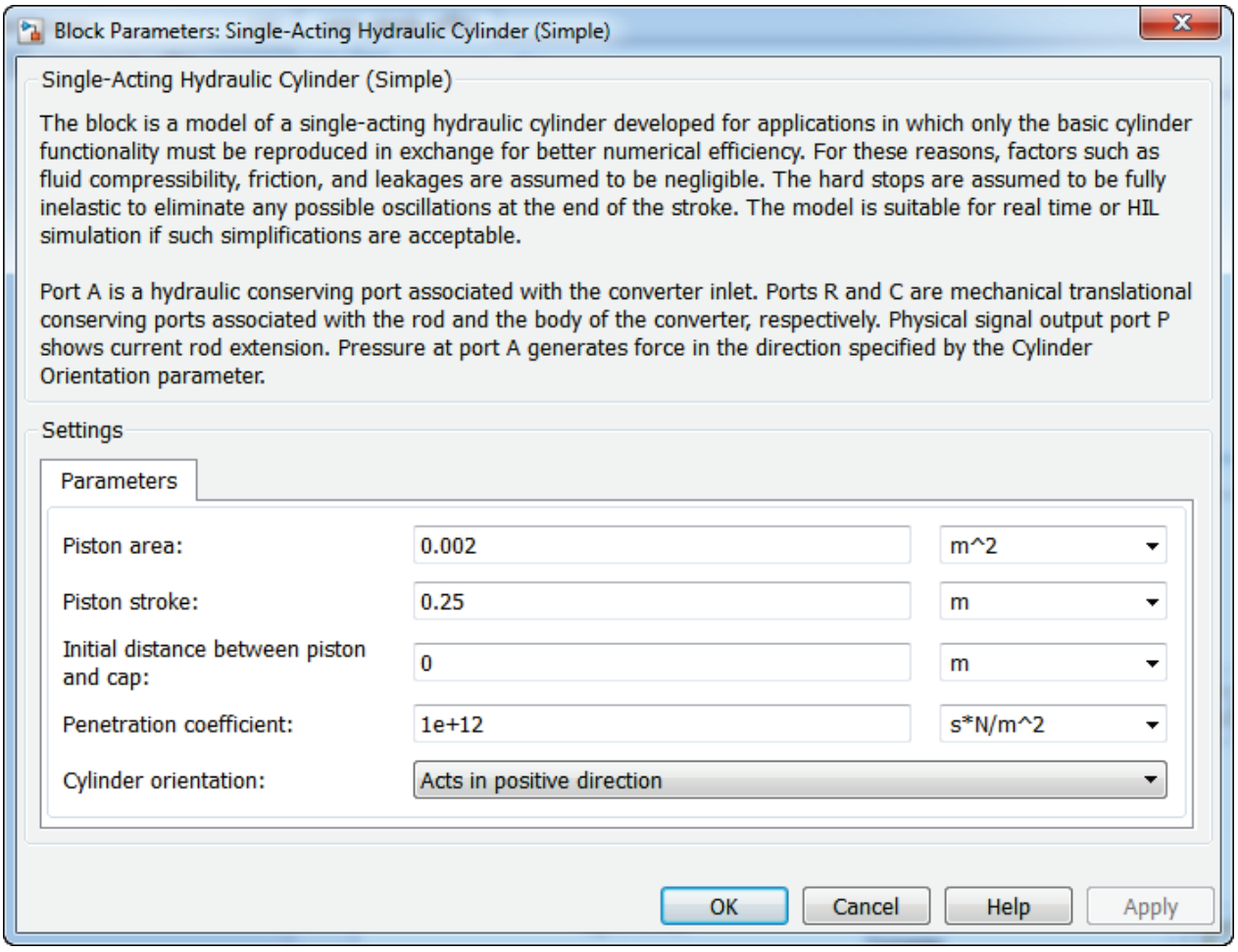
Single-Acting Hydraulic Cylinder (Simple)



Simscape/SimHydraulics/Valves/Directional Valves

Ce composant permet de modéliser un vérin hydraulique simple effet. Il possède 3 ports de type **PCP,** un du domaine hydraulique (**A**) et deux du domaine mécanique de translation (**C** et **R**)

- Port A : orifice d'alimentation hydraulique du vérin
- Port C : associé au corps du vérin
- Port R : associé à la tige du vérin



Piston area : surface du piston

Piston stroke : course du piston

Initial distance between piston an d cap : position initiale de la tige du vérin (0 si le vérin est complètement rentré au début de la simulation).

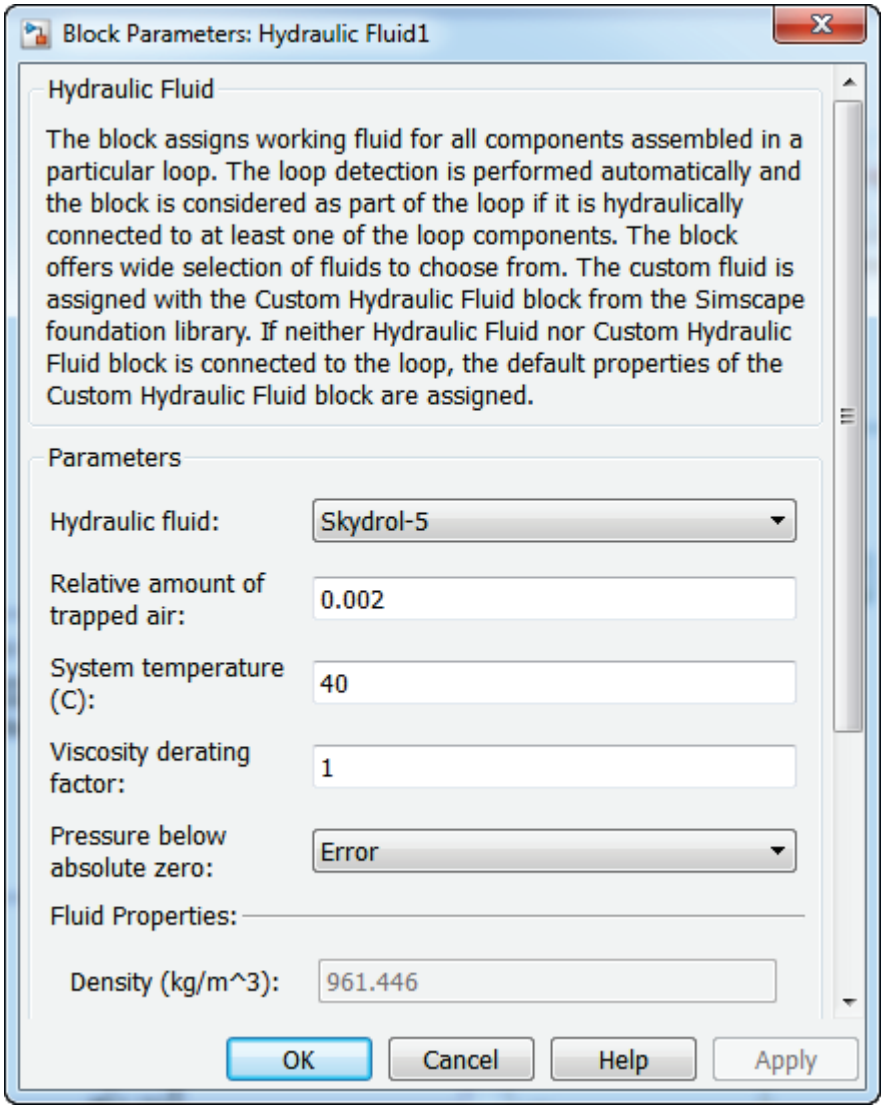
Cylinder orientation : permet de choisir l'orientation de l'effort généré en bout de tige.

Paramétrage

Hydraulic fluid 

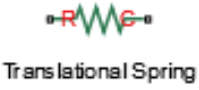
Simscape/SimHydraulics/Hydraulic Utillities

Ce composant permet de modéliser les propriétés du fluide utilisé.
Un bloc du même type existe dans la bibliothèque Simscape/Foundation Library/Hydraulics/Hydraulics Utilities qui permet de rentrer manuellement les caractéristiques du fluide si cela est nécessaire.



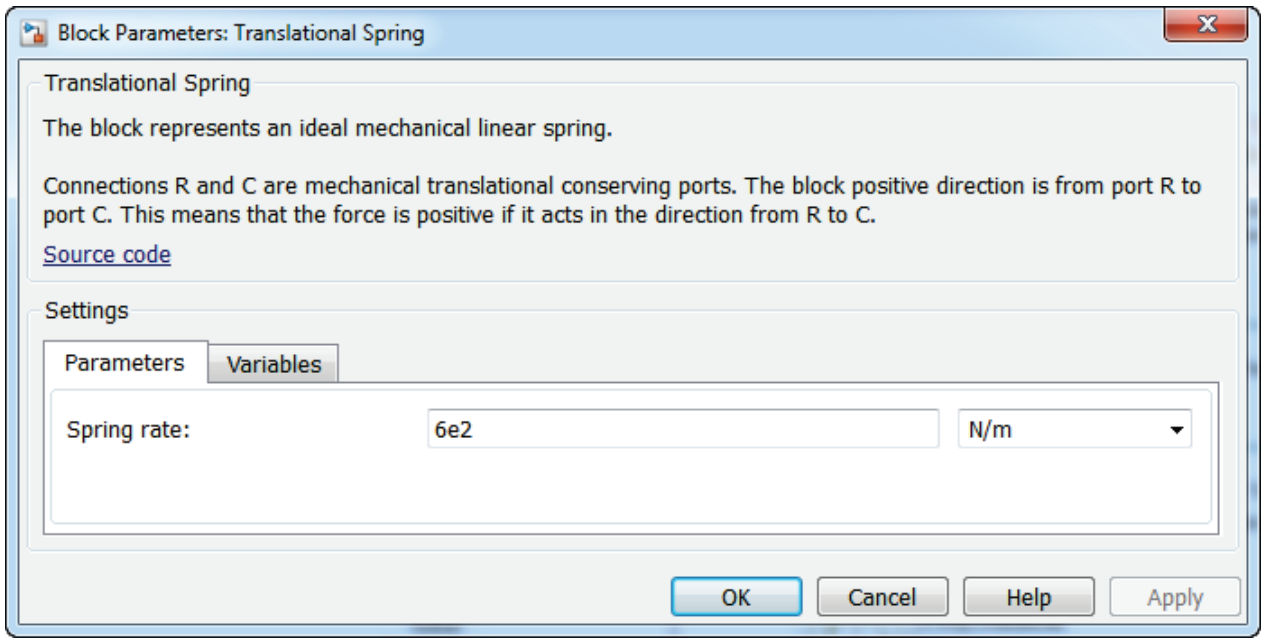
Paramétrage

Translational Spring



Simscape/Fondation Library/
Mechanical/Translational Elements

Ce composant modélise un ressort linéaire, il faut spécifier la raideur et l'allongement initial.

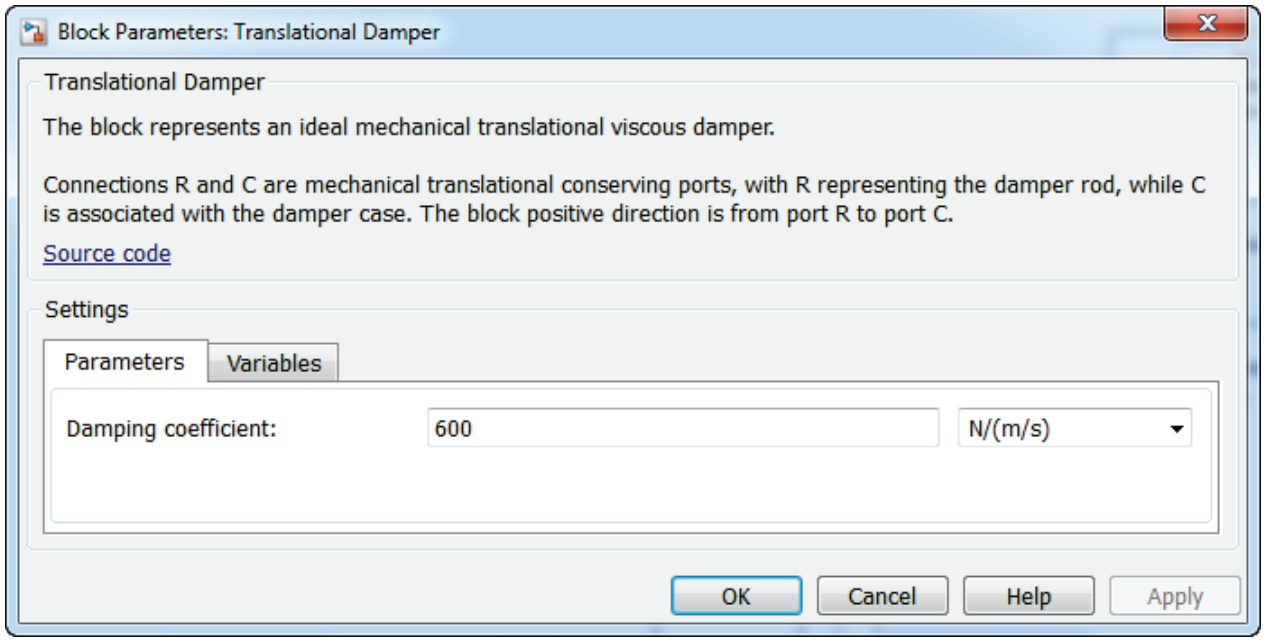


Paramétrage

amortisseur en translation

Simscape/Fondation Library/
Mechanical/Translational Elements

Ce composant modélise un amortisseur, il faut spécifier le coefficient d'amortissement.



Scope  Simulink/Sinks

Pour paramétrer un Scope à deux entrées, se reporter à l'annexe paramétrage des **Scopes**.

Le fichier contenant le modèle paramétré est disponible sous le nom **verin_simple_effet_pression.slx**.

3. Simulation

Choisir le solveur **ode23t**.

Lancer la simulation en spécifiant un temps de simulation de **10 s**.

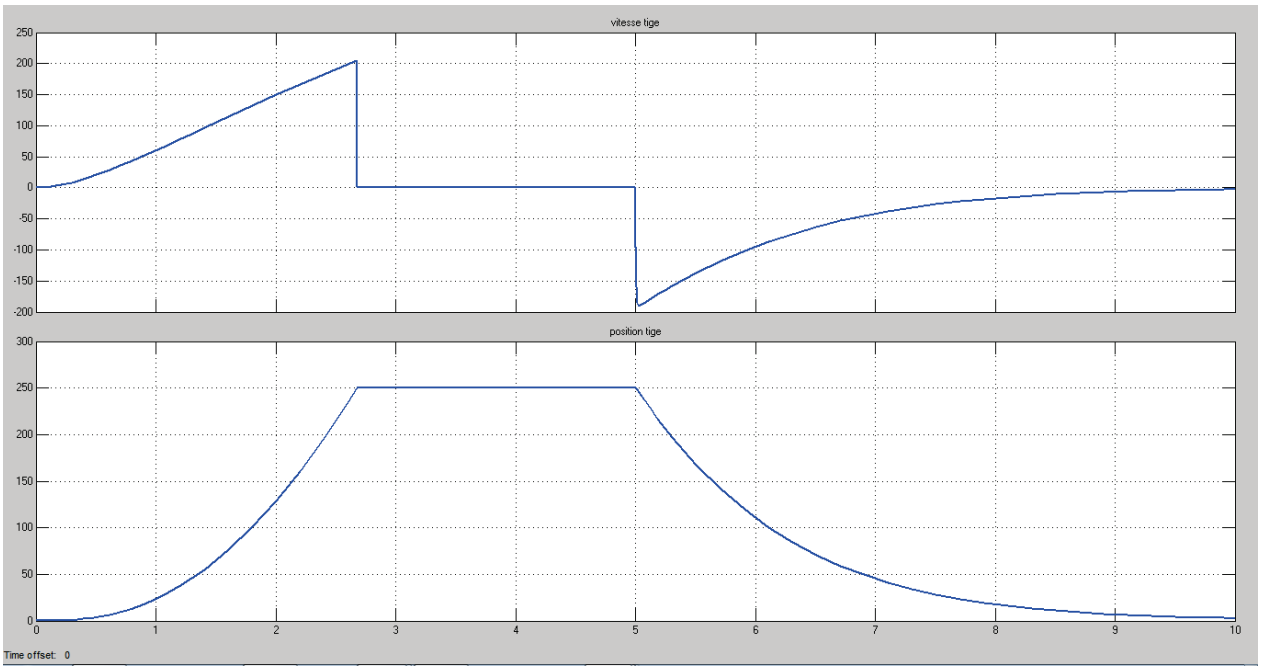


Figure 145 : évolution de la vitesse et de la position de la tige du vérin

On observe sur la Figure 145 que la tige du vérin sort, jusqu'à atteindre sa course maximale à t=3s. La tige est à l'arrêt jusqu'à la disparition de la commande à t=5s. La tige revient alors en position initiale sous l'action du ressort.

4. Utilisation des fonctionnalités de routage des signaux

Afin de simplifier la visualisation des résultats, il est parfois utile d'avoir recours aux fonctionnalités de routage des signaux.

Ouvrir le fichier **verin_simple_effet_tag.slx.**

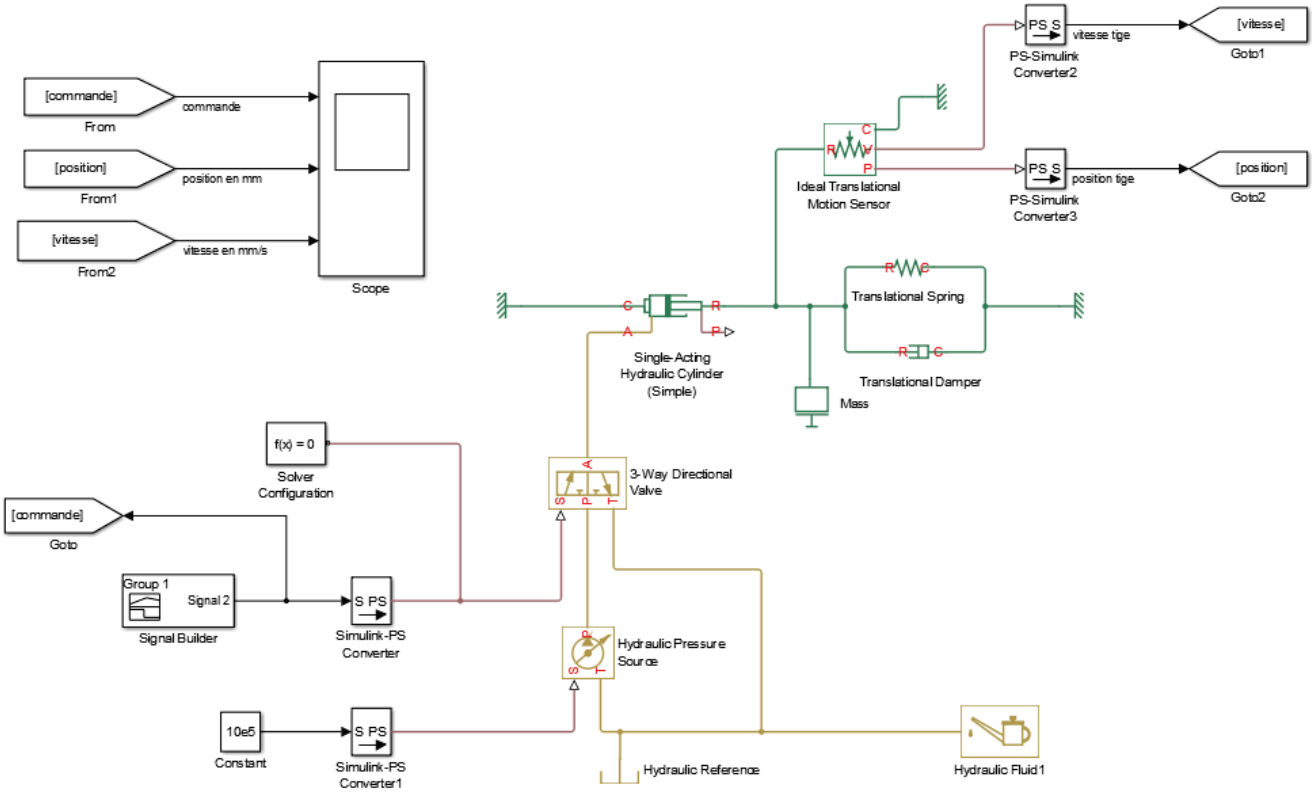
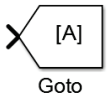


Figure 146 : modèle Simscape de la commande du vérin hydraulique avec routage des signaux

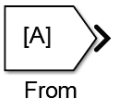
Le modèle est celui de la commande du vérin hydraulique simple effet. Les signaux relevés sur le modèle sont ici visualisés à l'aide d'un scope unique. Pour éviter aux connexions de traverser inutilement le modèle, il est possible de router les signaux à l'aide des blocs **Goto** et **From**.

Goto 

Simulink/Signal Routing

Ce bloc permet de mettre un **Tag** sur un signal et de récupérer ce signal à l'aide d'un bloc **From**.

Goto tag : indique le nom que l'on donne au signal
Tag visibility : indique si le **Tag** est **Local** (uniquement exploitable pour le sous-système dans lequel il est créé) ou **Global** (exploitable dans tout le système)

From 

Simulink/Signal Routing

Ce bloc permet de récupérer le Tag d'un signal créé à l'aide d'un bloc **Goto**.

Le choix du Tag se fait directement à partir du menu déroulant. La commande **Update Tags** permet de rafraîchir les Tags afin de faire apparaître tous les Tags dans le menu déroulant.

5. Remplacement de la source de pression par une source de débit

Il est possible de remplacer la source de pression par une source de débit. Dans ce cas, il faudra prévoir impérativement une soupape de décharge afin que le fluide puisse s'écouler vers le réservoir lorsque la tige du vérin est en butée (Figure 147).

Ouvrir le fichier **verin_simple_effet_débit.slx.**

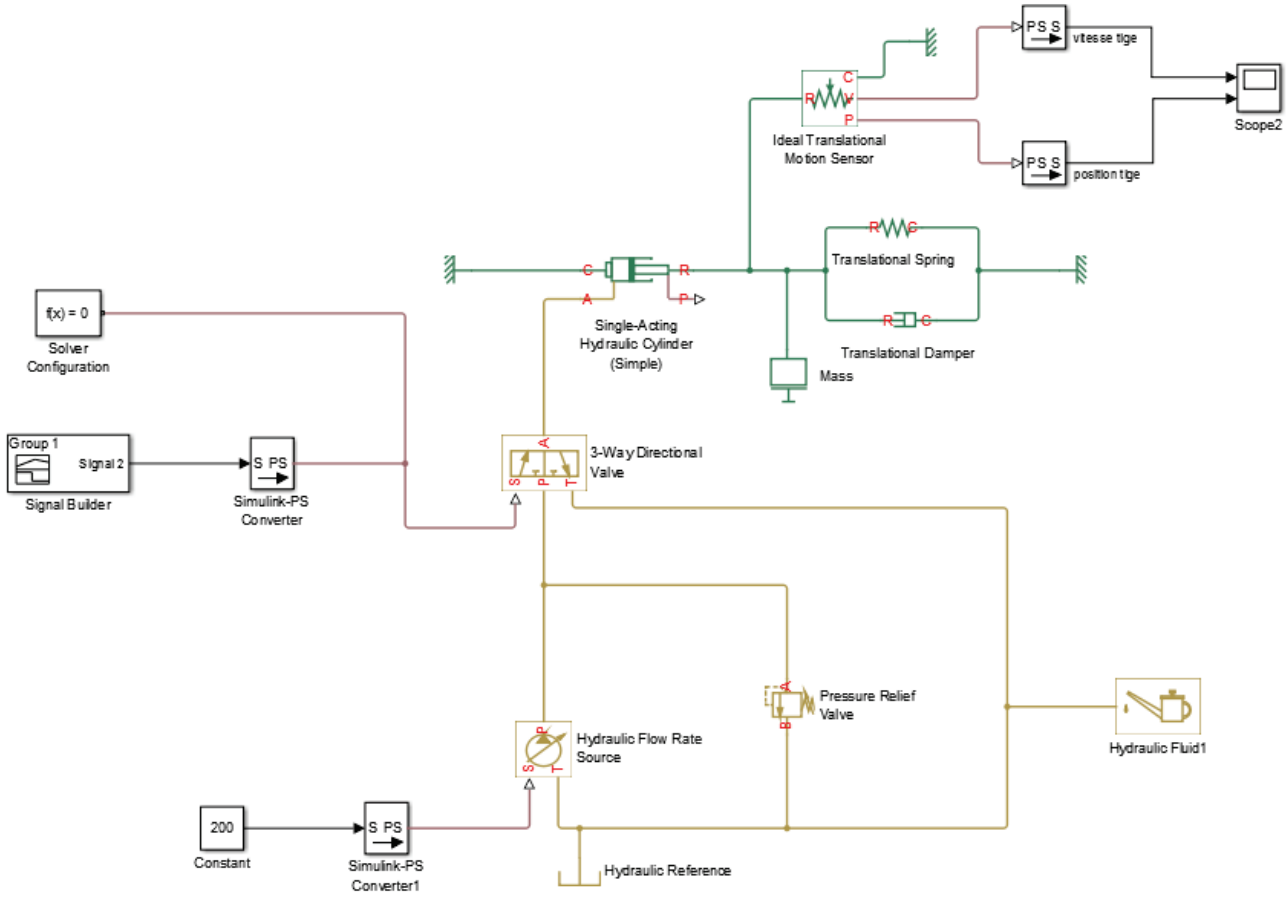


Figure 147 : modèle Simscape de la commande d'un vérin hydraulique avec source de débit

Le modèle est le même que précédemment. La source de pression a été remplacée par une source de débit de 200 l/min. Une soupape de décharge a été placée sur la canalisation d'alimentation afin de limiter la pression lorsque le vérin arrive en butée.

Paramétrage

Hydraulic Flow Rate Source		Simscape/Fondation Library/ Hydraulic/ Sources

Ce composant permet de modéliser une source de débit parfaite. Le débit sera constant quelles que soient les conditions de fonctionnement. Il possède 2 ports (**T** et **P**) de type **PCP** du domaine hydraulique et un port de type signal physique qui indique la valeur du débit. Ici le débit sera réglé à 200 l/min. Le bloc « **Constant** » alimente cette source de débit par l'intermédiaire d'un bloc **S-PS** dont l'unité sera réglée sur **l/min**.

Pressure Relief Valve

Simscape/SimHydraulics/Valves/Pressure Control Valves

Ce composant modélise un limiteur de pression.

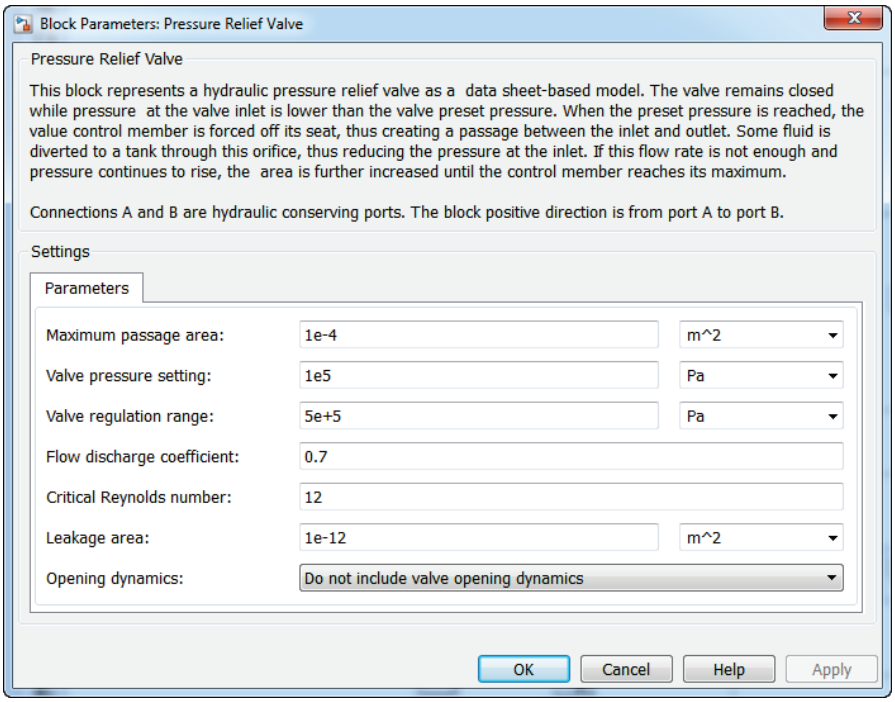


Figure 148 : paramétrage d'un bloc limiteur de pression

Maximum passage area : indique la surface maximale de passage du fluide
Valve pressure setting $\left(p_{set}\right)$: indique la valeur de la pression à partir de laquelle la valve commence à s'ouvrir.
Valve regulation range $\left(p_{reg}\right)$: indique la variation de pression dans la valve entre la position ouverte et la position fermée.

La courbe de la Figure 149 montre la variation de l'ouverture de la valve en fonction de la pression.

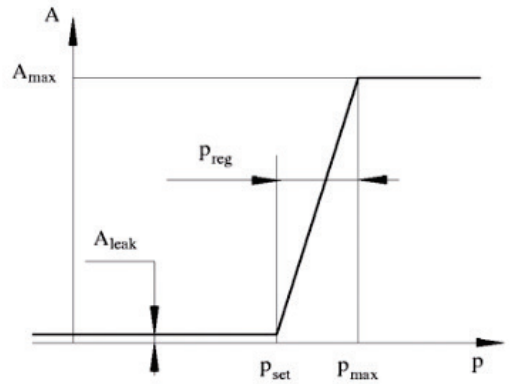


Figure 149 : variation de l'ouverture de soupape en fonction de pression

$$p_{\max} = p_{set} + p_{reg}$$

Lancer la simulation et observer l'évolution de la vitesse et de la position de la tige du vérin.

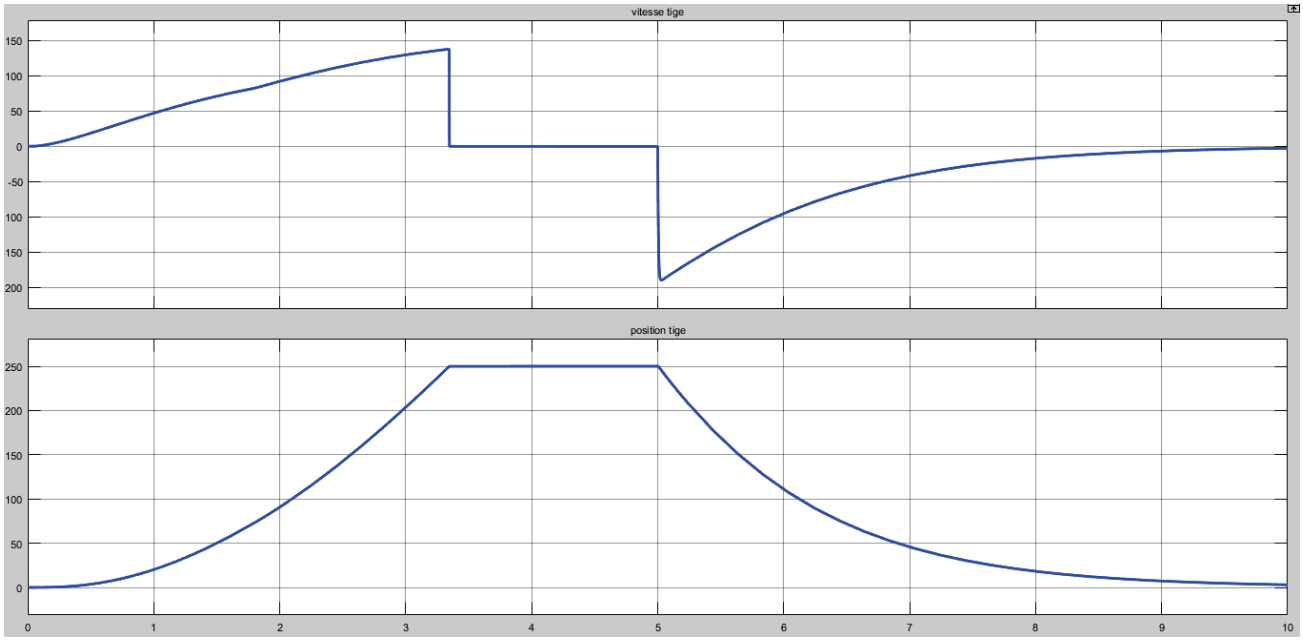


Figure 150 : évolution de la vitesse et de la position de la tige du vérin

On observe que la tige met environ 3.3 s pour atteindre la butée.

Augmenter la valeur de la **Valve pressure setting** dans le limiteur de pression **1e6 Pa** et relancer la simulation.

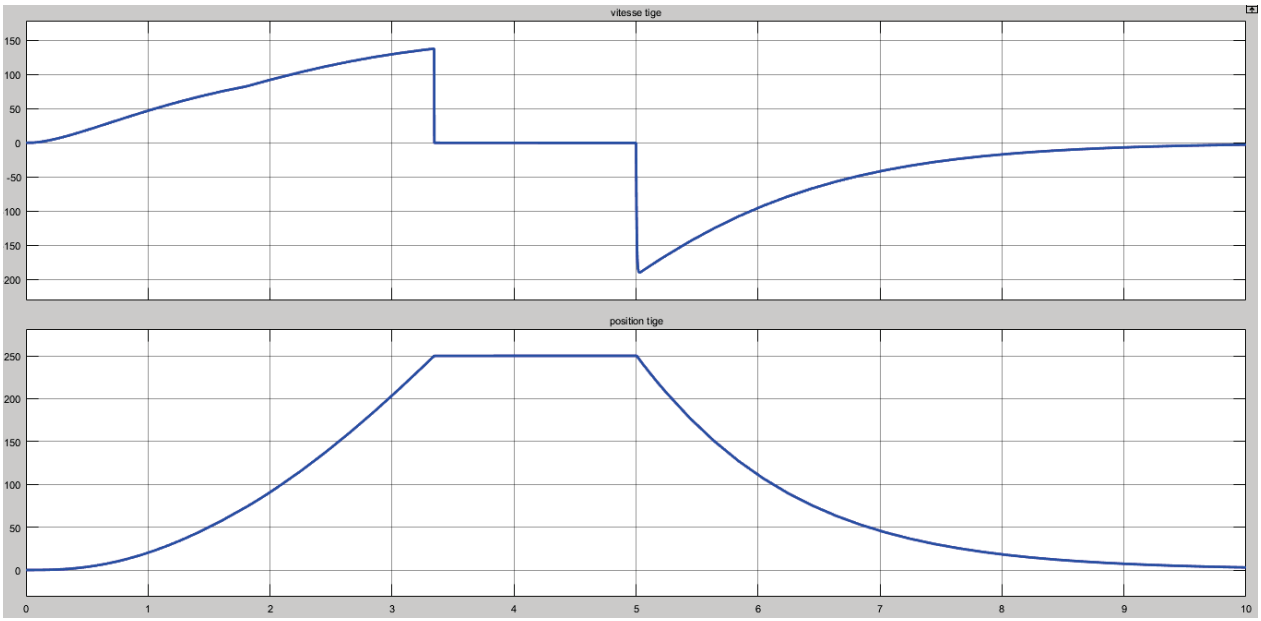


Figure 151 : évolution de la vitesse et de la position de la tige du vérin

On observe bien que l'augmentation de la pression admissible dans la canalisation d'alimentation du vérin permet une sortie de tige plus rapide. Le vérin arrive en butée au bout de 2.4 s.

C. Domaine électrique –Commande PWM d'un moteur à courant continu

Dans cette partie nous apprendrons à utiliser les composants nécessaires à la conception d'une commande PWM d'un moteur à courant continu.

Une tension de commande PWM est caractérisée par sa fréquence, son rapport cyclique (duty cycle) et son amplitude.

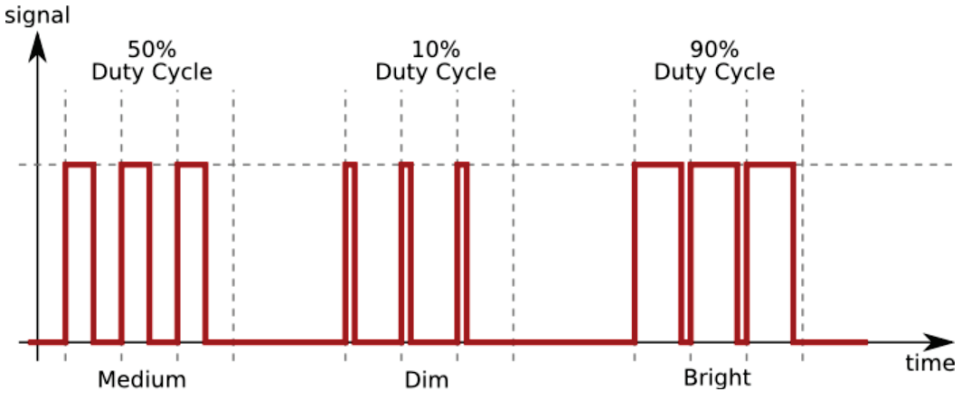
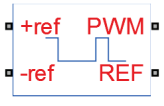
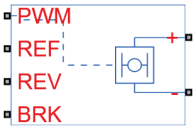


Figure 152 : principe de commande PWM

Pour générer cette tension nous utiliserons le composant « **Controlled PWM Voltage Source** ».

Pour inverser le sens de rotation du moteur ou le stopper nous utiliseront le composant « **H-Bridge** » (pont en H).

Ces composants sont issus de la bibliothèque **SimElectronics** :

Fonction du composant	Représentation	Bibliothèque
Génération d'une tension de commande de type PWM	 Controlled PWM Voltage	Simscape/SimElectronics/Actuators and Drivers/Drivers
Pont en H	 H-Bridge	Simscape/SimElectronics/Actuators and Drivers/Drivers

1. Utilisation du composant « Controlled PWM Voltage »

Ouvrir le fichier **bloc_PWM.slx.**

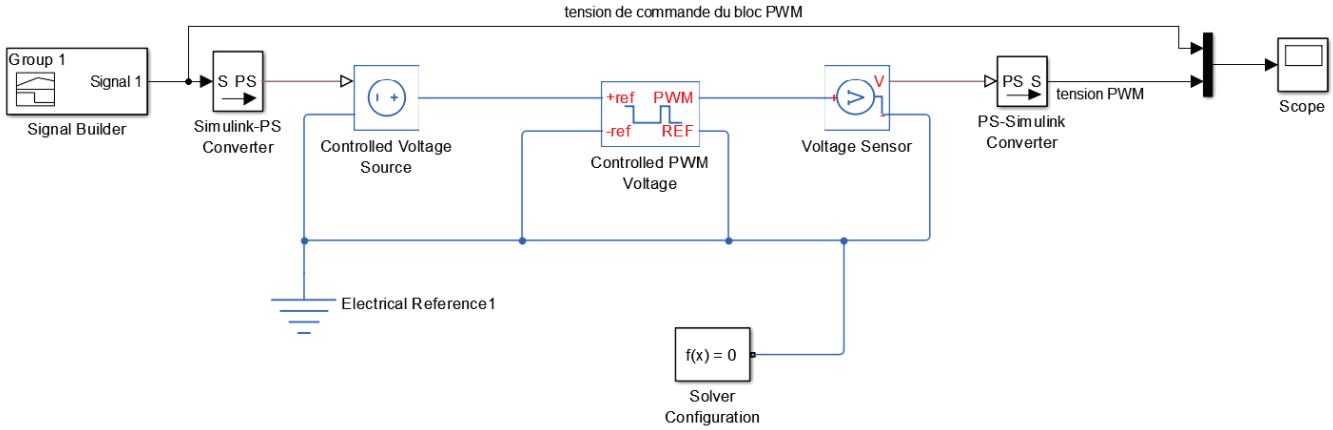


Figure 153 : Illustration du fonctionnement du bloc « Controlled PWM Voltage »

Double-cliquer sur le **Signal Builder** pour observer l'allure de la tension imposée au port +ref (Figure 154).

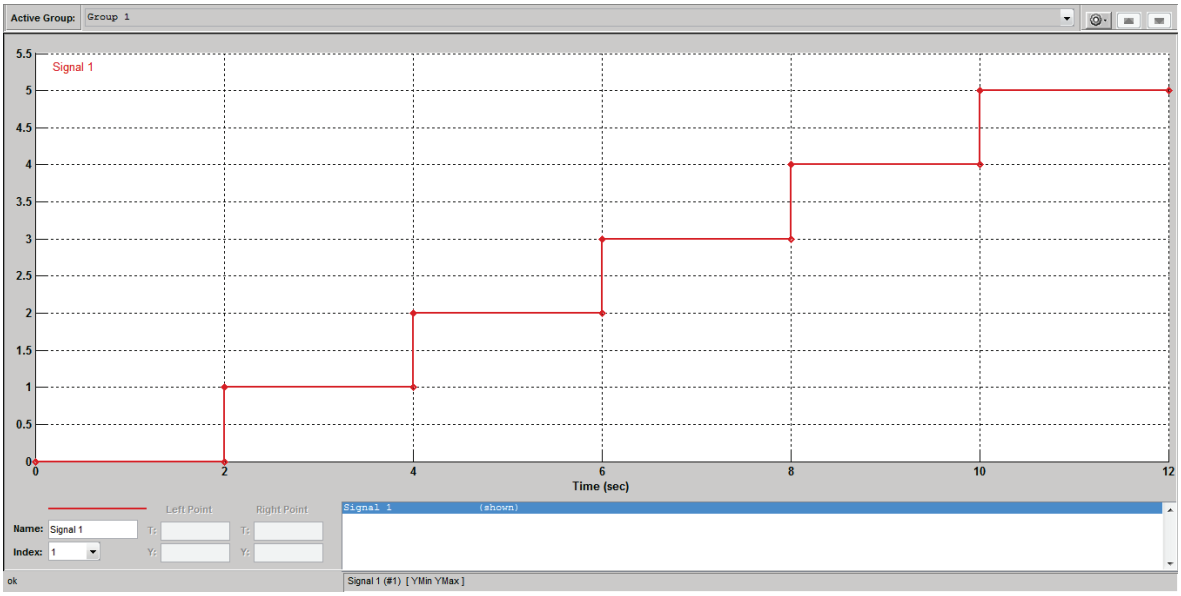


Figure 154 : tension de commande du bloc Controlled PWM Voltage

Cette tension varie de 0 V à 5 V et imposera le rapport cyclique du signal PWM.

Lancer la simulation et observer le résultat sur le scope.

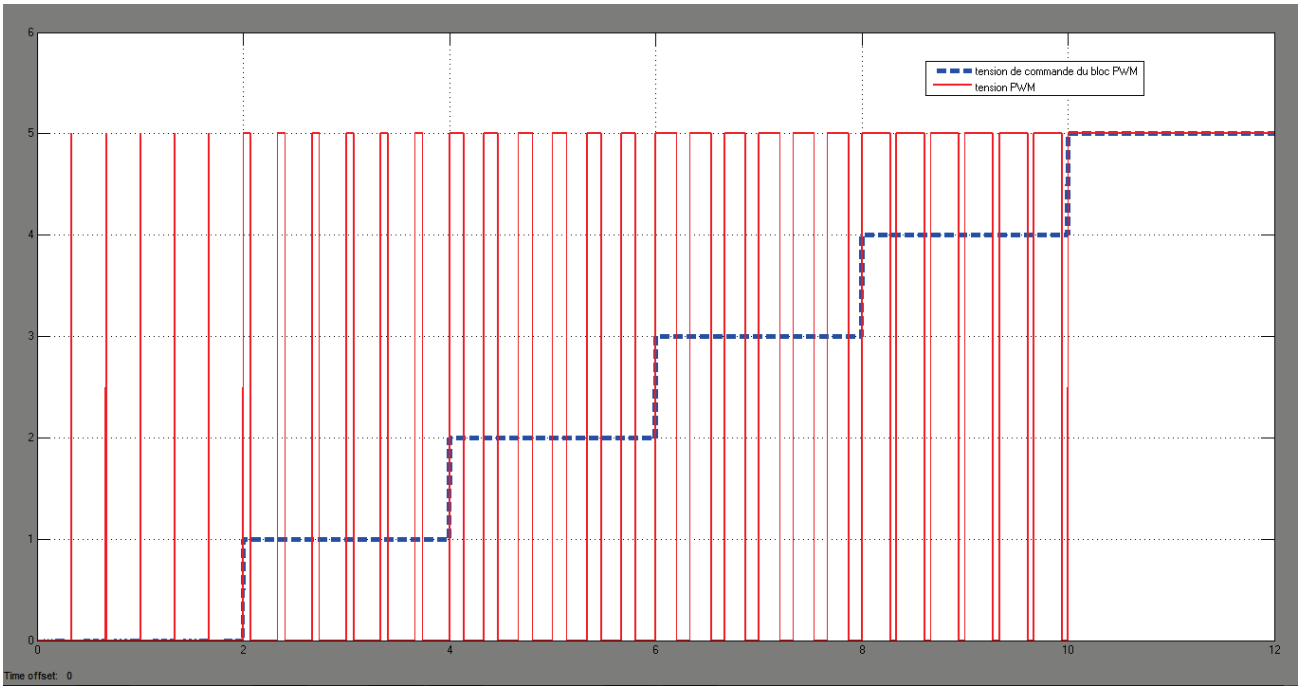


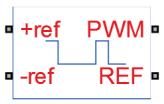
Figure 155 : signal PWM en fonction de la tension de commande

En bleu apparaît le signal de commande imposé sur le port **+ref**, en rouge la tension PWM qui sort sur le port **PWM**.

La fréquence du signal PWM dépend de l'amplitude de la tension **+ref.**

La fréquence et l'amplitude sont imposées dans le paramétrage du composant.

Paramétrage

Controlled PWM Voltage Source	 Controlled PWM Voltage	Simscape/SimElectronics/Actuators and Drivers/Drivers

Ce composant modélise un générateur de tension PWM. Il possède 4 ports de type **PCP** du domaine électrique. Il permet de générer une tension PWM à partir des informations prélevées sur les ports d'entrée et du paramétrage du composant.

- **+ref** : cette tension sera l'image du rapport cyclique du signal PWM.
- **-ref** : référence de la tension +ref(souvent relié à la masse)
- **PWM** : signal de tension PWM dont le rapport cyclique dépend de l'entrée +ref. L'amplitude et la fréquence sont spécifiées dans les paramètres du composant.
- **REF** : référence de la tension du signal PWM (souvent relié à la masse)

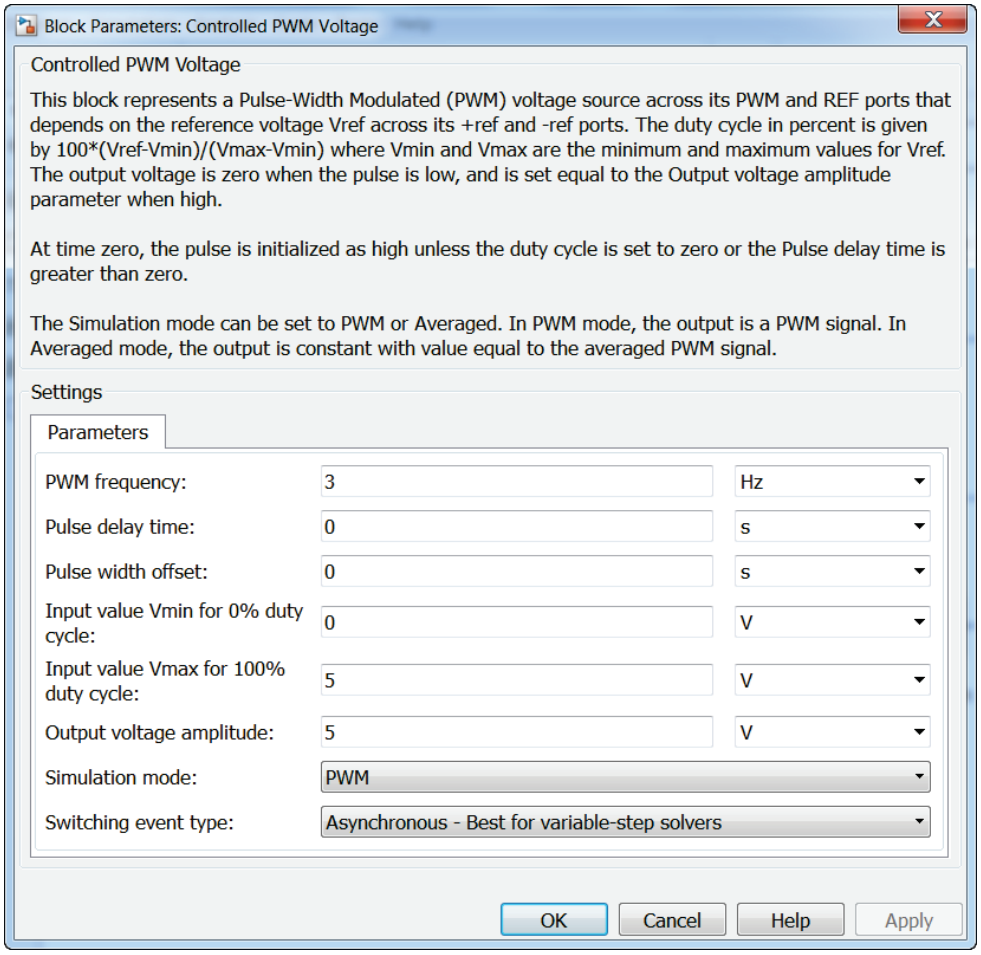


Figure 156 : paramétrage du bloc Controlled PWM Voltage Source

PWM frequency : fréquence du signa PWM

Pulse delay time : permet de décaler dans le temps le démarrage de la génération du signal PWM

Pulse width offset : permet d'imposer un offset sur la fréquence du signal

Input value Vmin for 0% duty cycle : valeur de la tension sur le port +ref qui donne un rapport cyclique de 0%.

Input value Vmax for 100% duty cycle : valeur de la tension sur le port +ref qui donne un rapport cyclique de 100%.

Output voltage amplitude : amplitude de la tension PWM.

Simulation mode : il est possible de choisir un signal en sortie de type **PWM** ou **Averaged** qui moyenne le signal PWM et permet de limiter le temps de calcul du solveur.

2. Commande PWM d'un moteur à courant continu

Il est possible de commander un moteur à courant continu à l'aide du bloc « Controlled PWM Voltage Source ».

Ouvrir le fichier **commande_moteur_PWM.slx**.

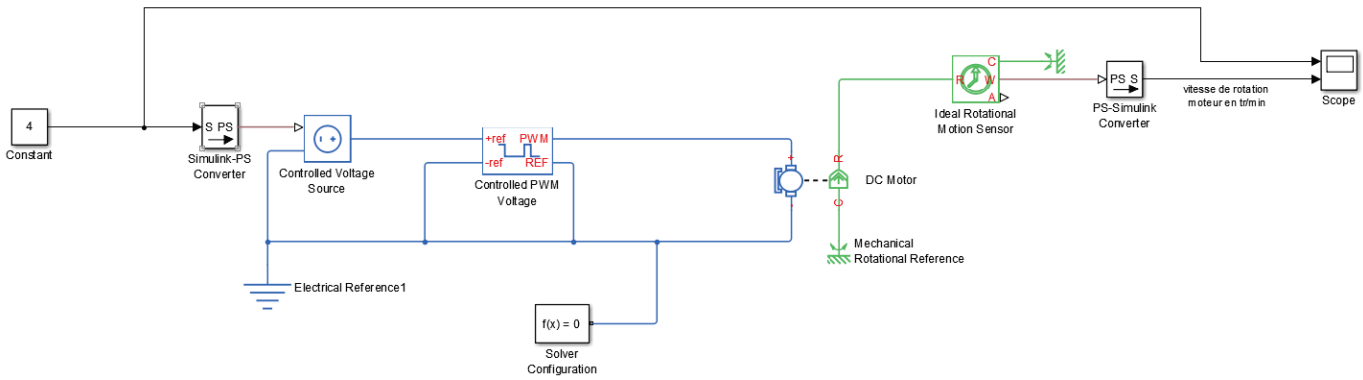


Figure 157 : commande PWM d'un moteur à courant continu

Ce modèle représente un moteur à courant continu commandé par un signal de tension PWM.

Lancer la simulation et observer la vitesse de rotation du moteur à l'aide du scope (Figure 158).

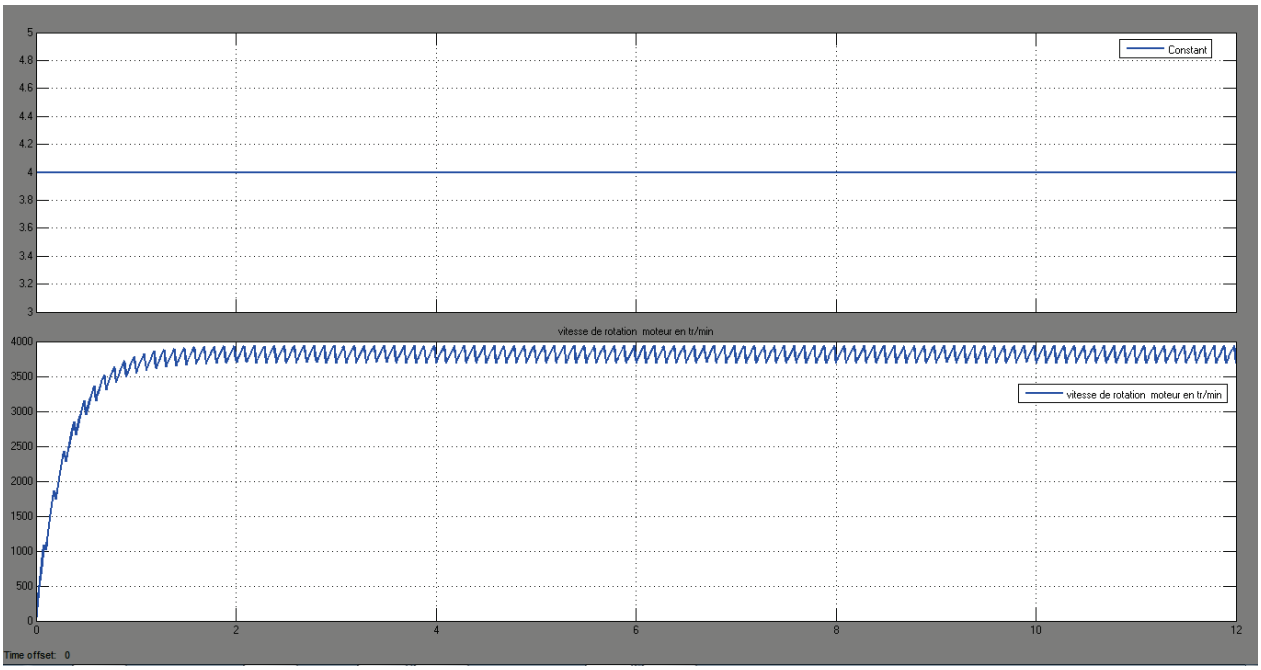


Figure 158 : vitesse de rotation du moteur

La fréquence de la tension PWM est très faible (10 hz). La période du signal est trop proche de la constante de temps du moteur ce qui explique l'allure « hachée » de la vitesse. Pour obtenir une réponse en vitesse du moteur plus satisfaisante, il faut augmenter la fréquence du signal PWM en modifiant le paramétrage du bloc « Controlled PWM Voltage Source ».

Régler la **PWM Frequency** du bloc sur **1000 hz** et relancer la simulation.

Observer la vitesse de rotation du moteur.

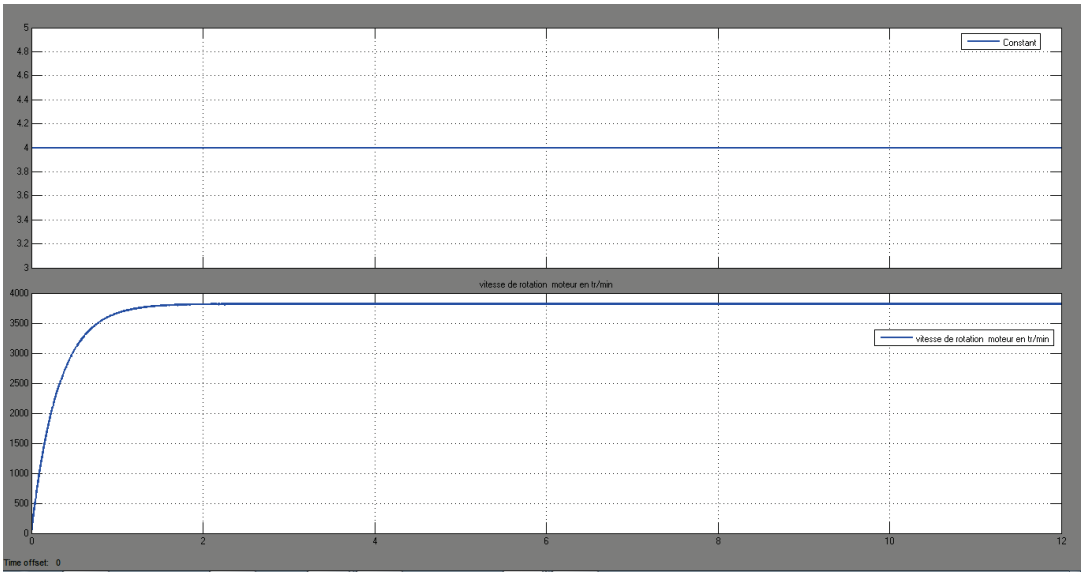
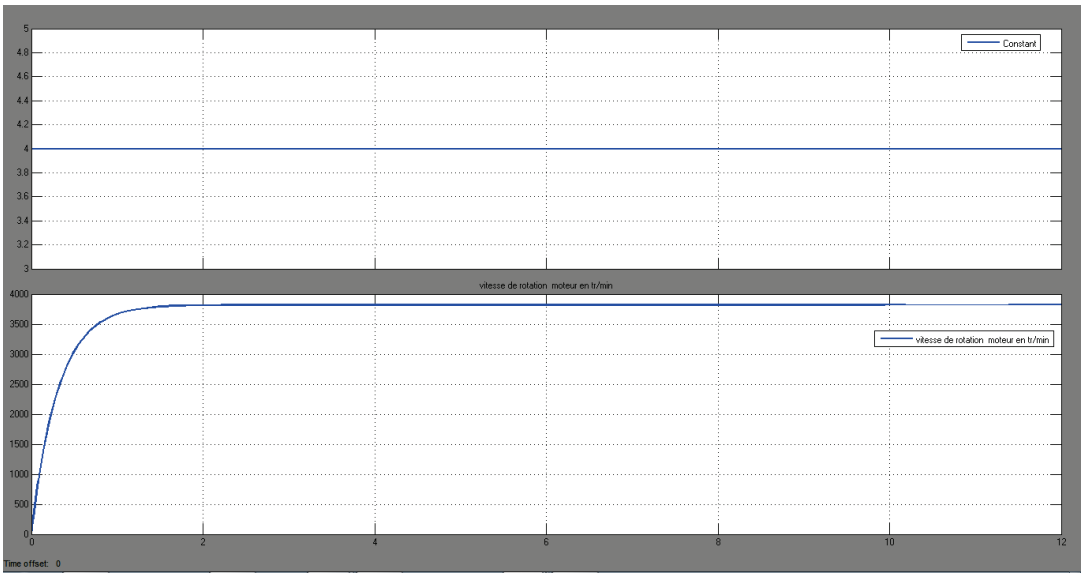


Figure 159 : vitesse de rotation du moteur après modification de la fréquence PWM

On remarque sur la Figure 159 que la réponse en vitesse est maintenant plus lisse et l'influence de la commande PWM est moins visible.

On constate également que le temps de calcul du solveur a considérablement augmenté. Si on règle une fréquence PWM de l'ordre de 20 kHz le temps de simulation devient très long puisque le solveur doit s'imposer au moins 20000 pas de calcul par seconde. Afin de simplifier le travail du solveur on peut régler le **Simulation mode** sur **Averaged** afin de commander le moteur avec la valeur moyenne de la tension. Pour des fréquences élevées, les résultats seront identiques et le temps de calcul largement diminué.

Régler le **Simulation mode** sur **Averaged** et relancer la simulation pour observer le résultat.



Le résultat est le même et l'on a pu évaluer le gain de temps du solveur pour résoudre le modèle.

3. Utilisation du composant « H-Bridge » (pont en H).

Afin de pouvoir contrôler le sens de rotation du moteur, il est nécessaire d'utiliser un pont en H. Le composant « **H-Bridge** » remplit cette fonction.

Ouvrir le fichier **pont_en_H.slx**.

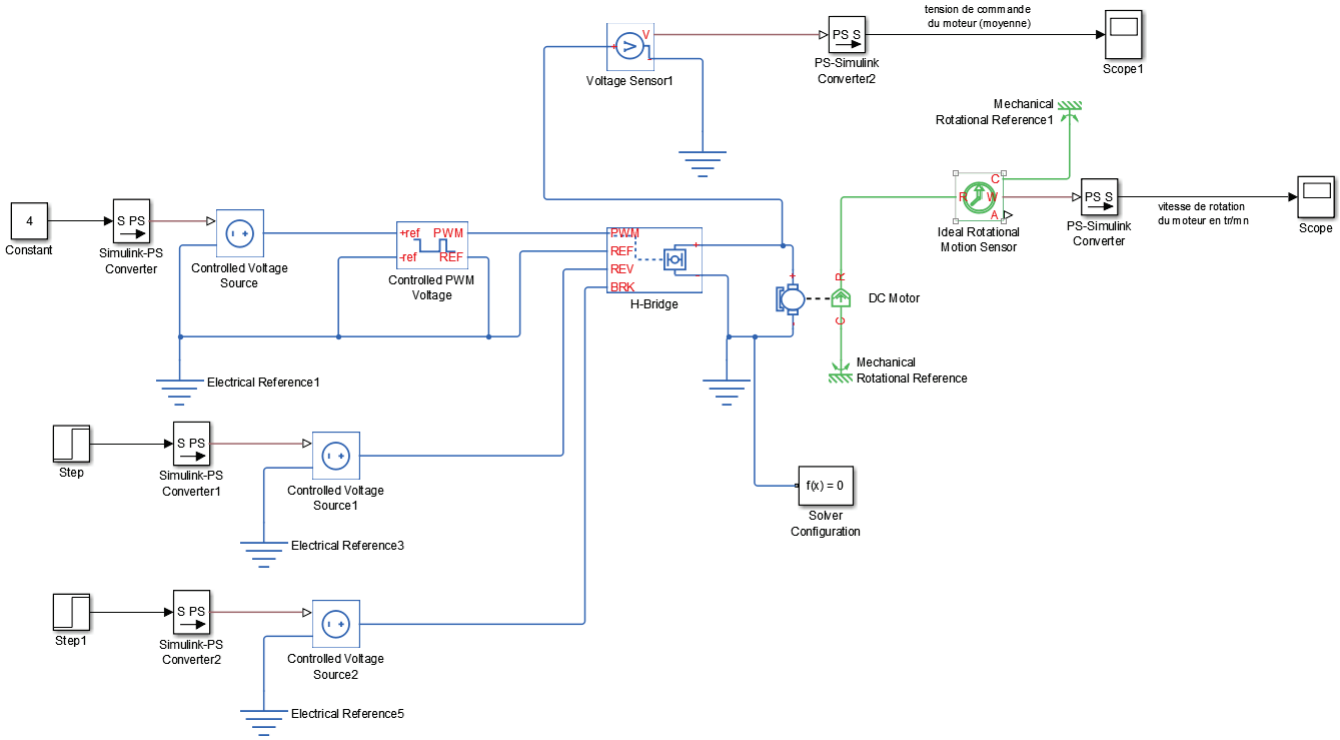


Figure 160 : commande PWM d'un moteur à courant continu avec pont en H

Ce modèle permet de commander un moteur à courant continu à partir d'une commande PWM et d'un pont en H. Il est possible de visualiser la vitesse de rotation du moteur ainsi que la tension moyenne d'alimentation du moteur

Lancer la simulation et **observer** la vitesse de rotation du moteur.

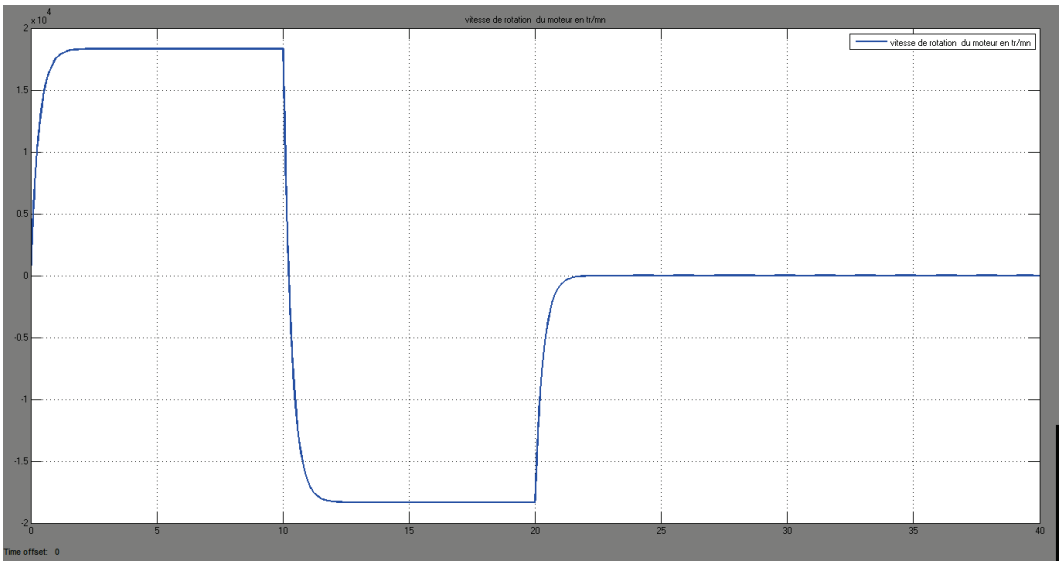


Figure 161 : vitesse de rotation du moteur commandé par le pont en H

Le moteur tourne dans le sens positif jusqu'à t=10 s, puis dans le sens négatif jusqu'à t=20 s, puis s'arrête à partir de t=20 s.

On remarque qu'à l'instant t=10 s un échelon de tension de 5 V est imposé sur l'entrée **REV** du pont en H provoquant l'inversion du sens de rotation. A l'instant t=20 s, un échelon de tension de 5 V est imposé sur l'entrée **BRK** du pont en H provoquant l'arrêt du moteur.

Observer les variations de la tension moyenne d'alimentation du moteur.

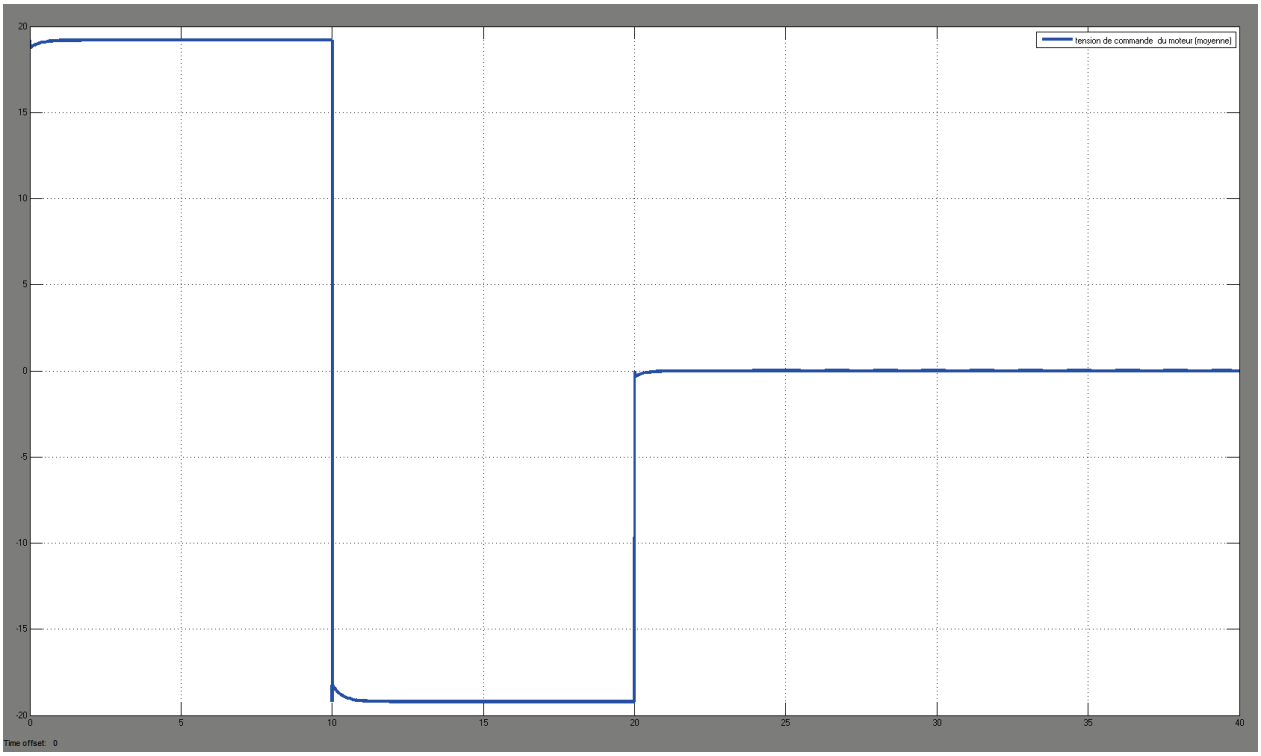
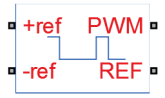


Figure 162 : tension moyenne d'alimentation du moteur

Paramétrage

Controlled PWM Voltage Source	 Controlled PWM Voltage	Simscape/SimElectronics/Actuators and Drivers/Drivers

La fréquence PWM est réglée sur 1000 hz et le **Simulation mode** sur **Averaged.**

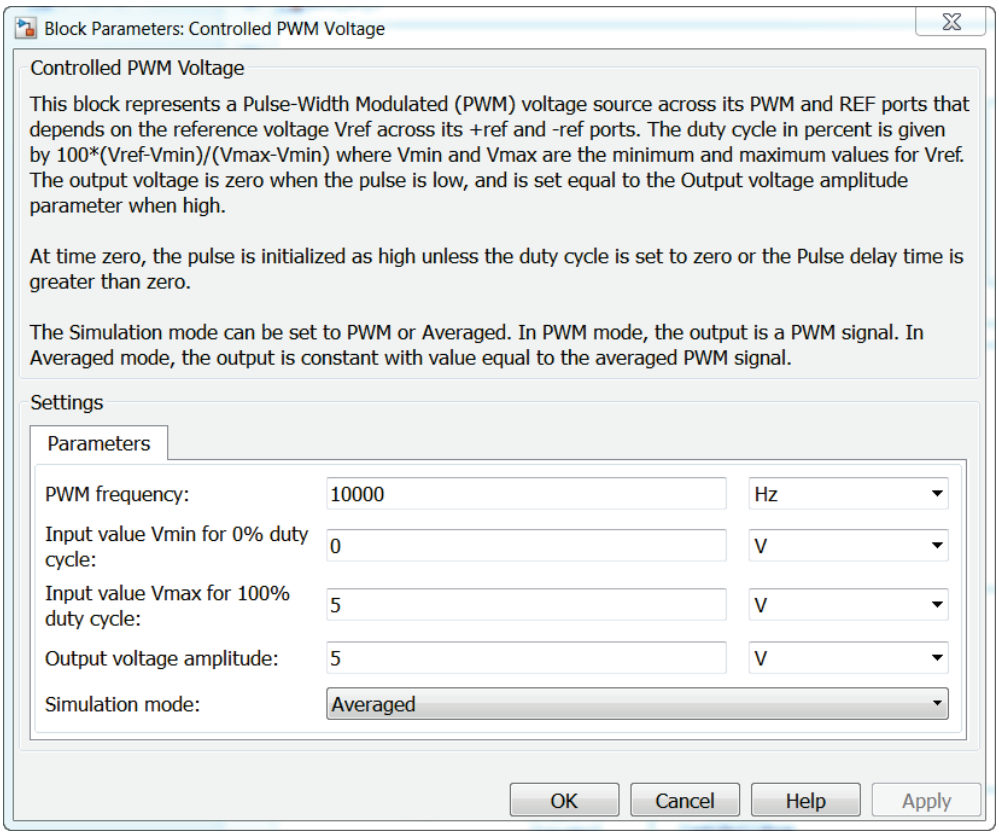
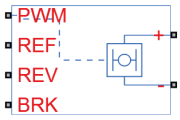


Figure 163 : paramétrage du bloc Controlled PWM Voltage

Paramétrage

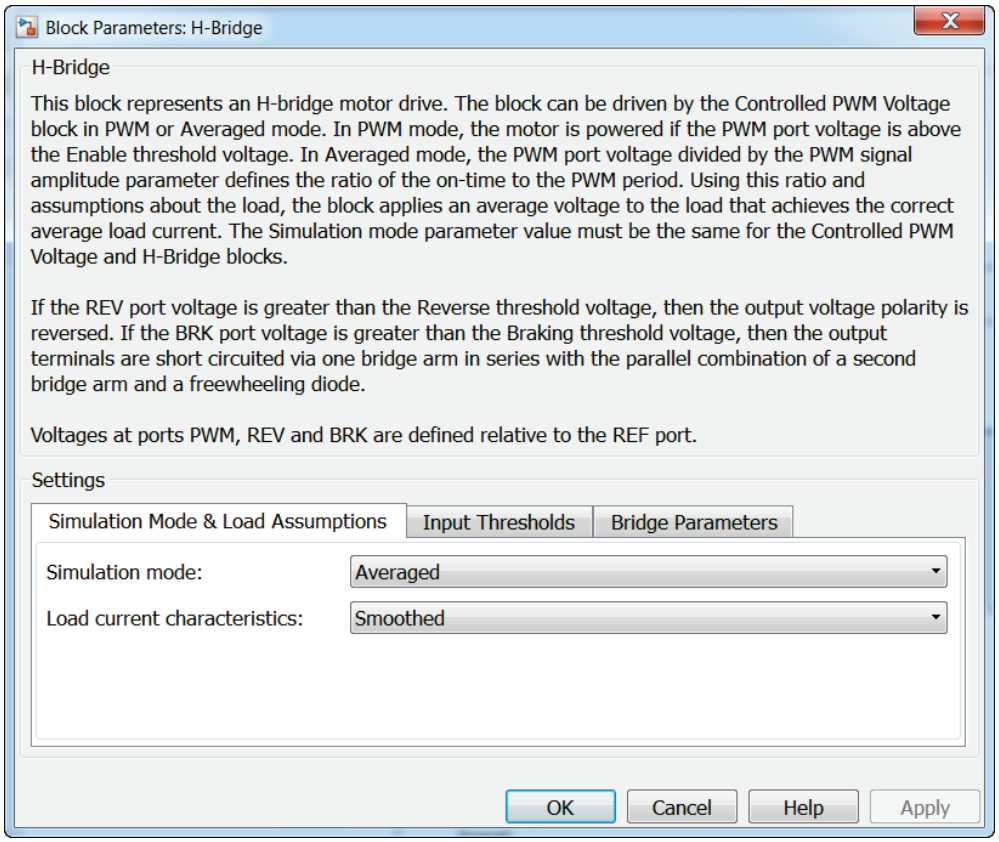
H-Bridge	 H-Bridge	Simscape/SimElectronics/Actuators and Drivers/Drivers

Ce composant modélise un pont en H. Il possède 6 ports de type **PCP** du domaine électrique.

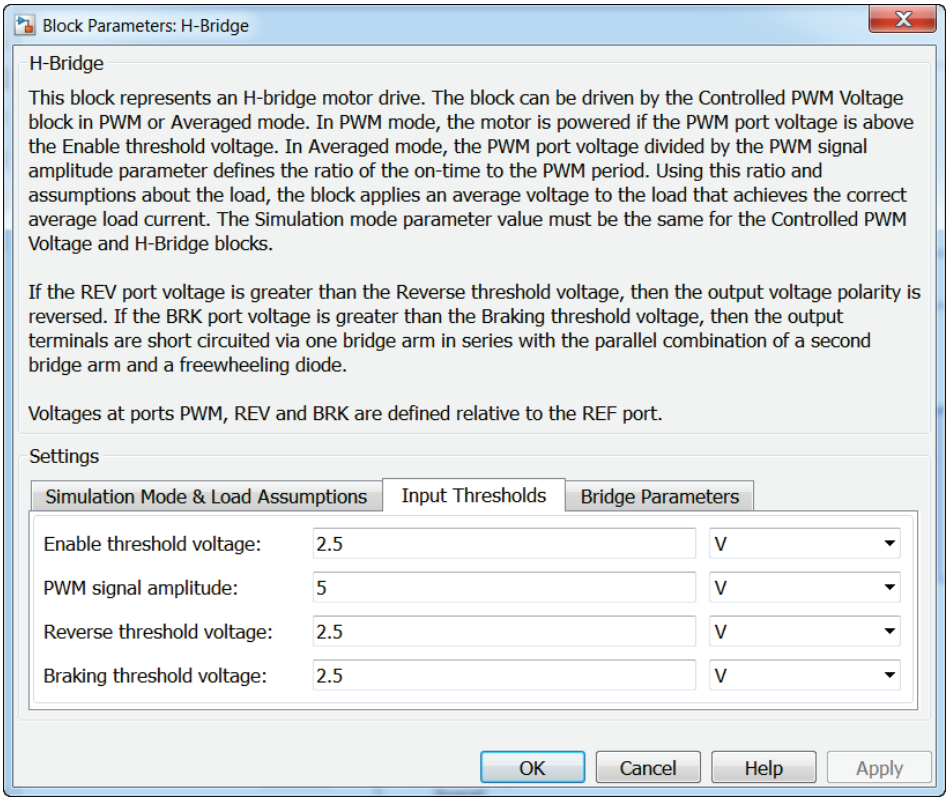
- **PWM** : signal d'entrée PWM en provenance du bloc « Controlled PWM Voltage Source »
- **REF** : référence de la tension du signal PWM (souvent relié à la masse)
- **REV** : commande de l'inversion du sens de rotation
- **BRK** : commande de l'arrêt du moteur
- **+** et **-** : ces deux ports sont reliés aux bornes du moteur

Le pont en H reçoit, la tension PWM sur le port d'entrée **PWM** et alimente le moteur directement avec cette tension. Si la tension de l'entrée **REV** dépasse un seuil fixé dans les paramètres du composant, le sens de rotation s'inverse. Si la tension de l'entrée **BRK** dépasse un seuil fixé dans les paramètres du composant, l'alimentation du moteur est coupée.

Onglet **Simulation Mode & Load assumptions**



Le **Simulation Mode** est réglé sur **Averaged** afin de minimiser le temps de calcul du solveur.
Onglet **Input Thresholds**



C'est ici que sont entrés les différents seuils qui caractérisent le fonctionnement du pont en H.

Enable thresholds voltage : niveau de tension que doit atteindre l'amplitude du signal PWM pour déclencher l'alimentation du moteur (uniquement valable si le **Simulation mode** est sur **PWM**)

PWM signal amplitude : amplitude du signal PWM qui doit correspondre avec l'amplitude du signal PWM généré par le bloc « Controlled PWM voltage source »

Reverse thresholds voltage : niveau de tension relevée sur le port **REV** qui provoque l'inversion du sens de rotation du moteur

Braking thresholds voltage : niveau de tension relevée sur le port **BRK** qui provoque l'arrêt du moteur.

Onglet **Bridge Parameters**

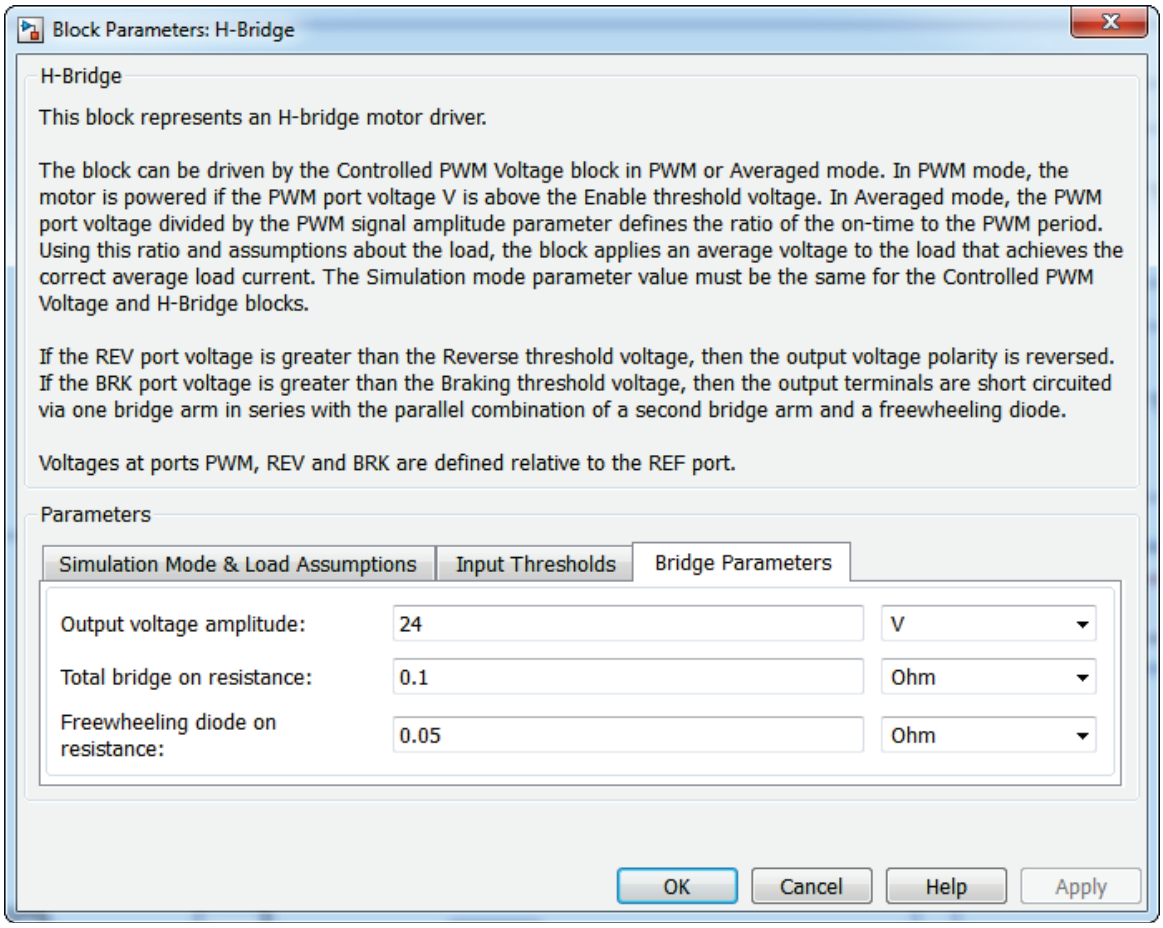


Figure 164 : paramétrage du bloc H-Bridge

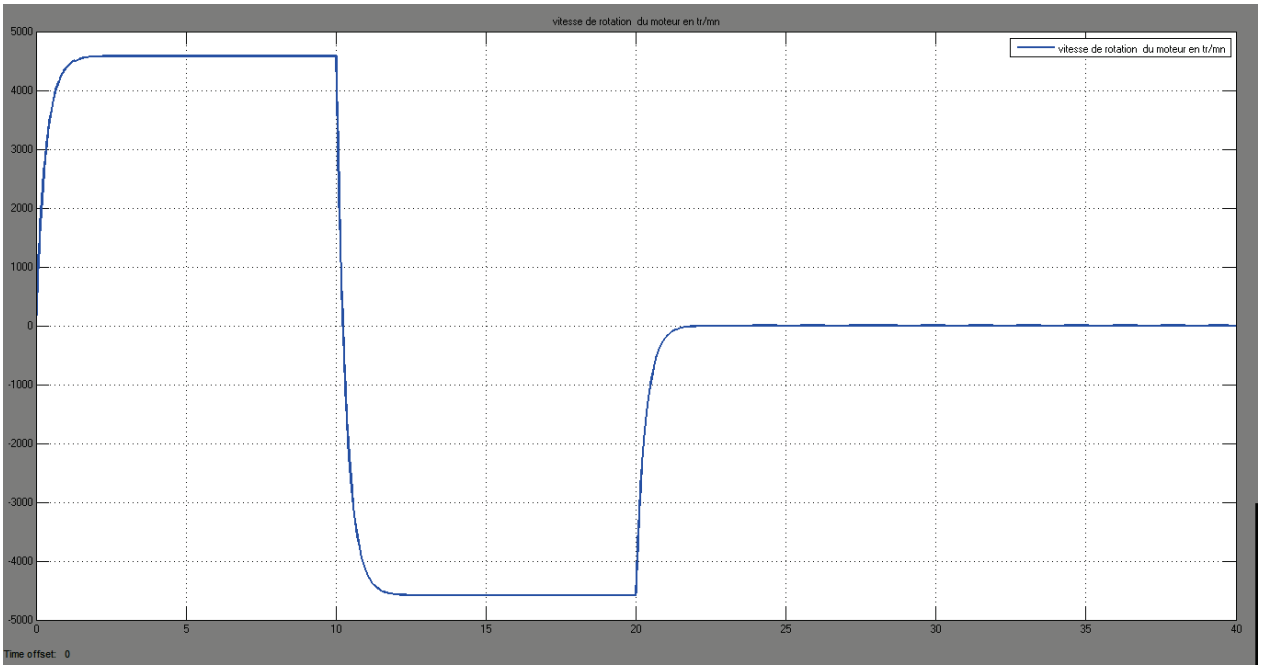
Output voltage amplitude : permet de fixer l'amplitude maximale de la tension d'alimentation du moteur.

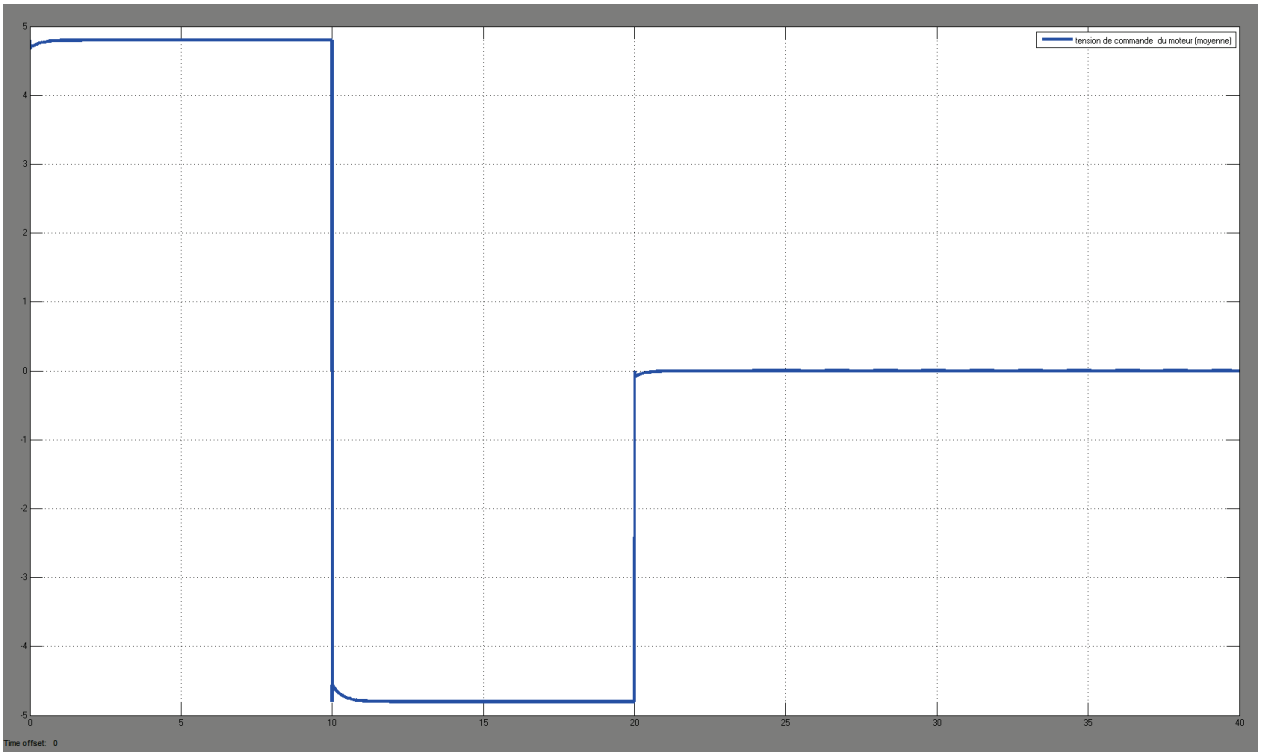
Total bridge on resistance : résistance des deux transistors qui jouent le rôle d'interrupteur pour le pont en H.

Freewheeling diode resistance : résistance des diodes de roue libre du pont qui permettent le passage transitoire du courant lors de la fermeture des interrupteurs (uniquement pour le **Simulation Mode** PWM)

Modifier le bloc Constant permettant d'alimenter le composant « Controlled PWM Voltage Source » et fixer sa valeur à **1** au lieu de **4**.

Lancer la simulation et **visualiser** l'influence sur la tension d'alimentation et la vitesse de rotation du moteur.





D. Rendre un modèle interactif, utilisation de la bibliothèque « dashboard »

Il est possible d'interagir avec un modèle durant la simulation. Pour cela, la bibliothèque de Simulink intitulée « **dashboard** » propose une série de boutons et d'interfaces de visualisation qui vont nous permettre de créer des modèles interactifs (Figure 165).

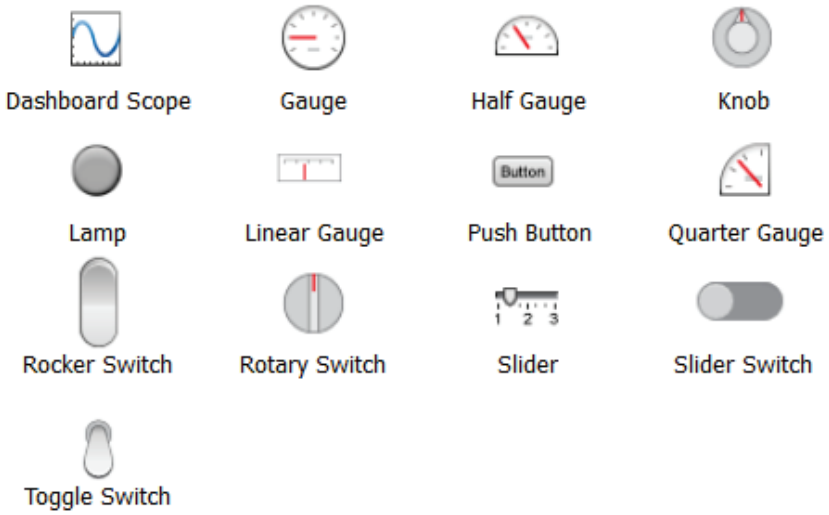


Figure 165 : les éléments de la bibliothèque de Simulink « Dashboard »

Il est ainsi possible d'agir sur des boutons pour modifier des paramètres du modèle et d'observer en temps réel l'influence de cette modification sur le comportement du modèle.
Afin de rendre la visualisation plus réaliste, il est préférable que le calcul s'effectue en temps réel, c'est-à-dire que 1 s dans le déroulement de la simulation représentera 1 s dans la réalité.
Le bloc **« Real-Time Pacer »** de la bibliothèque **Real-Time Pacer** permet d'effectuer ce réglage.

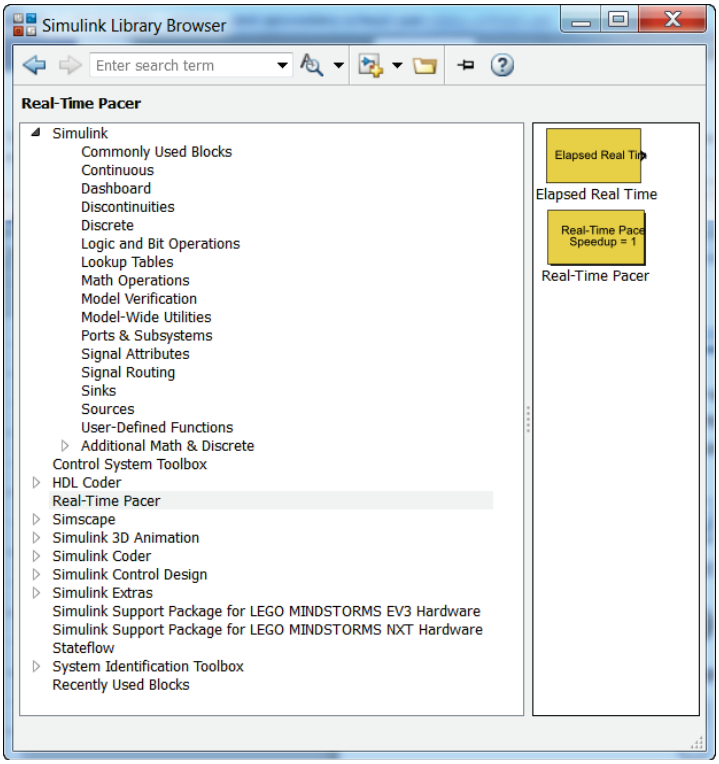


Figure 166 : la bibliothèque Real-Time Pacer de Simulink

Si la bibliothèque Real-Time Pacer n'apparaît pas dans l'arborescence des bibliothèques de Simulink, le dossier **« Real-Time Pacer »** doit se trouver dans le Path de MATLAB. **Cliquer** avec le bouton droit de la souris dans la fenêtre de la bibliothèque Simulink et sélectionner **Refresh Library Browser**.
La bibliothèque de Simulink est mise à jour avec tous les fichiers portant l'extension **.lib** présents dans le Path de MATLAB.

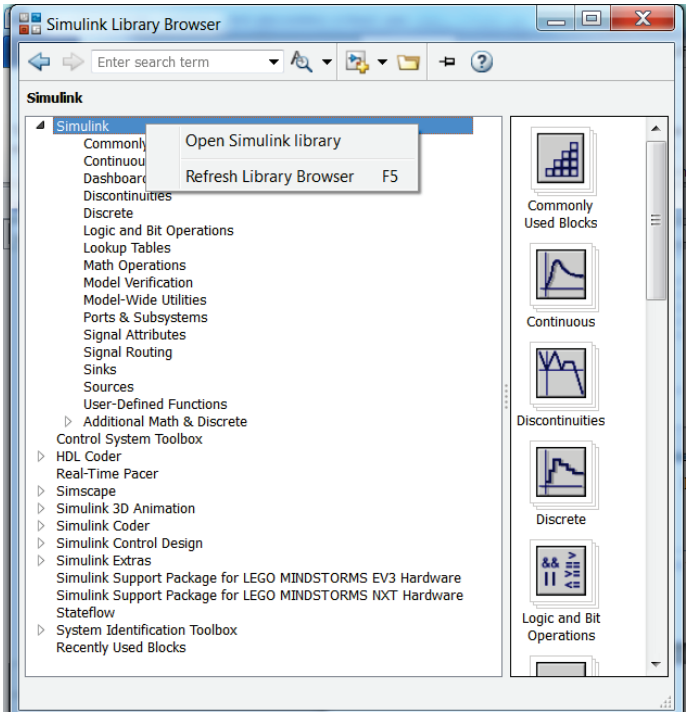


Figure 167 : procédure de rafraichissement de la bibliothèque Simulink

Cette opération doit impérativement être effectuée avant d'ouvrir les modèles utilisés dans cette partie afin que le bloc **Real Time Pacer** puisse apparaître.

1. Exemple de modèle interactif

Ouvrir le fichier **« axe_lineaire_dashboard.slx »**.

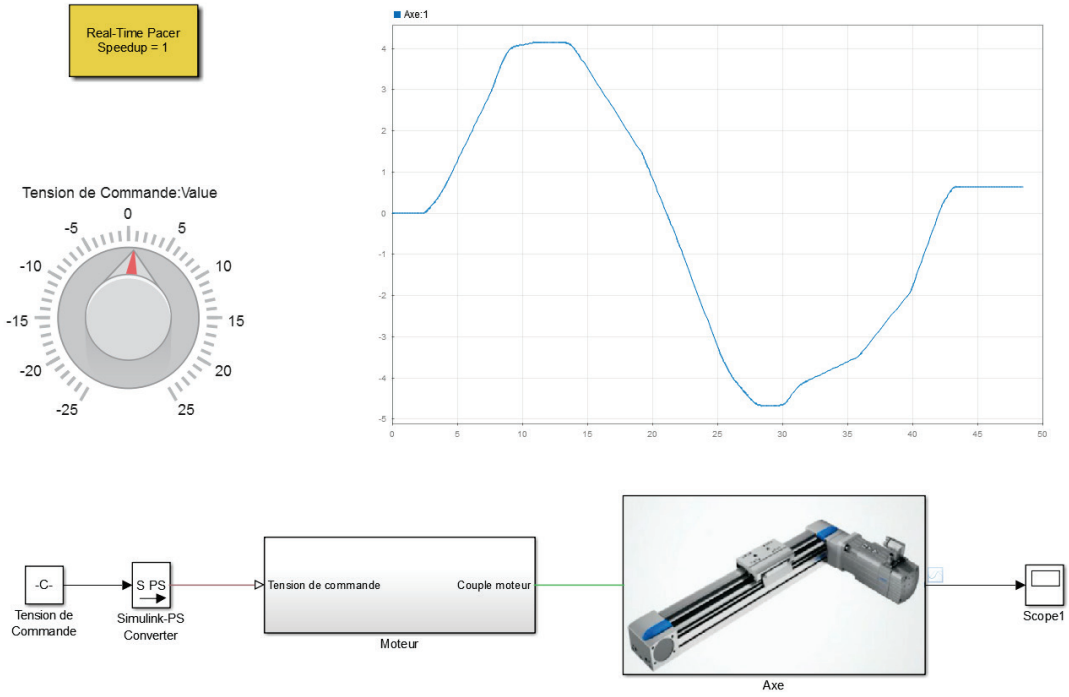


Figure 168 : pilotage interactif d'un modèle

Le **temps de simulation** est réglé sur **inf**, ce qui signifie que le temps de simulation est **infini**.

Pour stopper la simulation du modèle il faudra cliquer sur le bouton **Stop**  .

Lancer la simulation et agir sur le bouton **« Tension de commande »** par simple glisser déposer et observer la variation de la position linéaire de l'axe directement dans le scope interactif.

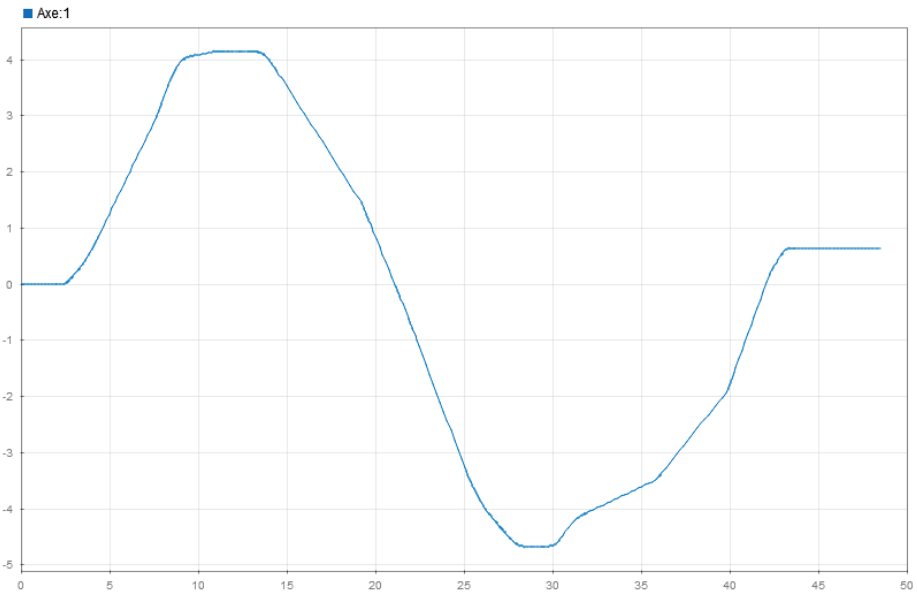


Figure 169 : visualisation en temps réel de la position de l'axe linéaire

2. Utilisation des blocs de la bibliothèque

Dans cet exemple nous allons paramétrer les blocs de la bibliothèque « Dashboard » du modèle utilisé précédemment.

Ouvrir le modèle **« axe_lineaire_dashboard_start.slx »:**

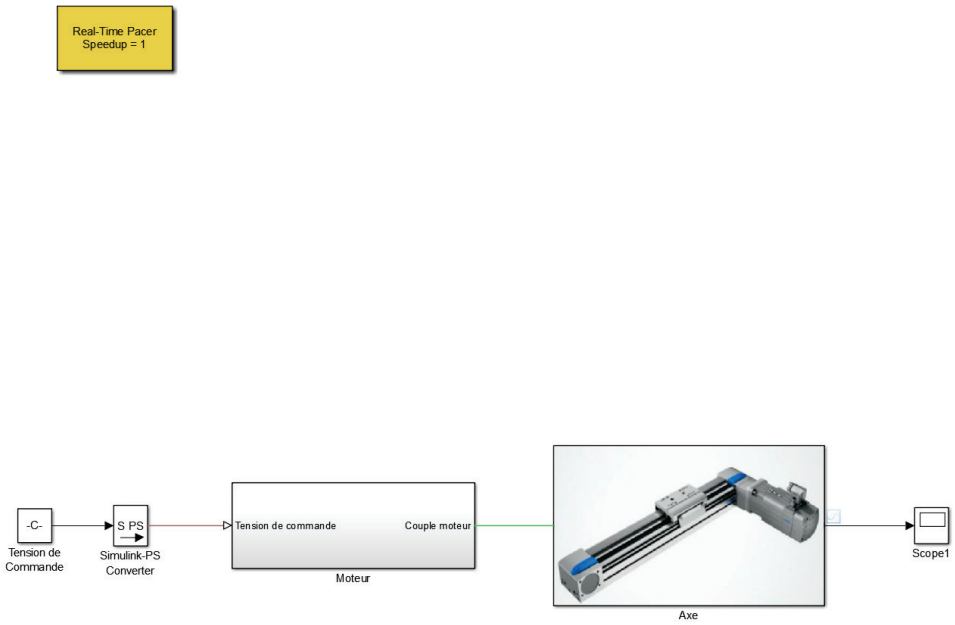


Figure 170 : modèle « axe_lineaire_dashboard_start.slx

Fonction du composant	Représentation	Bibliothèque
Bouton interactif (Knob)	 Knob	Simulink/Dashboard
Scope interactif	 Dashboard Scope	Simulink/Dashboard

Faire **glisser**, **positionner** et **redimensionner** le bloc **knob** et le bloc **Dashboard Scope** pour obtenir la configuration de la Figure 171.

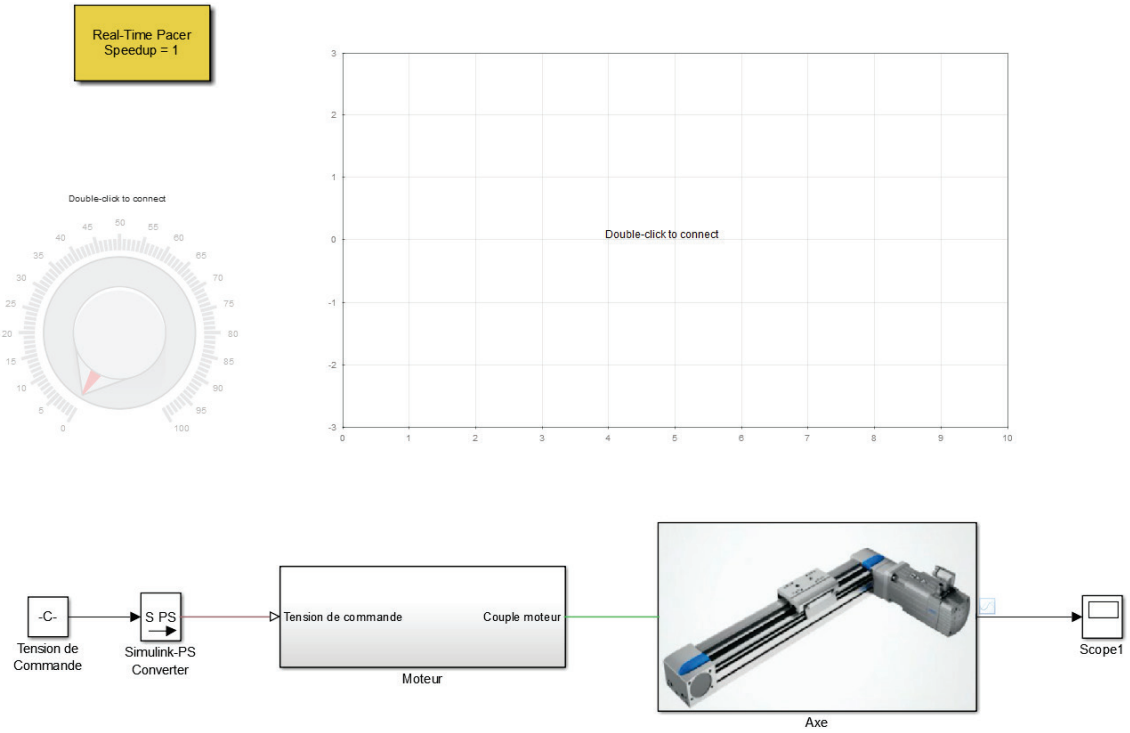


Figure 171 : insertion des blocs de la bibliothèque Dashboard

A ce stade les blocs de la bibliothèque Dashboard ne sont pas encore connectés au modèle et ne permettent pas d'interagir avec lui.

Paramétrage

Bouton interactif (Knob)  Simulink/Dashboard

Double cliquer sur le bloc **Knob**.
Cliquer ensuite sur le bloc qui sera piloté par le bouton. Ici, il faudra sélectionner le bloc **« Tension de commande »** qui impose la tension à l'entrée de l'axe linéaire puis valider la connexion en cliquant sur la case à cocher **connect (**Figure 172**)**.
Le paramétrage du composant consiste également à préciser la plage de variation du paramètre.

Minimum : limite inférieure de variation du paramètre

Maximum : limite supérieure de variation du paramètre

Ticks : intervalle entre deux graduations

La fenêtre de paramétrage doit être identique à la Figure 172.

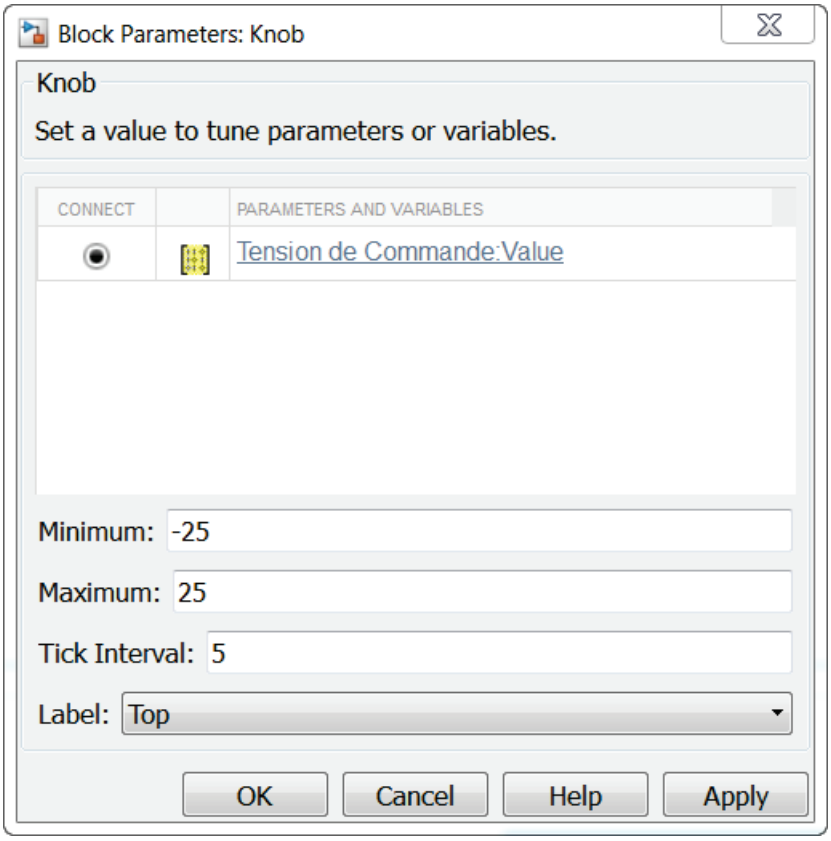


Figure 172 : fenêtre de paramétrage du bloc knob

Le bouton change alors d'aspect pour prendre la forme représenté sur la Figure 173.

Figure 173 : allure du bouton knob après paramétrage et connexion avec le modèle

Paramétrage

Scope interactif	 Dashboard Scope	Simulink/Dashboard
Bouton interactif (Knob)	 Knob	**Simulink/Dashboard**

Double cliquer sur le bloc **Scope Interactif.**
Cliquer ensuite sur le lien qui contient le signal que l'on souhaite afficher. Ici, il faudra sélectionner le signal de sortie du modèle (celui qui entre dans le scope et qui donne la position linéaire de l'axe) puis valider la connexion en cliquant sur la case à cocher **connect (**Figure 174**)**.
Le paramétrage du composant consiste également à préciser la plage de temps d'affichage de la courbe et les limites supérieures et inférieures de l'axe Y.

Time Span : plage de temps spécifiée pour l'observation du phénomène

Y-Axis Limits : limites inférieure et supérieure de l'axe Y

La fenêtre de paramétrage doit être identique à la Figure 174.

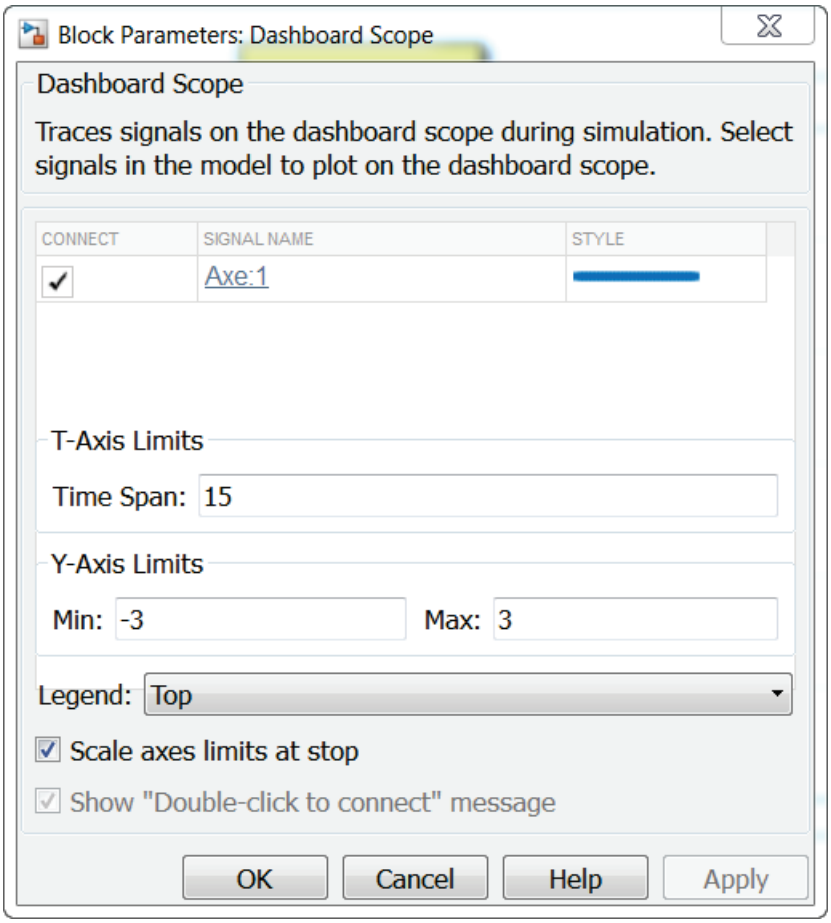


Figure 174 : fenêtre de paramétrage du bloc Dashboard Scope

Lancer la simulation et observer en temps réel l'évolution de la position de l'axe.

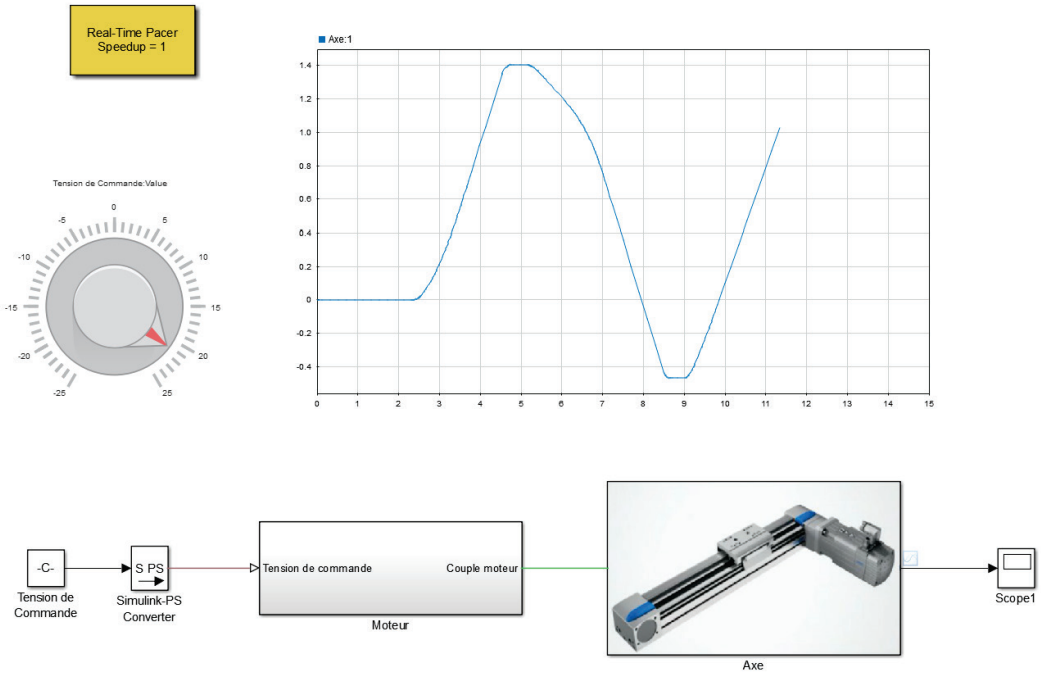


Figure 175 : simulation d'un modèle interactif avec les composants de la bibliothèque dashboard

V. Application pédagogique

Les paragraphes précédents ont montré comment utiliser les fonctionnalités de Simscape et de ses bibliothèques pour construire des modèles exploitables. Afin d'utiliser ces modèles à des fins pédagogiques, il est nécessaire d'approfondir l'analyse et d'expliquer la stratégie à mettre en œuvre pour utiliser ces modèles comme supports de fomation.

La simple présentation d'un modèle à des étudiants n'est pas suffisante pour garantir une efficacité dans le processus d'apprentissage. Un travail de réflexion pédagogique de la part de l'enseignant est impératif pour construire une séquence pertinente autour de l'exploitation d'un modèle numérique. Dans l'exemple qui va suivre, nous allons voir comment il est possible de passer de la simple conception d'un modèle à son exploitation pédagogique avec des étudiants. Cette démarche sera illustrée en exploitant le modèle du hacheur série.

A. Présentation du hacheur série

Les hacheurs sont des convertisseurs directs du type continu-continu. Ils permettent d'obtenir une tension continue réglable à partir d'une tension continue fixe. Ils sont particulièrement utilisés pour la commande des machines à courant continu (Figure 176).

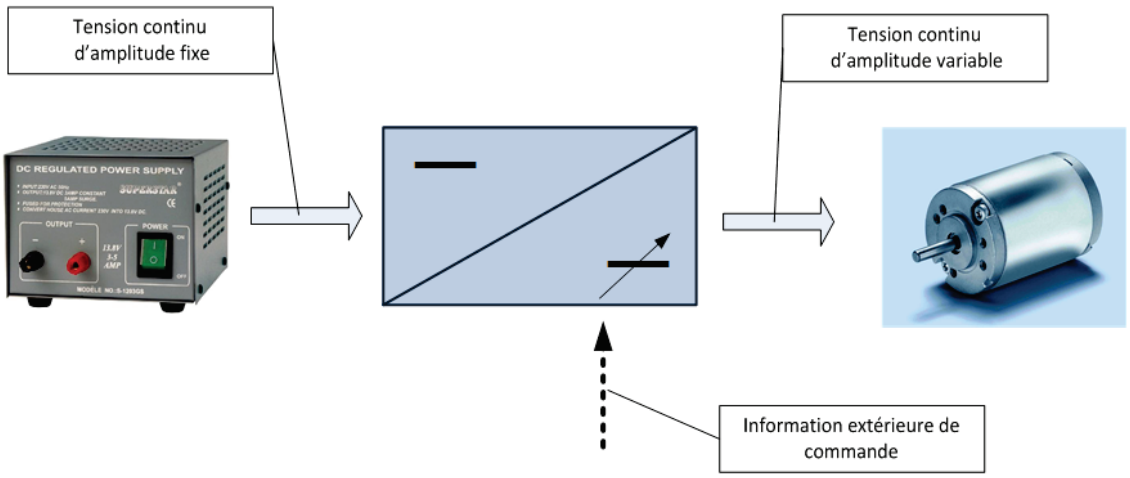


Figure 176 : présentation du hacheur série

Le hacheur va convertir une tension continue en une tension qui prend la forme d'un signal carré (**Figure 177**)

- T: période de hachage (ou de découpage)
- $\boldsymbol{\alpha = \frac{t_{on}}{T}}$: rapport cyclique, si $\boldsymbol{\alpha = 1}$, la tension de sortie est continu

Le signal obtenu en sortie du hacheur est appelé signal PWM (Pulse Width Modulation – Modulation de la largeur d'impulsion).

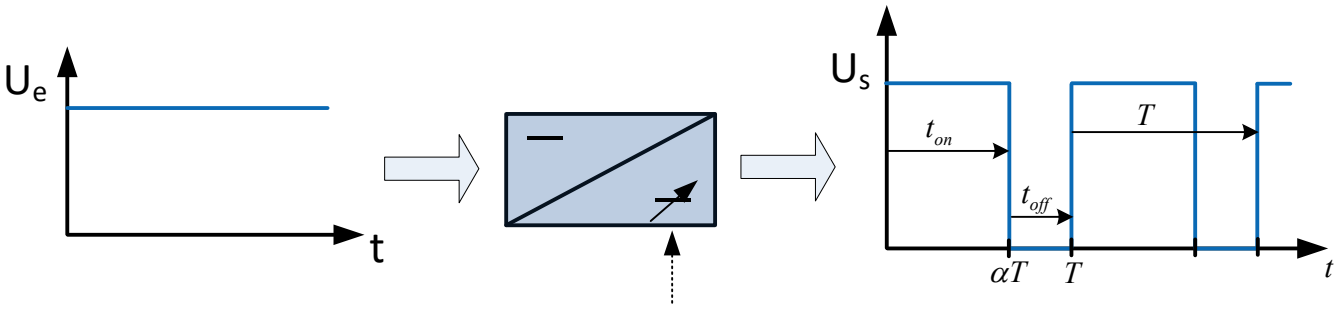


Figure 177 : principe de fonctionnement du hacheur série

Une source d'entrée délivre une tension continue $\mathbf{U_e}$. Périodiquement, on ferme un interrupteur K pour obtenir la tension $\mathbf{U_s}$ aux bornes de la charge. En pratique, l'interrupteur K est un transistor qui travaille en commutation.

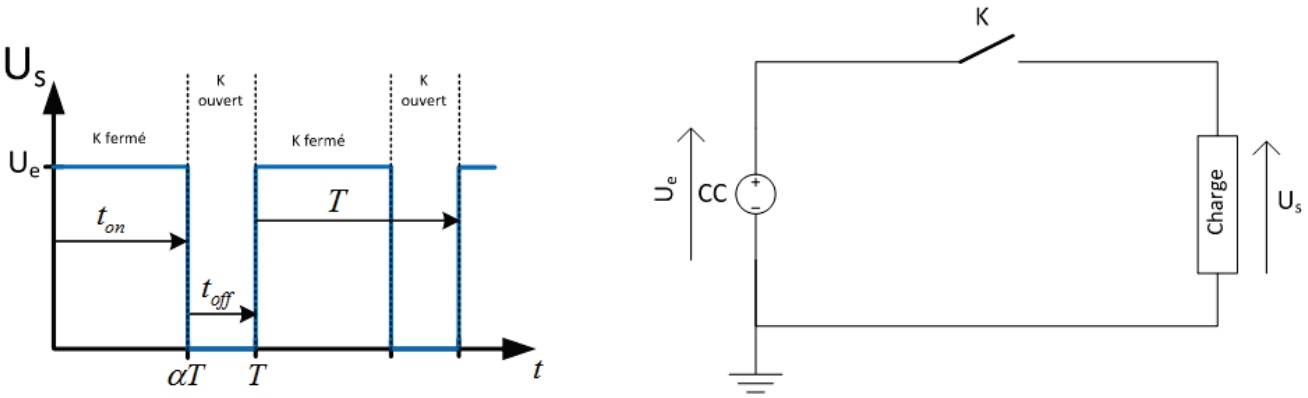


Figure 178 : schéma de principe du hacheur série

Durant une période T :

- la charge est alimentée durant la durée $\mathbf{t_{on}}$, c'est la phase active
- l'alimentation est coupée durant la durée $\mathbf{t_{off}}$, c'est la phase de roue libre

En pratique, la fréquence de hachage est de l'ordre de quelques kHz et la charge se comporte comme si elle était alimentée par une tension moyenne $\langle \mathbf{U_c} \rangle = \boldsymbol{\alpha} \mathbf{U_e}$.

Le hacheur série permettant de commander un moteur à courant continu doit comporter 2 interrupteurs (**Figure 179**) :

- l'interrupteur K1 est un transistor de type MOS ou IGBT, unidirectionnel en tension et en courant (commutation commandée)
- l'interrupteur K2 est une diode, unidirectionnelle en tension et en courant (commutation spontanée)

En phase active, le transistor $\mathbf{K_1}$ est équivalent à un interrupteur fermé. La diode $\mathbf{K_2}$ est bloquée (VK>VA) et se comporte comme un interrupteur fermé.

En phase de roue libre, le transistor $\mathbf{K_1}$ est équivalent à un interrupteur ouvert. La diode $\mathbf{K_2}$ est passante (VA=VK) . Une diode ainsi utilisée est appelée « diode de roue libre ».

K_1 est un transistor de type MOS ou IGBT (commutation commandée)

K_2 est une diode (commutation spontanée)

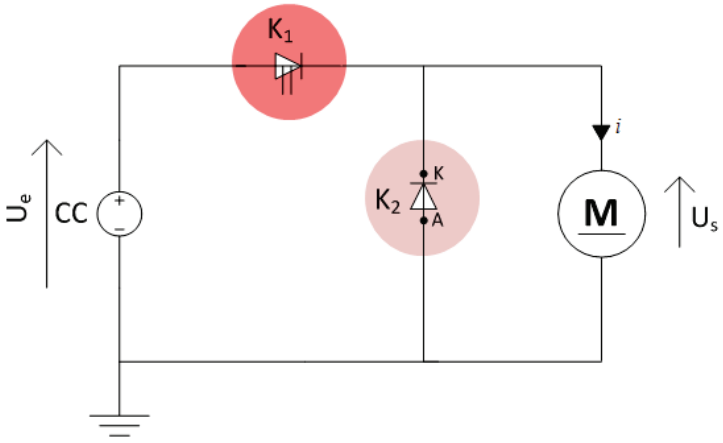


Figure 179 : schéma de commande d'un moteur à courant par un hacheur série

B. Objectifs pédagogiques

A l'issue de la séquence l'étudiant doit être en mesure de comprendre les principes de fonctionnement du hacheur série :

Objectif 1 : Comprendre la circulation du courant dans le circuit en phase active et en phase de roue libre (Figure 180).

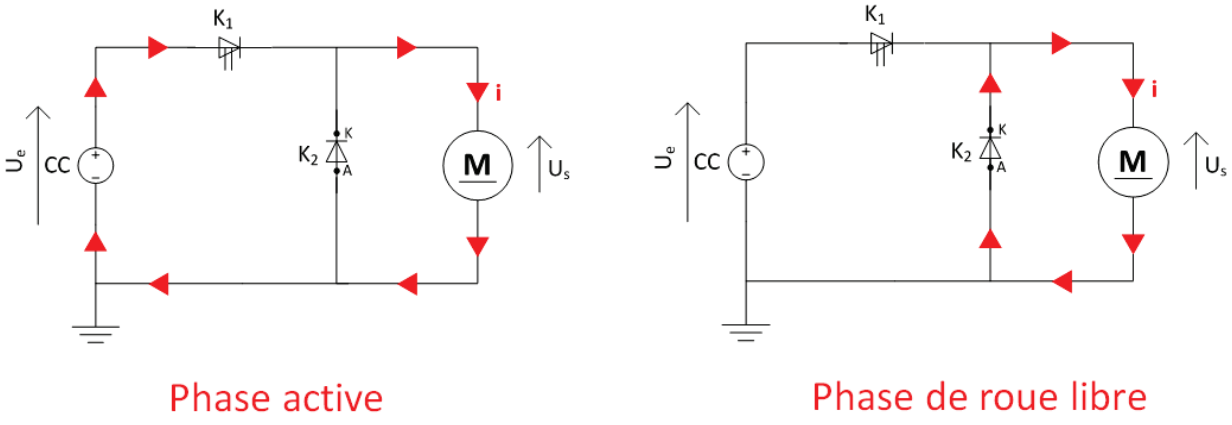


Figure 180 : circulation du courant dans le circuit en phase active et en phase de roue libre

Objectif 2 : Visualiser et évaluer l'influence du rapport cyclique sur l'ondulation du courant (Figure 181)

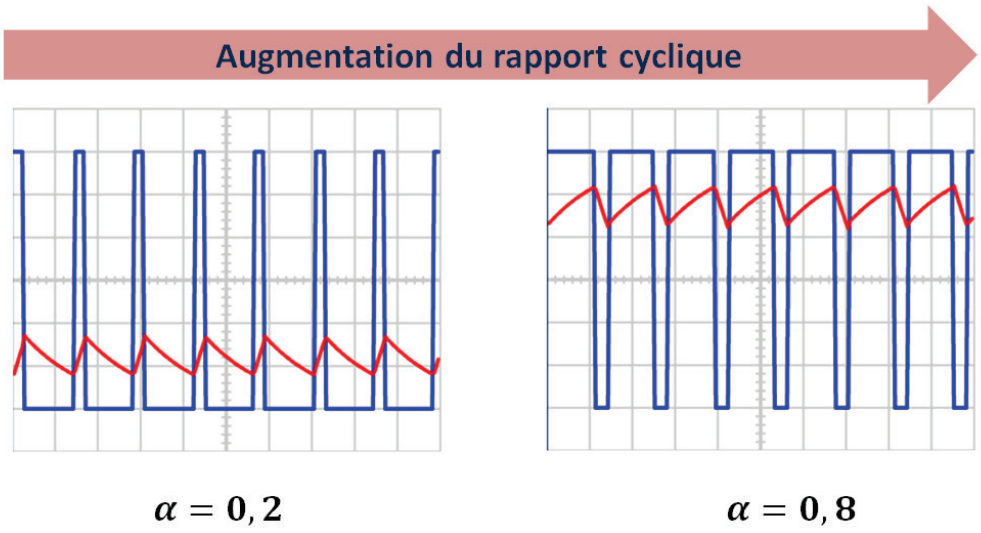


Figure 181 : visualisation de l'influence du rapport cyclique sur l'ondulation de courant

Objectif 3 : Visualiser et évaluer l'influence de la fréquence de hachage sur l'ondulation du courant (Figure 182)

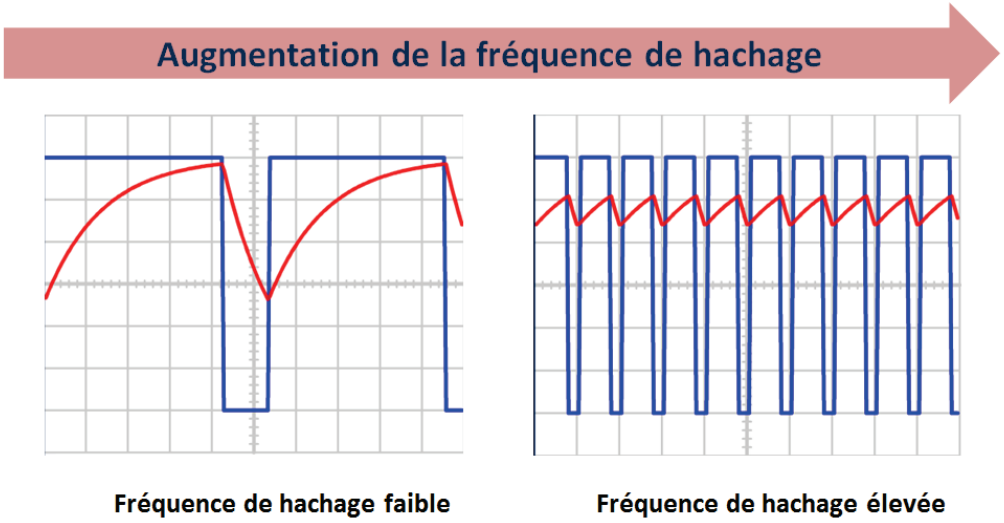


Figure 182 : visualisation de l'influence de la fréquence de hachage sur l'ondulation de courant

Objectif 4 : Visualiser et évaluer l'influence de la valeur de l'inductance sur l'ondulation du courant (Figure 183)

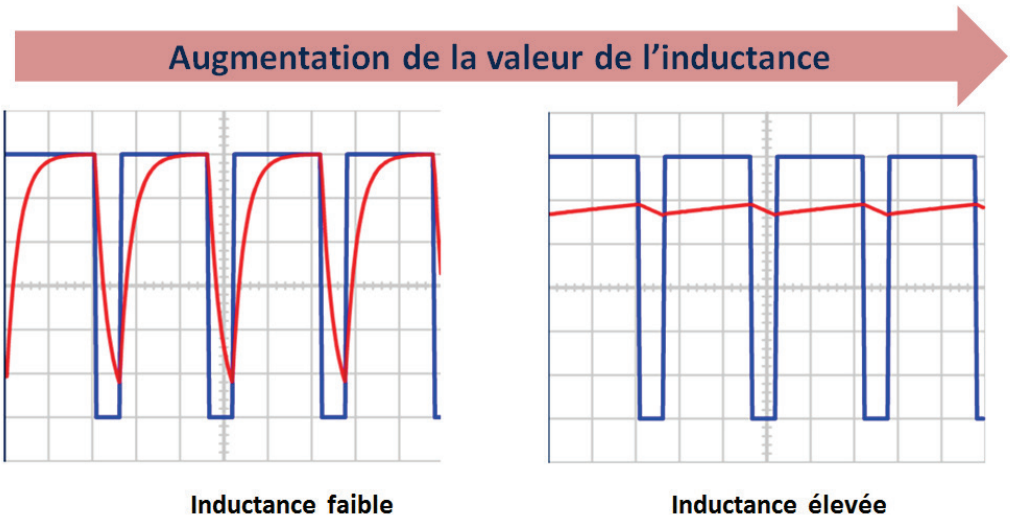


Figure 183 : visualisation de l'influence de la valeur de l'inductance sur l'ondulation de courant

C. La construction du modèle

Le modèle numérique doit être construit, de manière à créer une analogie visuelle avec la schématisation afin de faciliter sa prise en main par les étudiants (Figure 184). L'étudiant identifie facilement tous les composants et peut faire le lien avec ses connaissances théoriques.

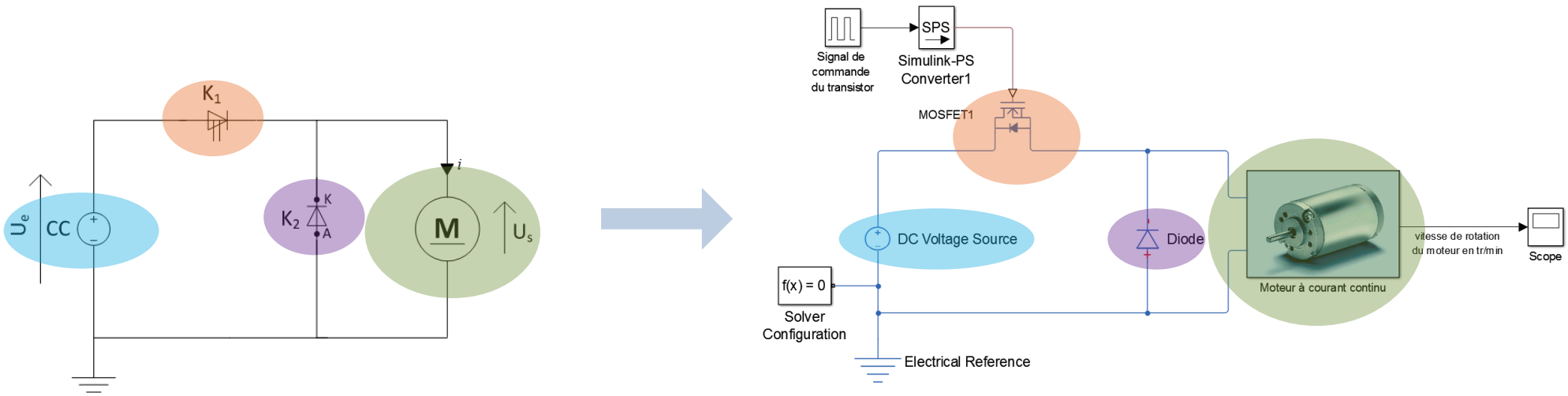


Schéma électrique de principe du hacheur série

Modèle Simscape du hacheur série

Figure 184 : analogie entre la forme du modèle Simscape et le schéma électrique de principe

Ouvrir le modèle **« hacheur-serie_0.slx ».**

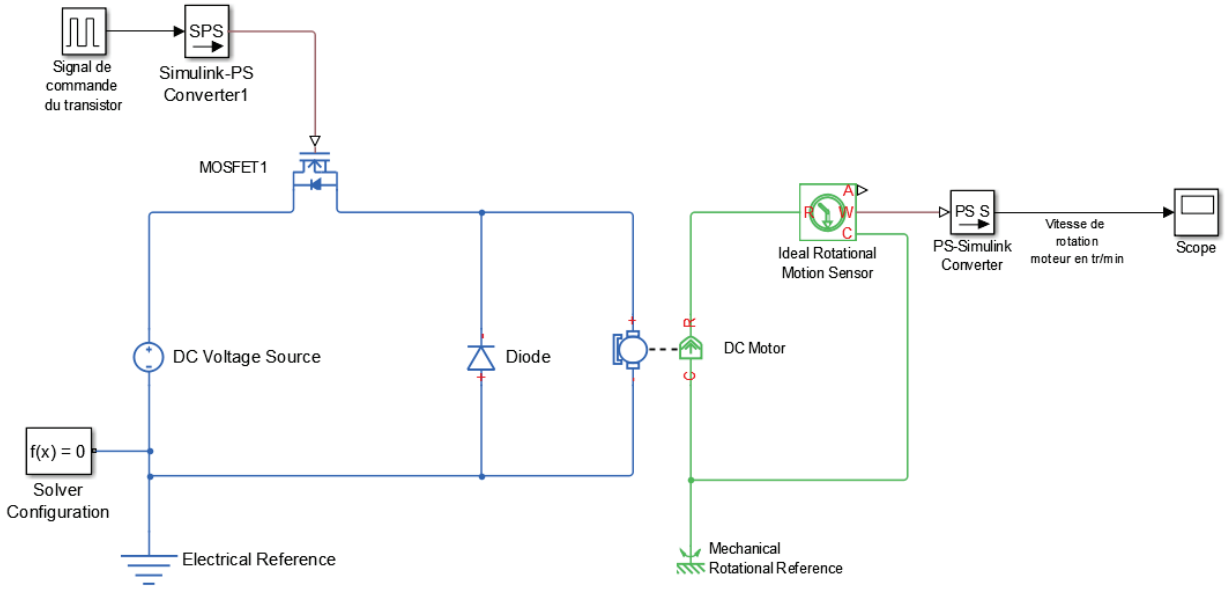


Figure 185 : modèle Simscape du hacheur série

Seuls sont référencés dans le tableau de la Figure 186 les composants qui n'ont pas encore été utilisés dans l'ouvrage.

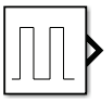
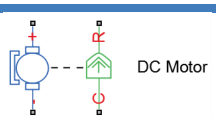
Fonction du composant	Représentation	Bibliothèque
Transistor MOFSET	 MOSFET	Simscape/SimPowerSystems/Simscape Components/SemiConductors/Fundamental Components
Diode	 Diode	Simscape/SimPowerSystems/Simscape Components/SemiConductors/Fundamental Components
Générateur de signaux carrés	 Pulse Generator	Simulink/Sources

Figure 186 : les composants nécessaires à la modélisation du hacheur série

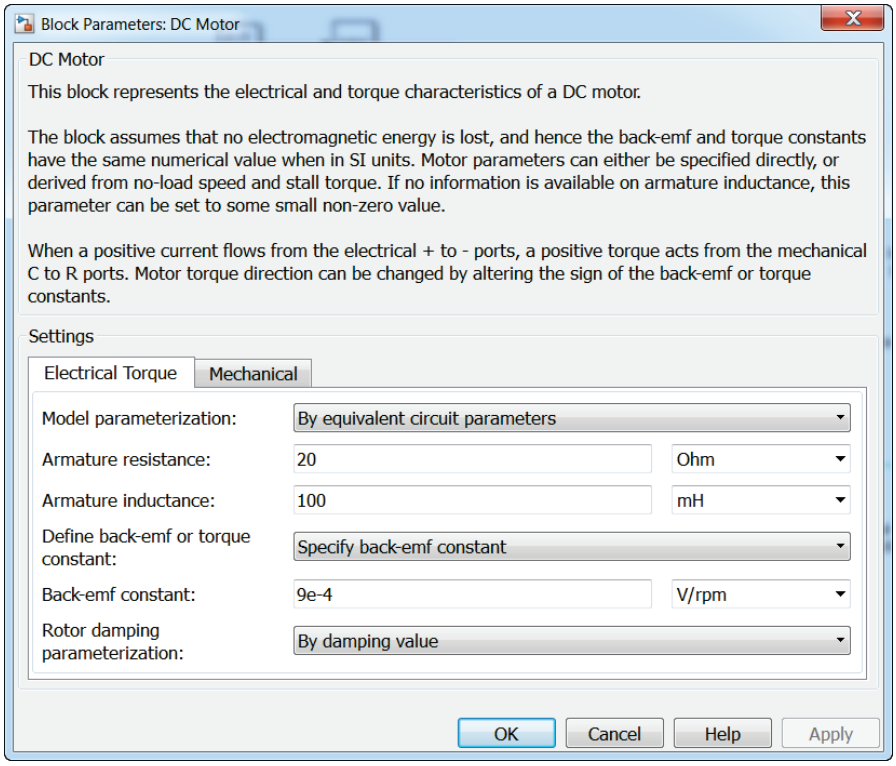
Ce modèle fait intervenir le domaine électrique et le domaine mécanique de rotation.

Paramétrage

DC MOTOR

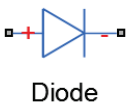


Simscape/SimElectronics/Actuators and Drivers/Rotational Actuators



Paramétrage

DIODE



Simscape/SimPowerSystems/Simscape Components/SemiConductors/Fundamental Components

Ce composant représente une diode. Le paramétrage se limite aux caractéristiques minimales du composant.

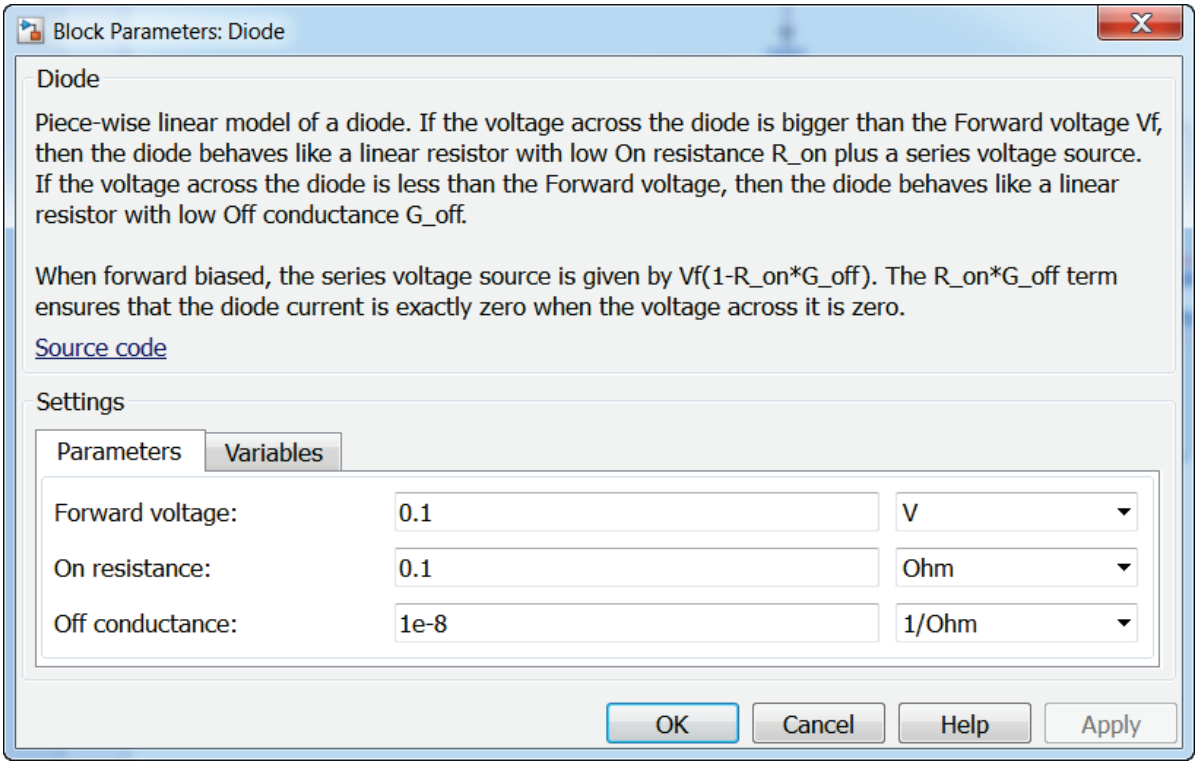


Figure 187 : paramétrage du bloc Diode

Forward Voltage : tension minimale à appliquer aux bornes de la diode pour la rendre passante (tension de seuil)

On Resistance : resistance de la diode à l'état passant

Off Conductance : conductance de la diode à l'état bloqué

Paramétrage

MOFSET

Simscape/SimPowerSystems/Simscape Components/SemiConductors/Fundamental Components

Ce composant représente un transistor de type MOFSET qui travaille en commutation. Le paramétrage se limite aux caractéristiques minimales du composant.

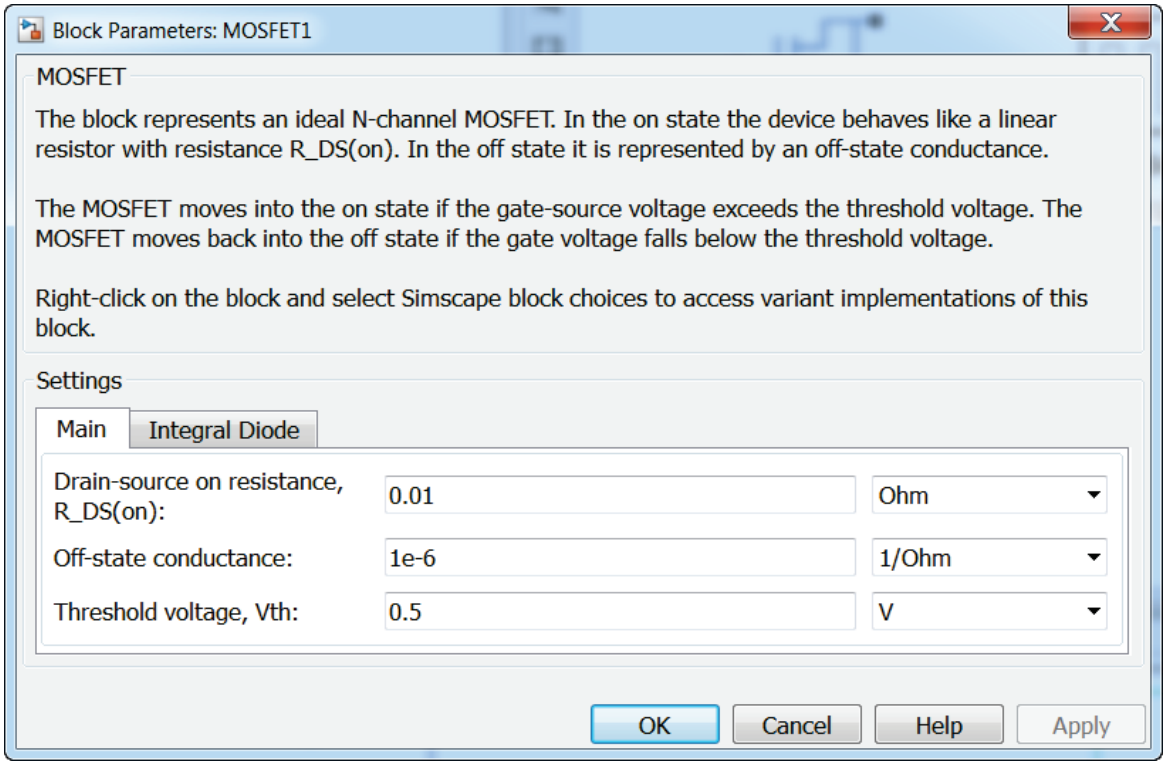


Figure 188 : paramétrage du bloc MOSFET

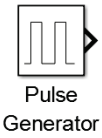
Drain-source on resistance : résistance à l'état passant

Off-State Conductance : conductance à l'état bloqué

Threshold voltage : tension de seuil de commutation

Paramétrage

PULSE GENERATOR



Simulink/Sources

Ce composant permet de modéliser une source de signaux carrés en spécifiant les caractéristiques du signal.

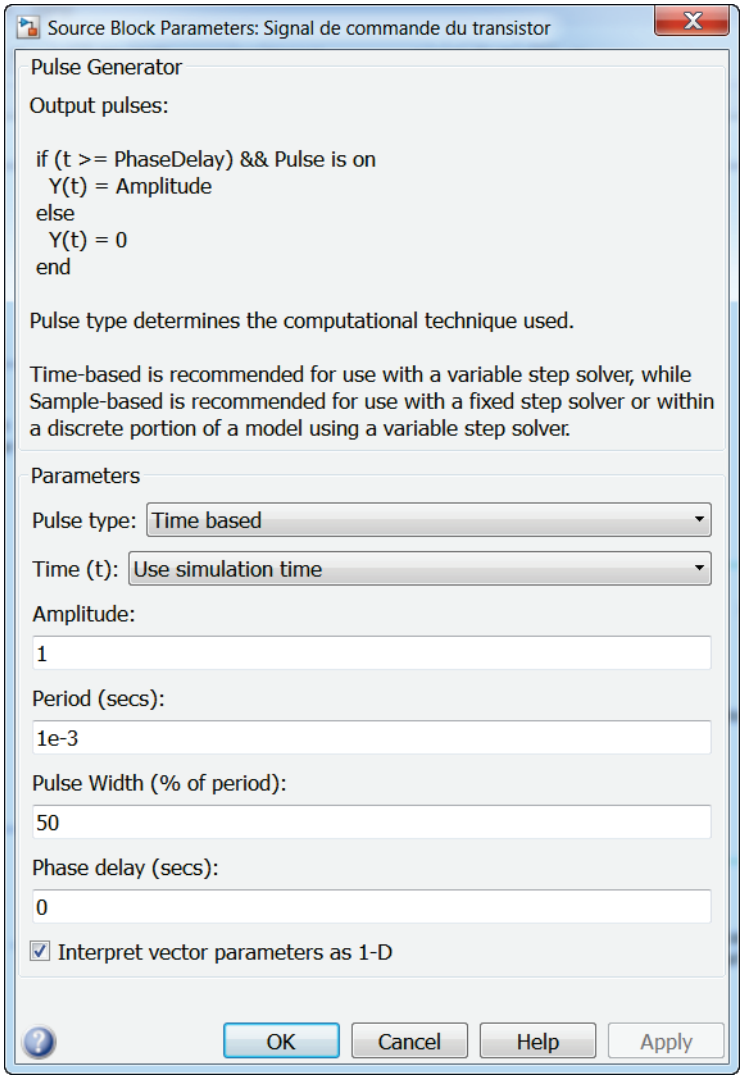


Figure 189 : paramétrage du bloc Pulse Generator

Amplitude : amplitude du signal carré

Period : période du signal carré

Pulse Width : rapport cyclique pouvant varier de 0% à 100%

Phase Delay : retard

Lancer la simulation et observer dans le scope l'évolution de la vitesse de rotation du moteur.

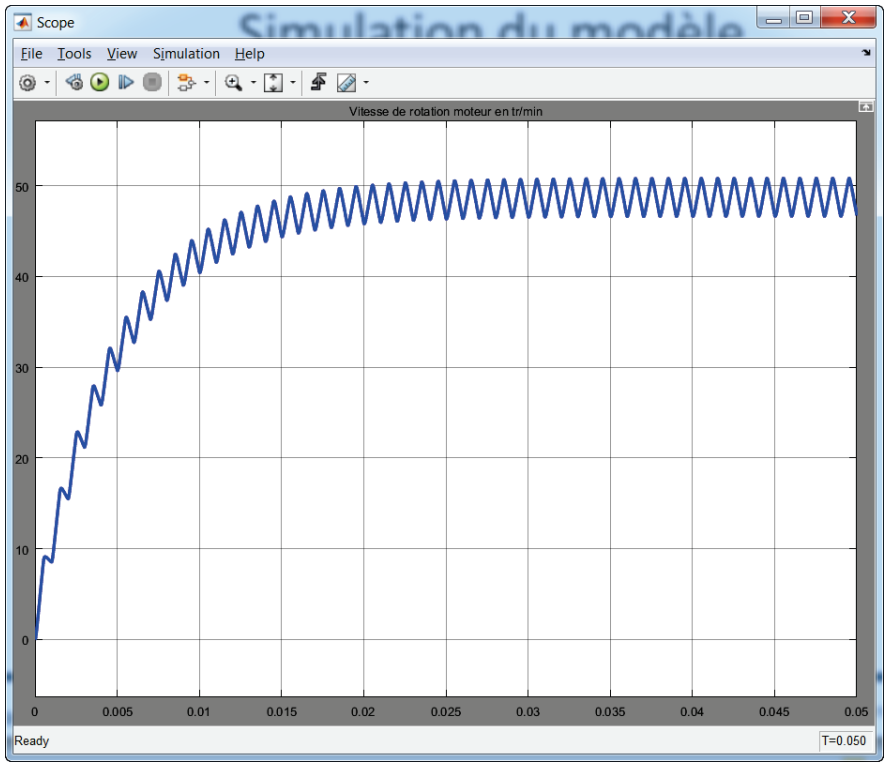


Figure 190 : évolution de la vitesse de rotation du moteur commandé par un hacheur série

La fréquence de hachage étant ici fixée à 1KHz et l'inertie de la charge étant très faible, la fréquence de commutation entraîne des irrégularités dans la vitesse de rotation du moteur.

Plusieurs étape vont maintenant nous permettre de « **didactiser** »le modèle, c'est-à-dire de le rendre accessible à des étudiants et de le rendre compatible avec les objectifs de l'apprentissage visé.

D. La didactisation du modèle

1. Création d'un sous-système et ajout d'une image

Ouvrir le modèle **« hacheur_serie_1.slx ».**
Ce modèle est le même que le précédent. Un sous-système représentant le moteur a été créé rendant plus visuel ce composant. Cette opération doit être réalisée à chaque fois que cela est possible pour améliorer la lisibilité d'un modèle.

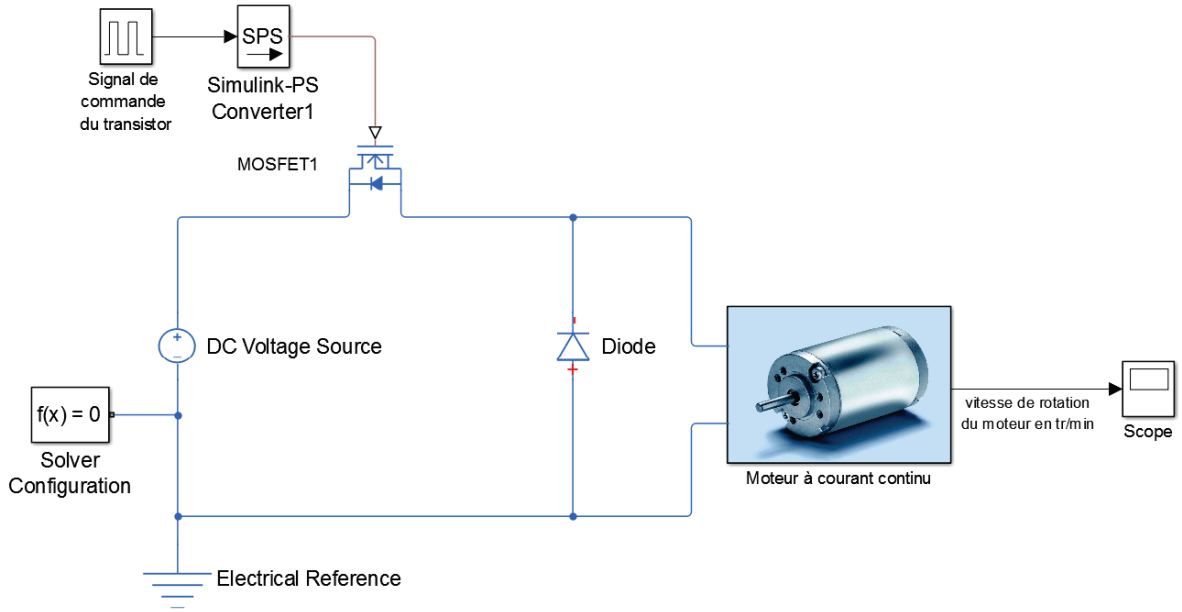


Figure 191 : création d'un sous-système pour didactiser le modèle

2. L'instrumentation du modèle

L'un des objectifs est de faire comprendre aux élèves comment circulent les courants dans le hacheur, nous allons mettre en place des capteurs de courant dans chacune des branches du circuit (Figure 192).

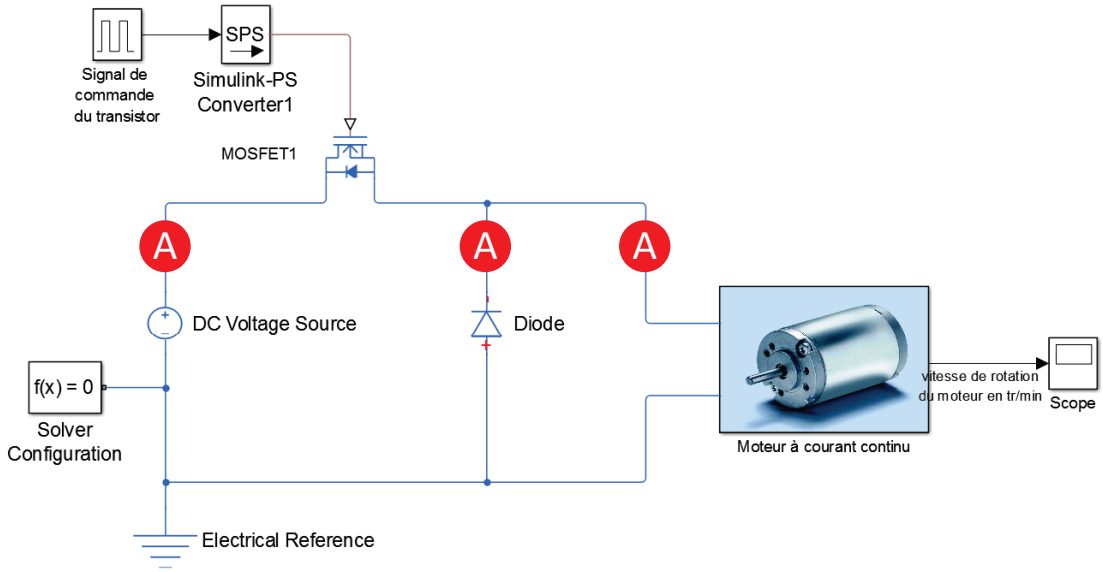


Figure 192 : mise en place des capteurs de courant dans le circuit

Nous pouvons mettre en place le premier capteur pour obtenir le modèle de la Figure 193.

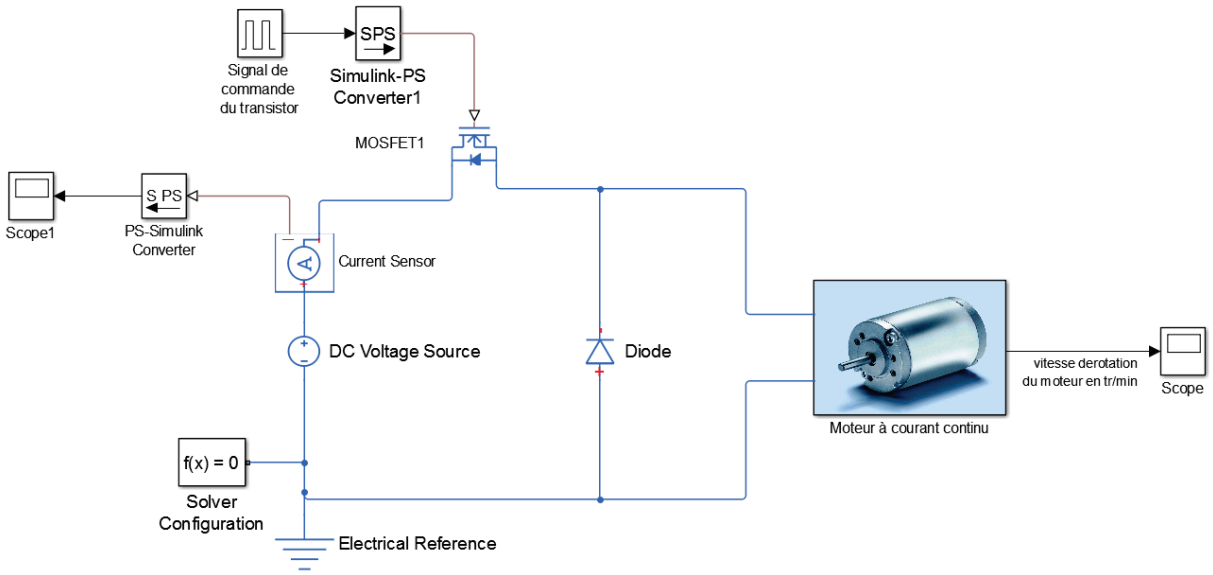


Figure 193 : ajout d'un capteur sur le modèle

Ce type d'instrumentation présente l'inconvénient majeur de surcharger le modèle en ajoutant un grand nombre de composants. Cette surcharge de composants va nuire à la lisibilité du modèle.

Afin d'optimiser cette phase et de gagner en lisibilité, il est conseillé de construire un sous-système réalisant la fonction souhaitée et de router les signaux avec des tags en les regroupant dans un même scope (les fonctionnalités de routage des signaux sont présentés page 139.

Ouvrir le fichier **« hacheur_serie_2.slx »** et observer le travail qui a été effectué pour instrumenter le modèle.

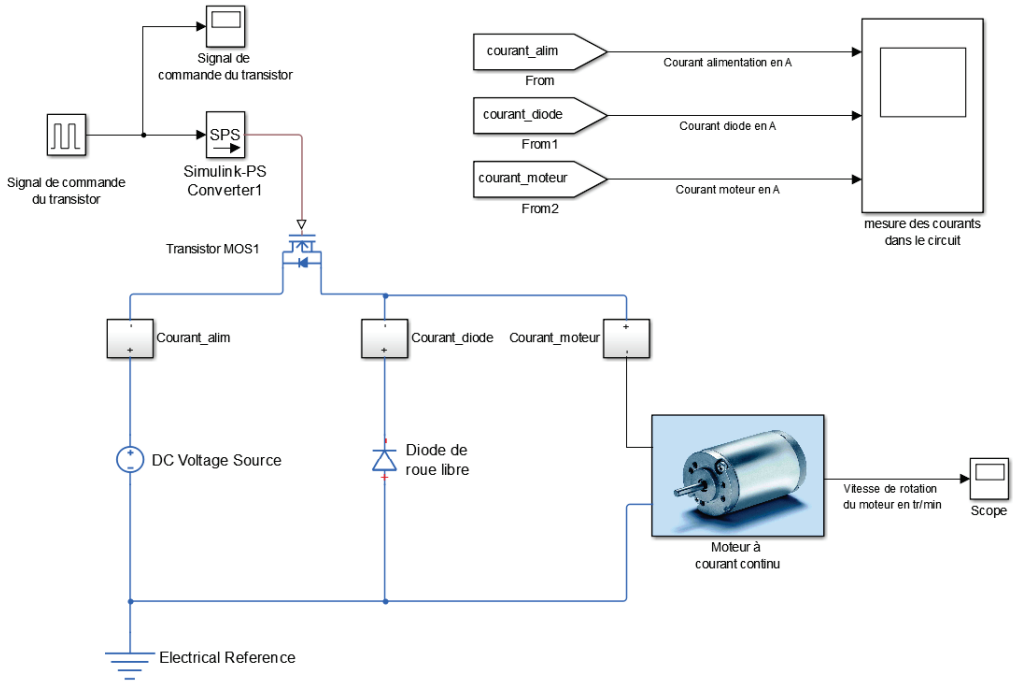


Figure 194 : modèle instrumenté et didactisé

Double-cliquer sur le capteur de courant **« Courant_alim »** et observer le contenu du sous-système (Figure 195).

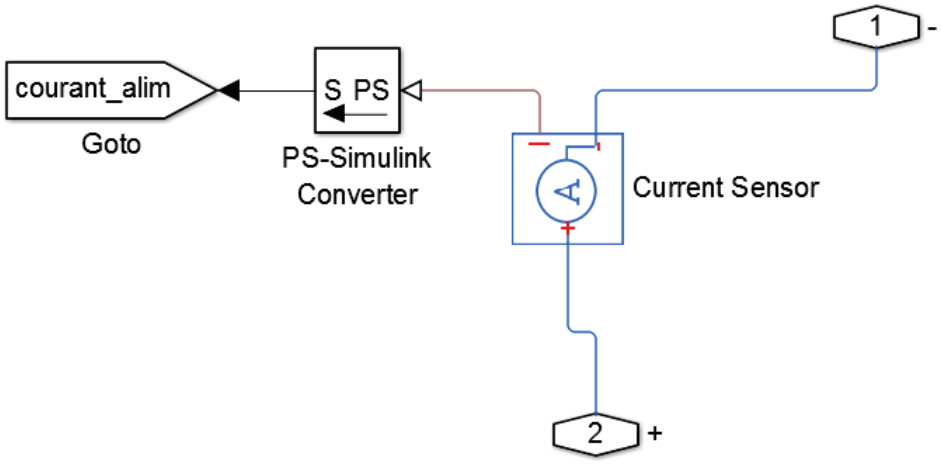


Figure 195 : sous-système « capteur de courant »

Ce capteur permet de relever l'intensité du courant dans une branche du circuit et route le signal à l'aide d'un tag qui sera repris par le scope.

L'avantage est que cette structure peut être recopiée et utilisée dans les deux autres branches du circuit afin de minimiser l'ajout de composants sur le modèle.

A ce stade le modèle devient exploitable par l'étudiant.

Lancer la simulation et visualiser l'allure des courants dans les différentes branches du circuit.

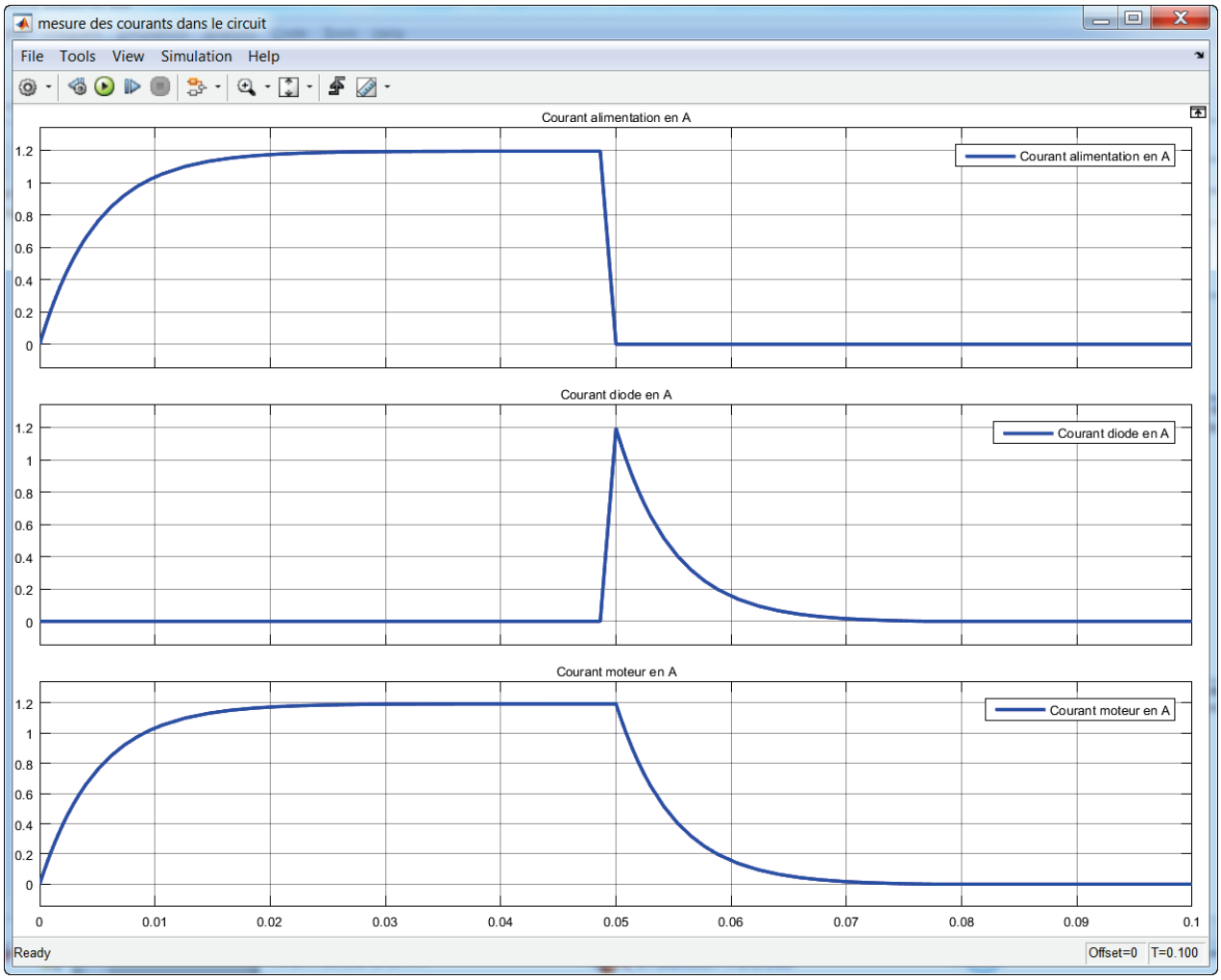


Figure 196 : évolution du courant dans les trois branches du circuit

Rappel :

Objectif 1 : Comprendre la circulation du courant dans le circuit en phase active et en phase de roue libre.

L'étudiant peut alors facilement faire le lien entre les résultats obtenus en simulation et le sens du passage du courant dans les différentes branches du circuit. La phase active et la phase de roue libre peuvent être identifiées. L'objectif 1 d'apprentissage peut être atteint.

3. Conclusion sur la didactisation du modèle

Les différentes phases d'amélioration de la lisibilité du modèle, que nous avons menées, participent à la « didactisation du modèle ». C'est une phase importante du travail qui permet de passer d'un modèle difficilement lisible par les étudiants à un modèle parfaitement lisible. Ces différentes étapes demandent un travail supplémentaire de la part de l'enseignant mais sont indispensables à la réussite de la séquence.

Le modèle didactisé :

- Il est construit pour permettre une prise en main rapide par un élève
- Il est facilement modifiable
- Il permet à l'élève de se focaliser sur l'objectif d'apprentissage visé
- Il doit dans la mesure du possible reproduire le descripteur

Le modèle non didactisé :

- Il donne les mêmes informations que le modèle didactisé
- L'objectif est centré sur les résultats exploitables
- Il est difficile à lire et à modifier
- Il ne doit pas être donné aux élèves

La Figure 197 montre le modèle non didactisé et permet de comprendre les difficultés de lecture et d'exploitation que l'utilisation de ce modèle pourraient engendrer.

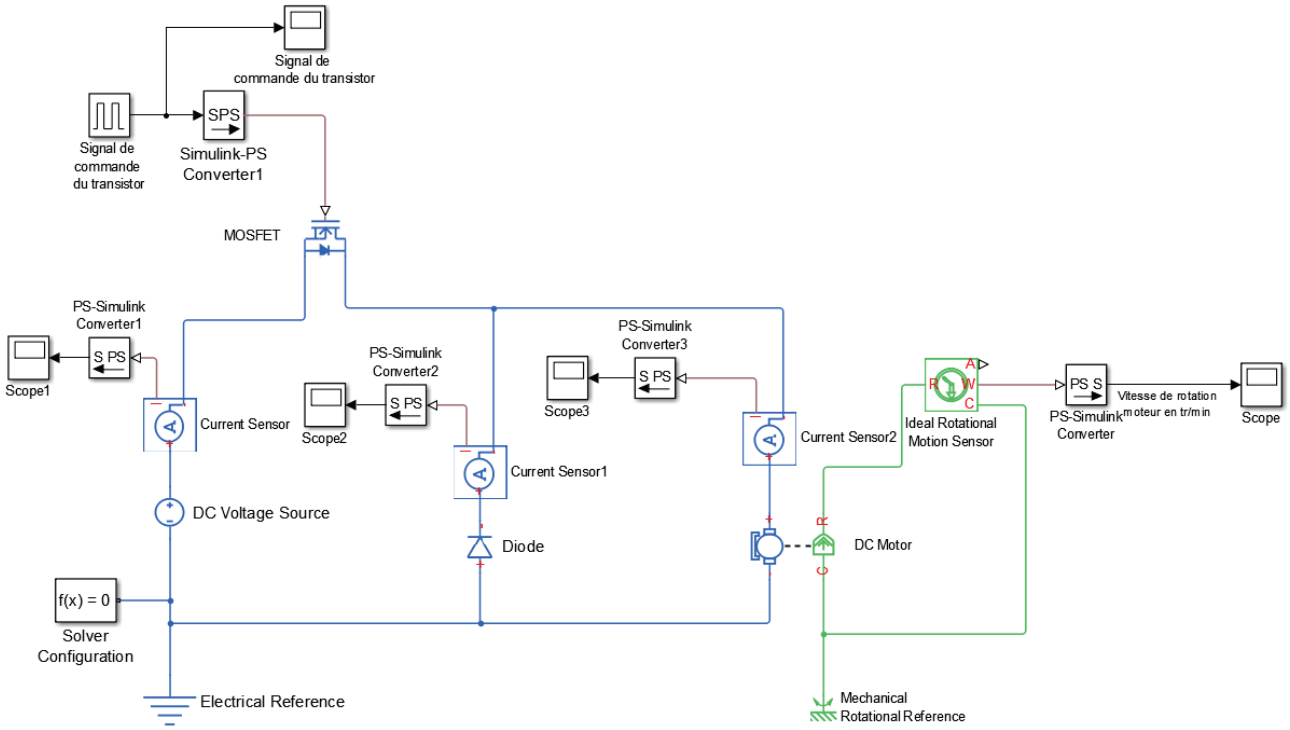


Figure 197 : le modèle non didactisé

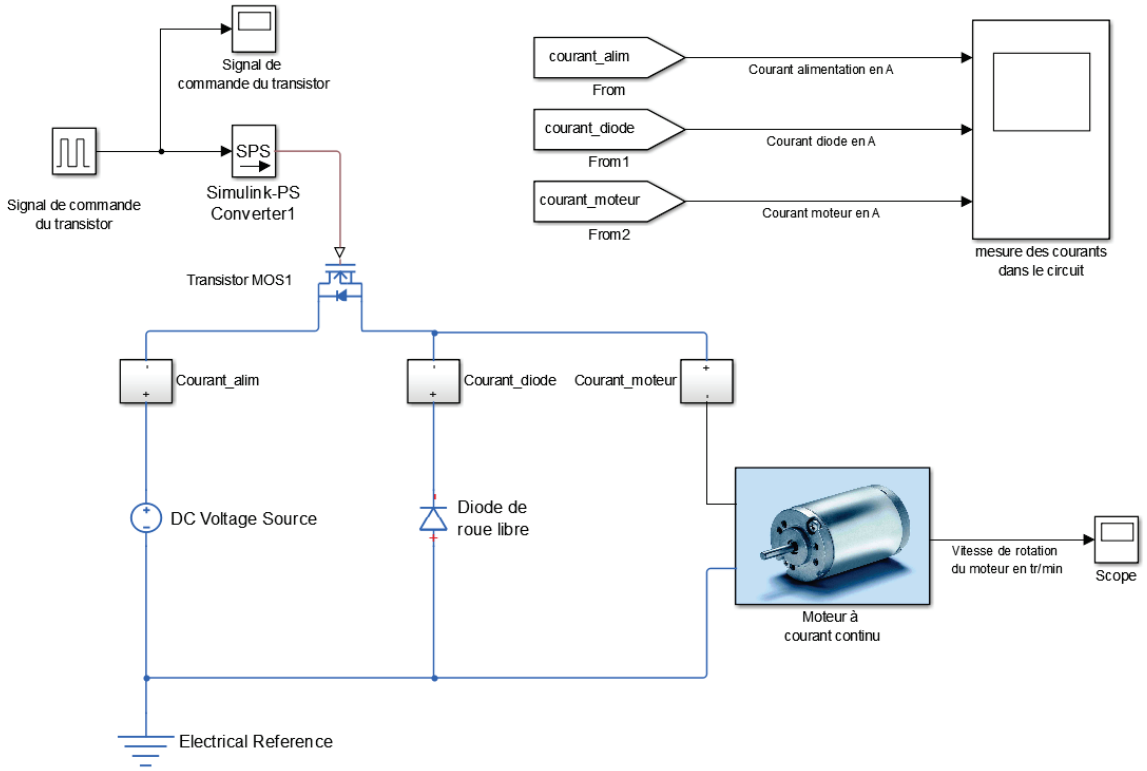


Figure 198 : le modèle didactisé

4. Optimiser la didactisation du modèle en fonction de l'objectif d'apprentissage visé

Nous avons vu que la forme du modèle didactisé précédemment établie est parfaitement adaptée à la compréhension du premier objectif d'apprentissage de la séquence qui était de de permettre aux élèves de comprendre la circulation du courant dans le hacheur série en phase active et en phase de roue libre.

Afin de poursuivre la séquence, et passer à l'objectif d'apprentissage suivant, il est nécessaire d'adapter la didactisation de notre modèle et de s'interroger sur les améliorations éventuelles à apporter pour atteindre les objectifs suivants.

Rappel :

Objectif 2, 3 et 4 : Visualiser et évaluer l'influence des différents paramètres sur l'ondulation du courant.

- Rapport cyclique
- Fréquence de hachage
- Valeur de l'inductance de la charge

Il faut focaliser l'attention des étudiants sur le courant moteur pour les objectifs suivants, les deux capteurs de courant dans les autres branches peuvent être supprimés. De plus il est nécessaire de disposer de la valeur moyenne du courant pour avoir une image du couple moteur (Figure 199).

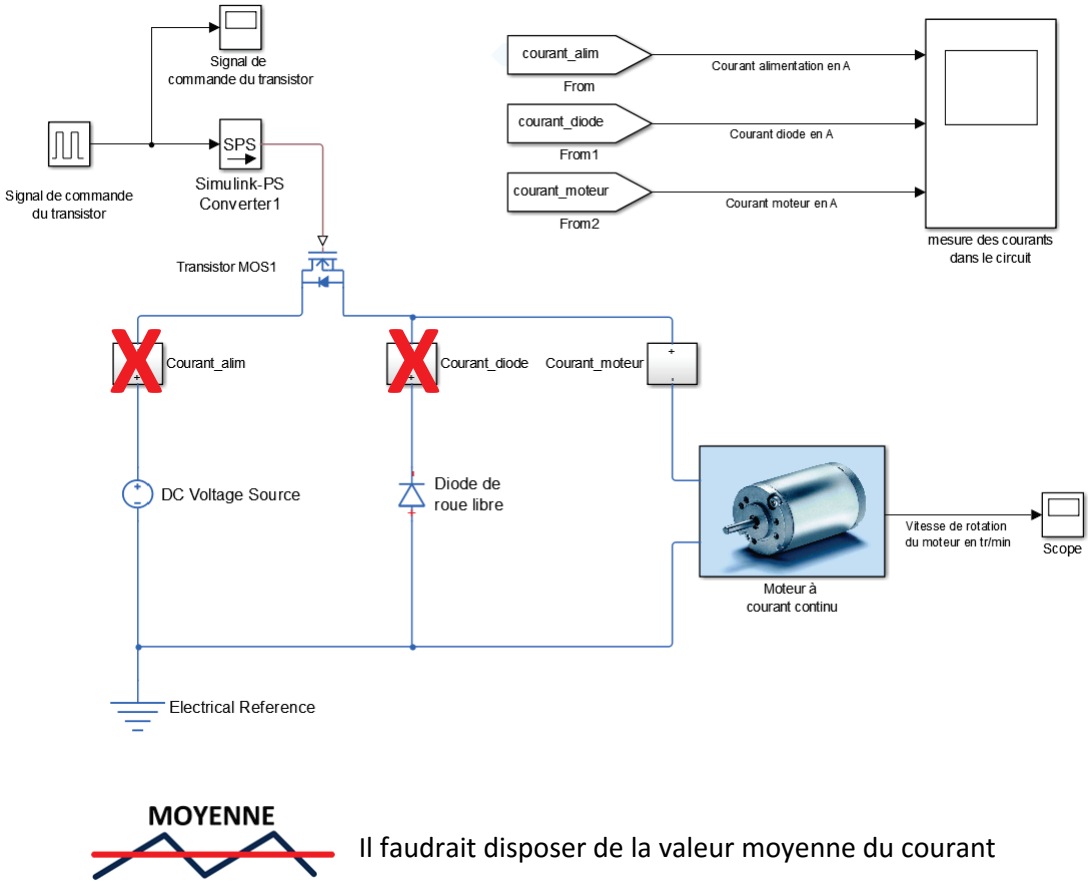
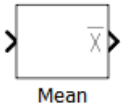
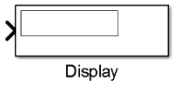


Figure 199 : amélioration de la didactisation du modèle

Le bloc **Mean** permet de calculer automatiquement la valeur moyenne d'un signal.

Fonction du composant	Représentation	Bibliothèque
Calcul de la valeur moyenne		Simscape/SimPowerSystems/Specialized Technology/Control and Measurements Library/Measurements
Affichage		Simulink/Sinks

Ouvrir le fichier « **hacheur_serie_3.slx** ». Ce fichier contient le modèle didactisé avec les améliorations proposées. Un **masque** a été mis sur le bloc **Mean** pour rendre plus explicite sa fonction (Figure 200).

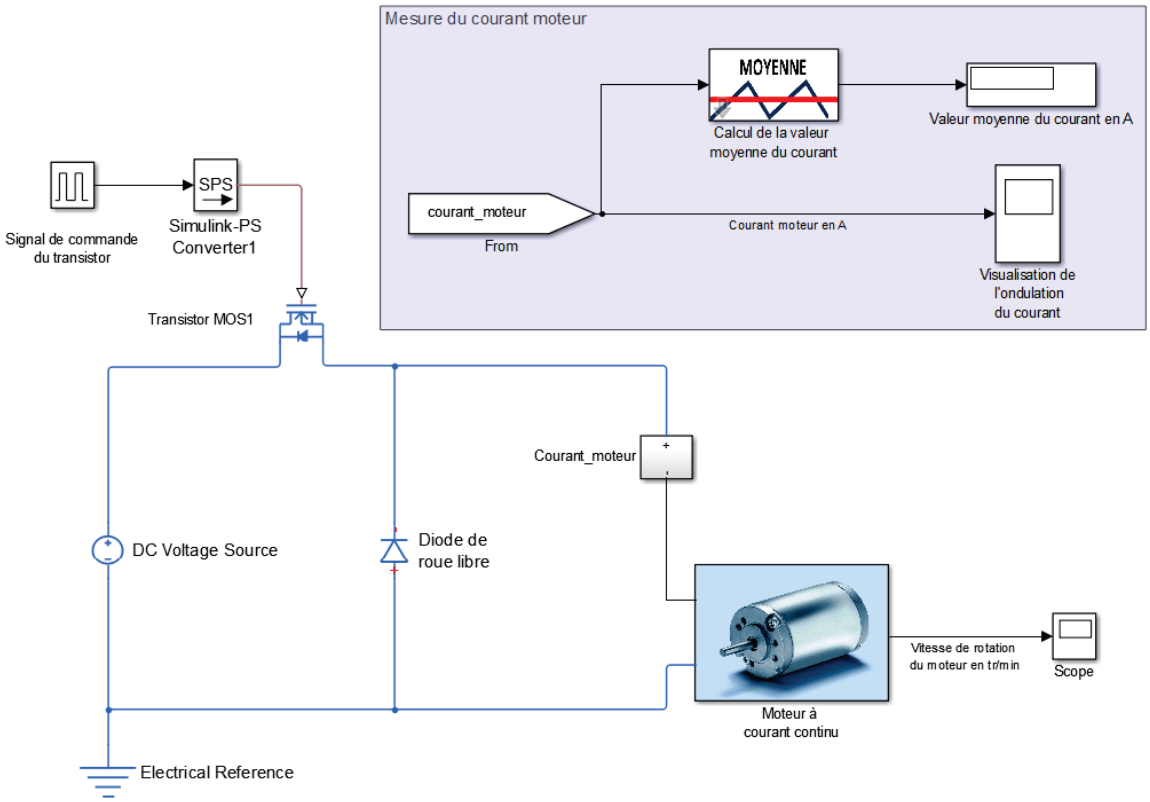


Figure 200 : le modèle didactisé et optimisé

Double cliquer sur le bloc **Signal de commande du transistor**.

Régler la **période de commutation** du transistor sur **0.01 s** et spécifier un **temps de simulation** de **0.1 s.**

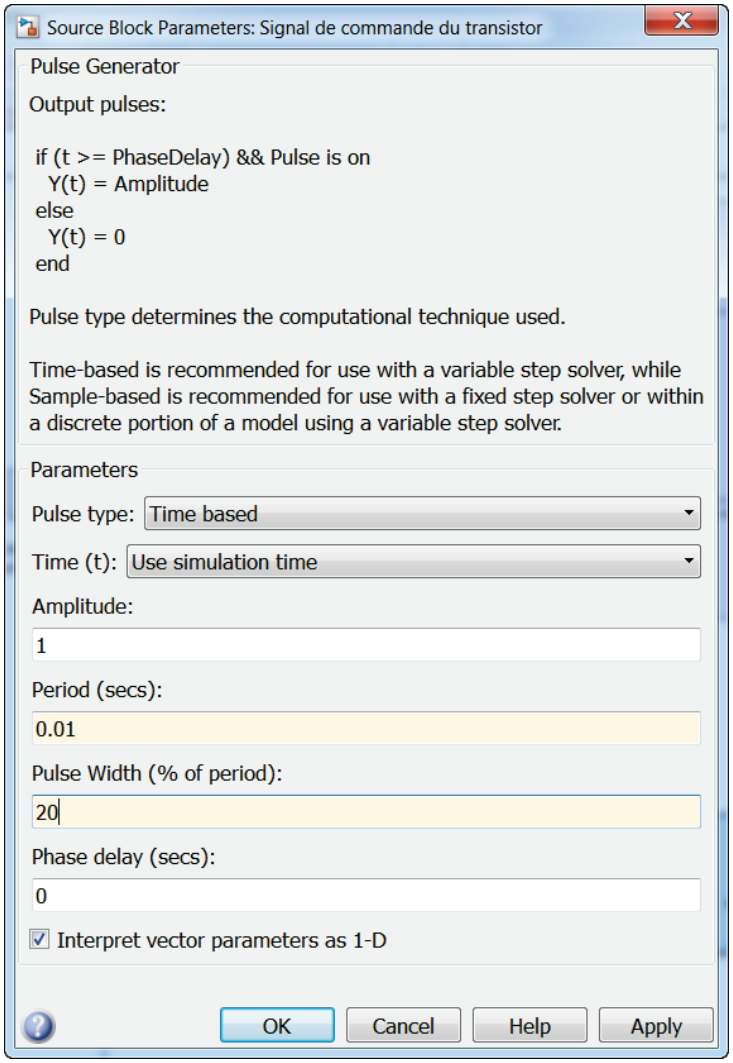
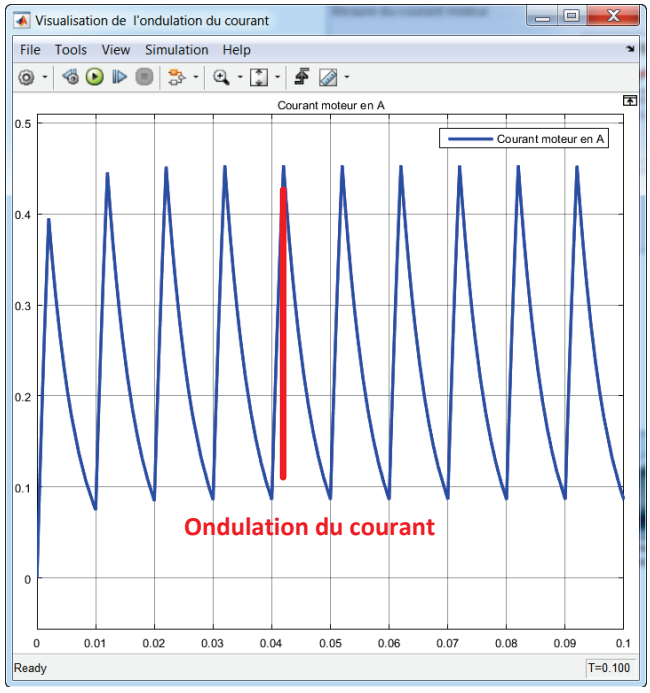
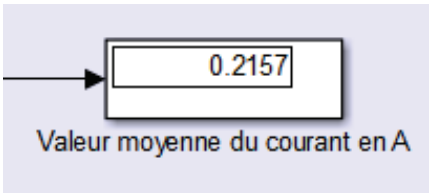


Figure 201 : réglage de la période de commutation du transistor

Lancer la simulation et constater que la valeur moyenne et l'allure du courant instantané sont maintenant disponibles dans le modèle. La valeur instantanée du courant est visualisable dans le scope et la valeur moyenne dans l'afficheur (Figure 202).



Valeur instantanée du courant moteur avec visualisation de l'amplitude de l'ondulation de courant



Valeur moyenne du courant moteur

Figure 202 : visualisation de la valeur moyenne et instantanée du courant moteur

Nous pouvons constater que les informations nécessaires à l'atteinte des objectifs sont très facilement accessibles par les étudiants. Ils pourront alors faire varier les paramètres du modèle comme la fréquence de hachage (période de commutation du transistor) ou le rapport cyclique.

Pour visualiser l'influence de l'inductance de la charge, il faudra faire une modification du modèle en insérant une inductance en série avec le moteur.

Ouvrir le fichier « **hacheur_serie_4.slx** ». Ce fichier contient le modèle précédent auquel a été ajoutée une inductance en série avec le moteur. L'étudiant pourra alors faire varier la valeur de l'inductance pour visualiser son influence sur le courant moteur (Figure 203).

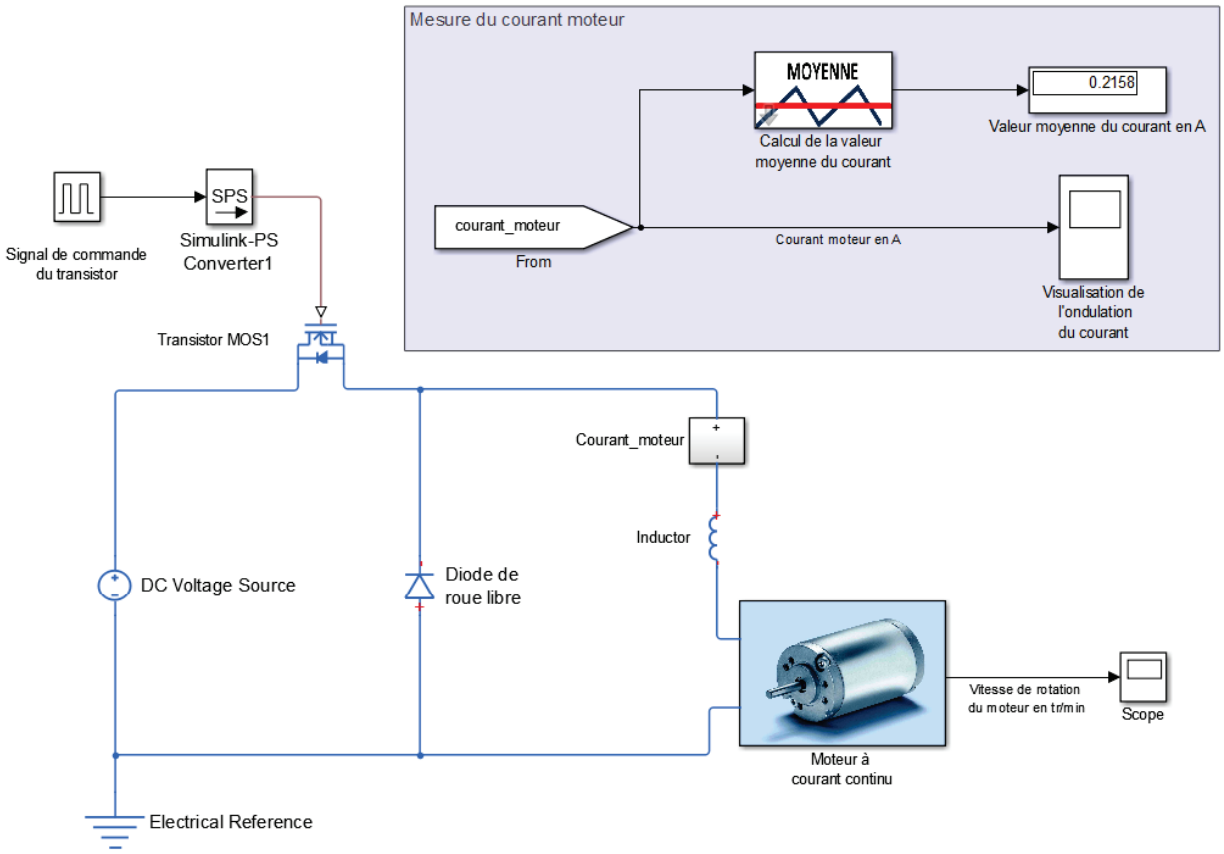


Figure 203 : ajout d'une inductance de lissage en série avec le moteur

E. Exploitation des résultats issus de la simulation du modèle

Dans cette partie, les résultats issus de la simulation des différents modèles sont présentés et montre ce que l'étudiant doit être capable de mettre en évidence au travers de l'exploitation des différents modèles.

1. Objectif 1 : Comprendre la circulation du courant dans le circuit en phase active et en phase de roue libre

La simulation du modèle **« hacheur_serie_2.slx »** (Figure 204) permet de mettre en évidence les résultats suivants :

- En phase active, le courant qui traverse le moteur est identique à celui qui traverse l'alimentation. Le courant dans la branche de la diode est nul.

- En phase de roue libre, le courant qui traverse le moteur est identique à celui qui traverse la branche de la diode. Le courant qui traverse l'alimentation est nul.

Circulation du courant

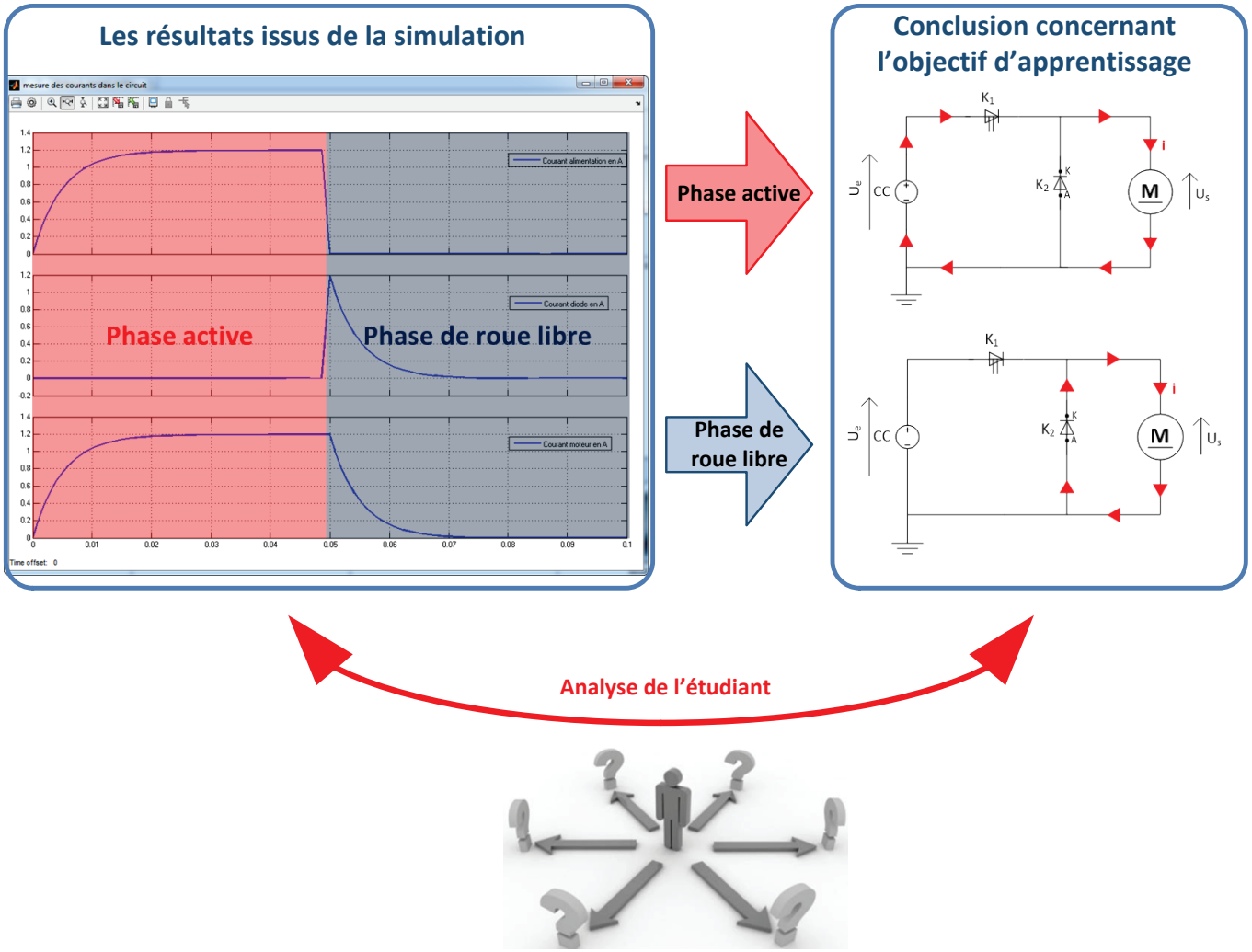


Figure 204 : exploitation et visualisation de la circulation du courant dans le hacheur série

2. Objectif 2 : Visualiser et évaluer l'influence du rapport cyclique sur le courant moteur

La simulation du modèle **« hacheur_serie_3.slx »** (Figure 206) permet de mettre en évidence l'influence du rapport cyclique.

Régler la période de commutation du transistor sur **0,0001 s**
Régler le rapport cyclique sur $\boldsymbol{\alpha = 50\%}$ (Figure 205)

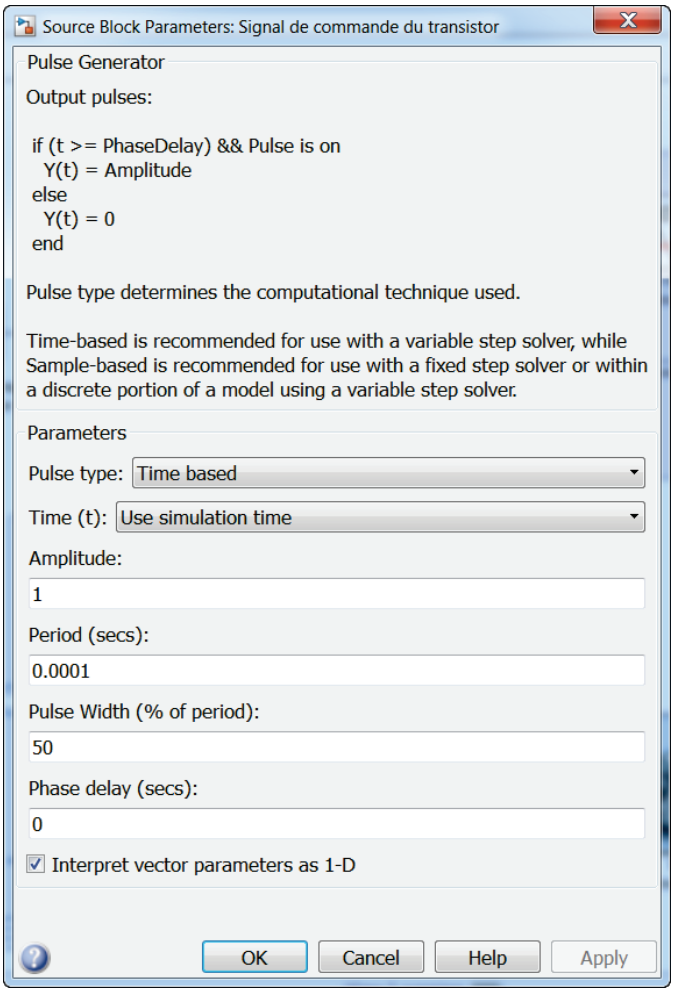


Figure 205 : réglages des paramètres de la simulation

Lancer la simulation et relever l'allure instantanée du courant moteur et sa valeur moyenne.
Relancer la simulation en modifiant la valeur du rapport cyclique sur $\boldsymbol{\alpha = 90\%}$**.**

Influence du rapport cyclique

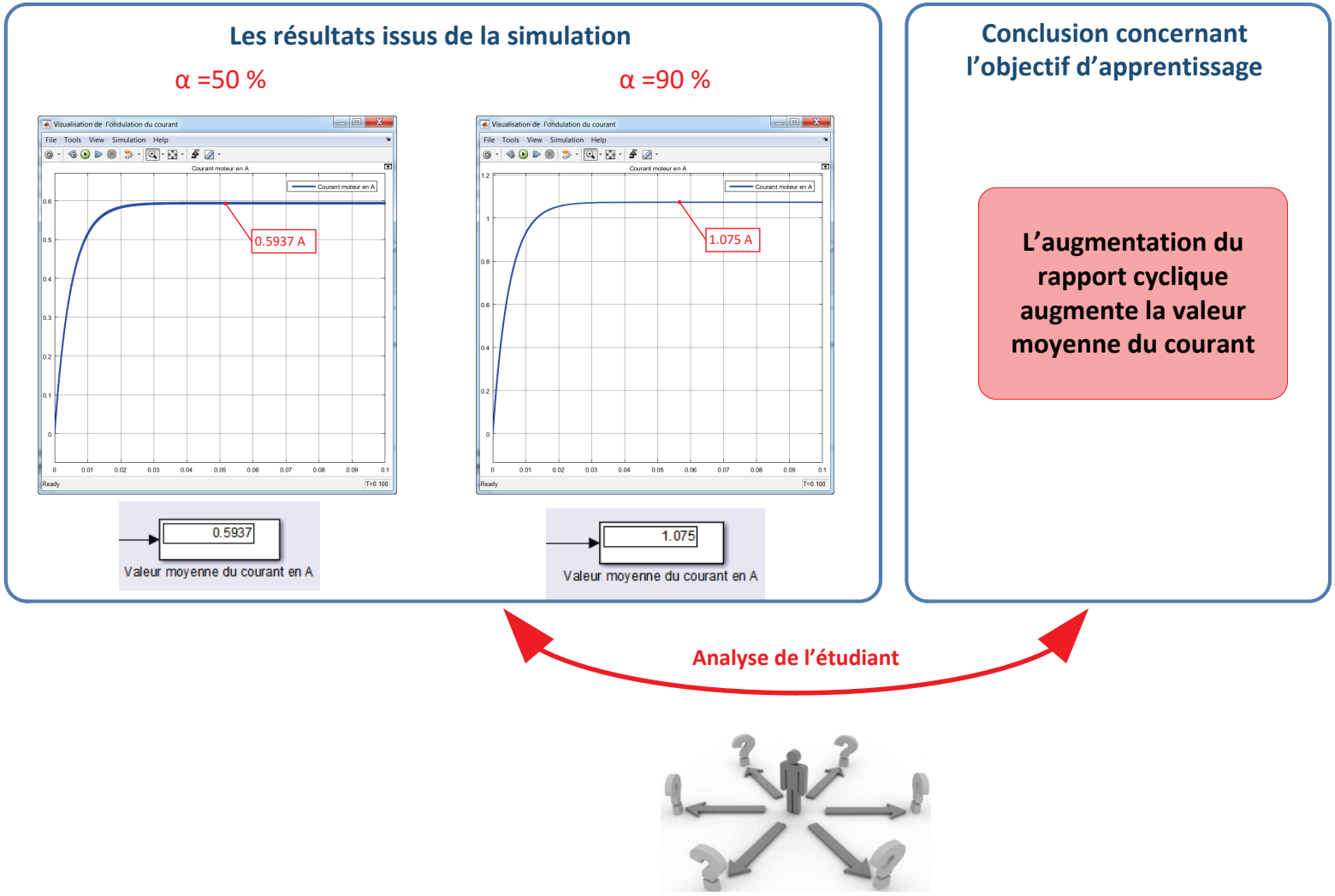


Figure 206 : exploitation et visualisation de l'influence du rapport cyclique sur le courant moteur

3. Objectif 3 : Visualiser et évaluer l'influence de la fréquence de hachage sur l'ondulation du courant

La simulation du modèle **« hacheur_serie_3.slx »** (Figure 208) permet de mettre en évidence l'influence du rapport cyclique.

Régler la période de commutation du transistor sur **0,001 s**
Régler le rapport cyclique sur $\boldsymbol{\alpha = 50\%}$

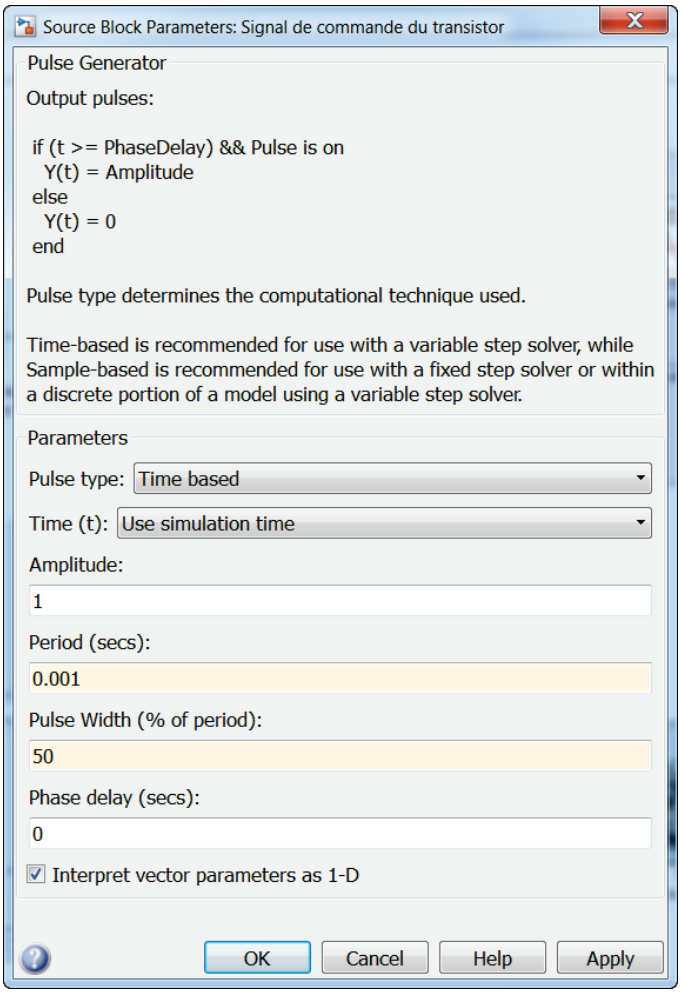


Figure 207 : réglage des paramètres de la simulation

Lancer la simulation et relever l'allure instantanée du courant moteur et sa valeur moyenne.
Relancer la simulation en modifiant la période de commutation sur **T=0.0001 s**

Influence de la fréquence de hachage

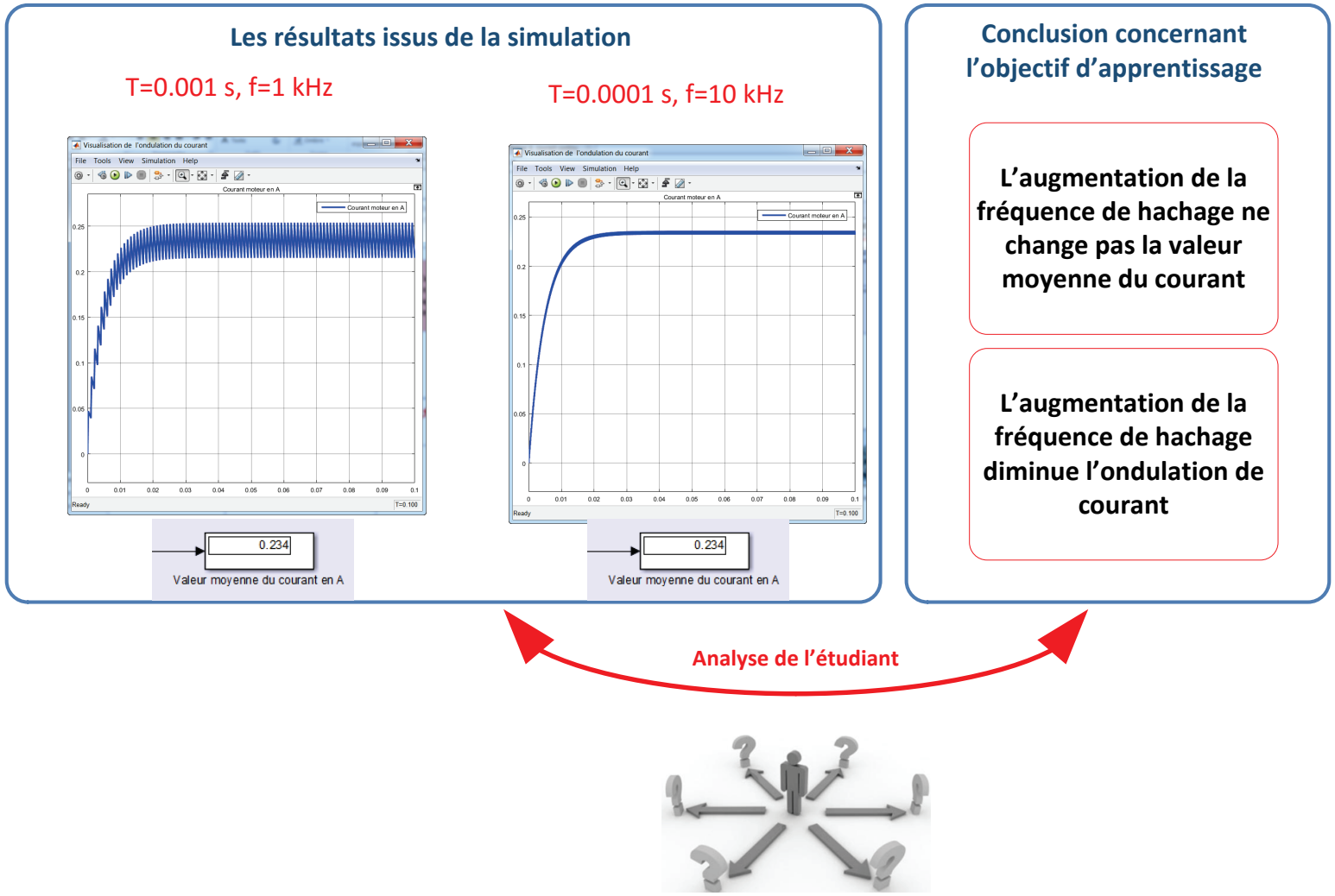


Figure 208 : exploitation et visualisation de l'influence de la fréquence de hachage sur le courant moteur

4. Objectif 4 : Visualiser et évaluer l'influence de l'inductance de la charge sur l'ondulation du courant

La simulation du modèle **« hacheur_serie_4.slx »** (Figure 210) permet de mettre en évidence l'influence de l'inductance de la charge.

Régler la valeur de l'inductance ajoutée à **1 mH**

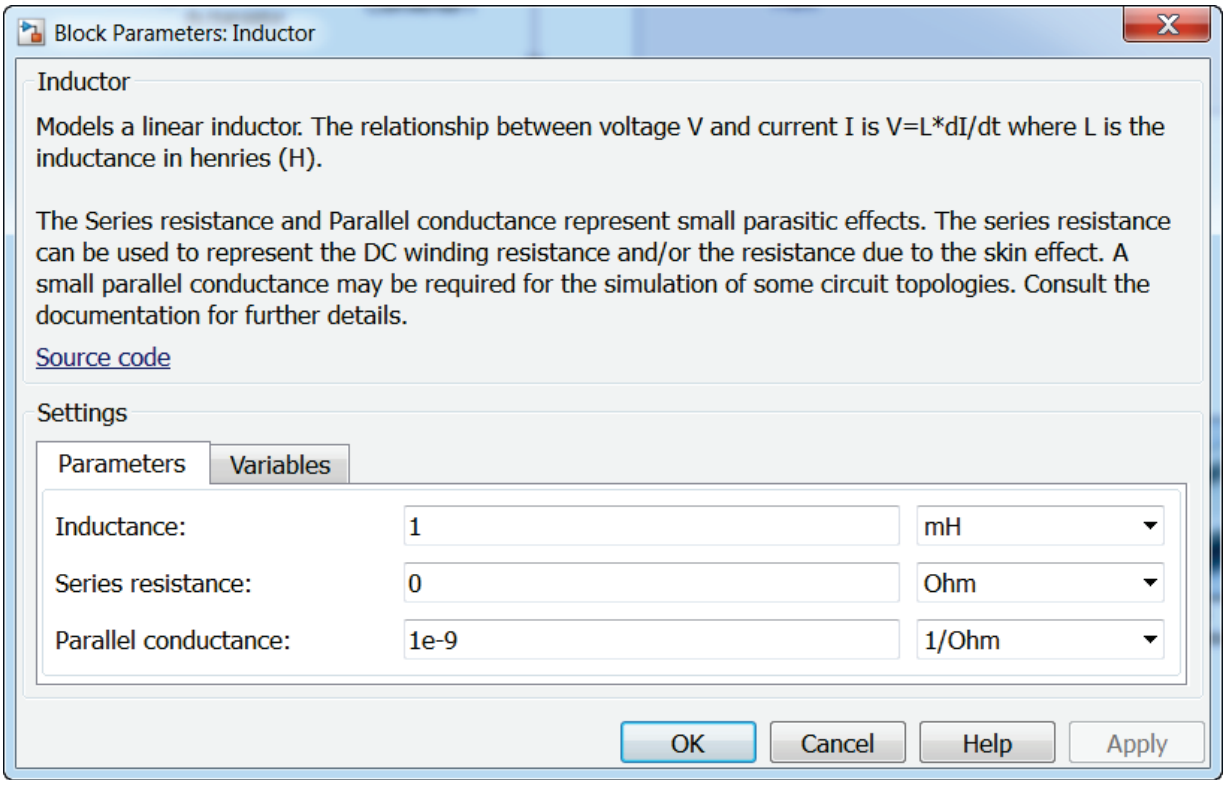


Figure 209 : réglage des paramètres de la simulation

Lancer la simulation et relever l'allure instantanée du courant moteur et sa valeur moyenne.
Relancer la simulation en modifiant la valeur de l'inductance à **100 mH**

Influence de l'inductance de la charge

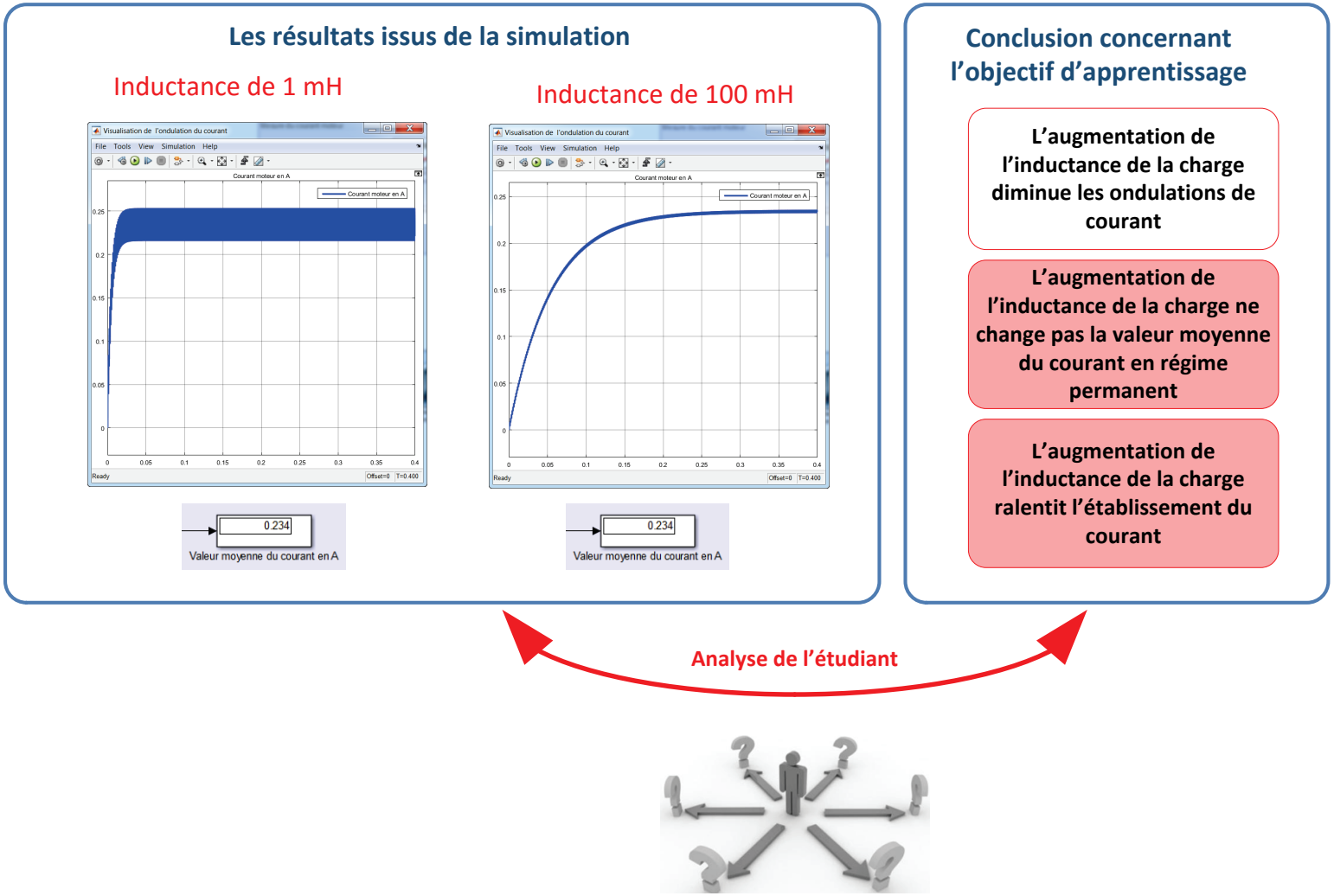


Figure 210 : visualisation et exploitation de l'influence de l'inductance de la charge sur le courant moteur

F. Conclusion

Le succès d'une séquence pédagogique ne se limite pas à la construction des modèles et à leur simulation. Un travail de didactisation doit être effectué sur le modèle. Cette didactisation doit permettre d'optimiser la forme du modèle à l'objectif d'apprentissage visé. Les étudiants sont alors dans les meilleures conditions pour focaliser leur attention sur l'atteinte des objectifs d'apprentissage.

Chapitre 3 : Prise en main de MATLAB

I. Introduction

Ce chapitre a pour objectif de présenter les fonctionnalités de base de **MATLAB** et de manipuler les commandes les plus utiles pour débuter. Les possibilités de calcul de **MATLAB** étant très vastes, il s'agit ici de présenter les commandes de bases qui peuvent être utiles dans le cadre des calculs orientés Sciences de l'Ingénieur.
Toutes les commandes qui suivent seront saisies dans la « fenêtre de commande » **MATLAB**. Chaque ligne de commandes doit être validée par la touche «entrée » du clavier pour être prise en compte par le logiciel.

A. Création de variable

Création de la variable a :

```
>> a=1 (valider)

a =

   1
```

MATLAB fait la différence entre majuscule et minuscule **a** et **A** seront deux variables distinctes.
On remarque que MATLAB répond qu'il vient de créer la variable **a**. Cette nouvelle variable apparaît dans la fenêtre du **Workspace** et la commande que l'on vient de saisir apparaît dans la fenêtre **Command History**.

Création de la variable b :

```
>> b=5 ;
```

Le point-virgule à la fin de la ligne de commande indique que l'on ne souhaite pas de réponse à notre commande. La variable b apparaît dans le **Workspace**.

Il est facile de réaliser une opération entre deux variables, par exemple une multiplication.

```
>> a*b

ans =

   5
```

En appuyant sur les flèches haut et bas du clavier, il est possible de rappeler toutes les commandes de la fenêtre **Command History**.

B. Création de vecteur

Considérons la série de données du tableau de Figure 211.

x	0	10	20	30	40	50
y1	0	16	35	49	25	6
y2	0	12	21	36	49	56

Figure 211 : tableau de données

Si l'on souhaite exploiter ces données il faut les saisir dans MATLAB sous la forme de vecteurs.

Le séparateur peut être un espace :

```
>> x=[0 10 20 30 40 50] ;
```

Le séparateur peut être une virgule :

```
>> y1=[0,16,35,49,25,6];
>> y2=[0 12 21 36 49 56];
```

Chaque composante i du vecteur x est stockée dans la variable x(i).

```
>> x(2)
ans =
   10
```

MATLAB propose également des fonctions de création automatique de vecteurs

```
>>u=[8 :0.1 :50]
```

Cette commande permet de créer le vecteur **u** qui aura comme composantes toutes les valeurs comprises entre **8** et **50** avec un pas de **0.1**. L'utilisation de cette commande permet de spécifier l'espacement entre deux composantes, le nombre de composantes étant calculé en fonction du pas et des bornes spécifiées.

Si le pas n'est pas spécifié, il prend par défaut la valeur de 1.

```
>>u=[8 :50]
```

Cette commande permet de créer le vecteur **u** qui aura comme composantes toutes les valeurs comprises entre **8** et **50** avec un pas de **1**.

Si au contraire, on souhaite spécifier le nombre exact de composantes, la fonction *linespace* peut être utilisée.

```
>>u=linspace(8,50,1000)
```

Cette commande permet de créer le vecteur u qui aura 1000 composantes régulièrement espacées entre les bornes 8 et 50.

C. Indexation des composantes d'un vecteur

Il est possible d'accéder facilement aux composantes d'un vecteur en utilisant l'indexation par ligne.
La variable **u(i)** contient la valeur de la $i^{ème}$ composante du vecteur **u**.

```
>>u(56)
ans=
10.313
```

D. Tracés de courbes

Deux vecteurs de même longueur peuvent être tracés à l'aide de la commande *plot* :

```
>> plot(x,y1)                    trace le vecteur y1 en fonction du vecteur x
```

Une fenêtre graphique « Figure 1 » s'ouvre avec la courbe demandée.

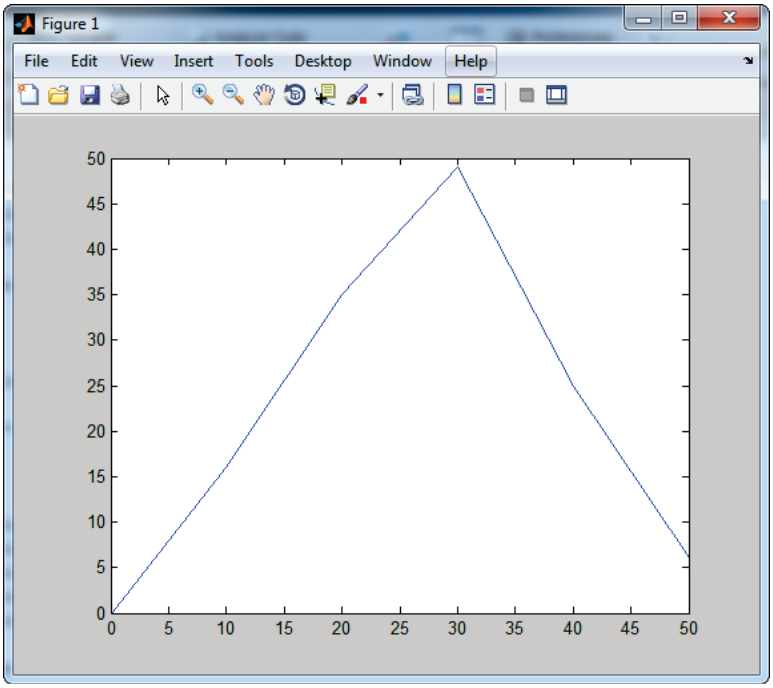


Figure 212 : tracé de deux vecteurs

On peut tracer le vecteur y2 en fonction du vecteur x

```
>> plot(x,y2)                    trace le vecteur y2 en fonction du vecteur x
```

On remarque que sur la fenêtre « Figure 1 », seule la courbe y2 en fonction de x est affichée. La courbe précédente a été supprimée.

Pour conserver les courbes dans une même fenêtre, on peut utiliser la commande *hold on*.

```
>> hold on
>> plot(x,y1)
```

La commande *hold off* annule la commande *hold on.*

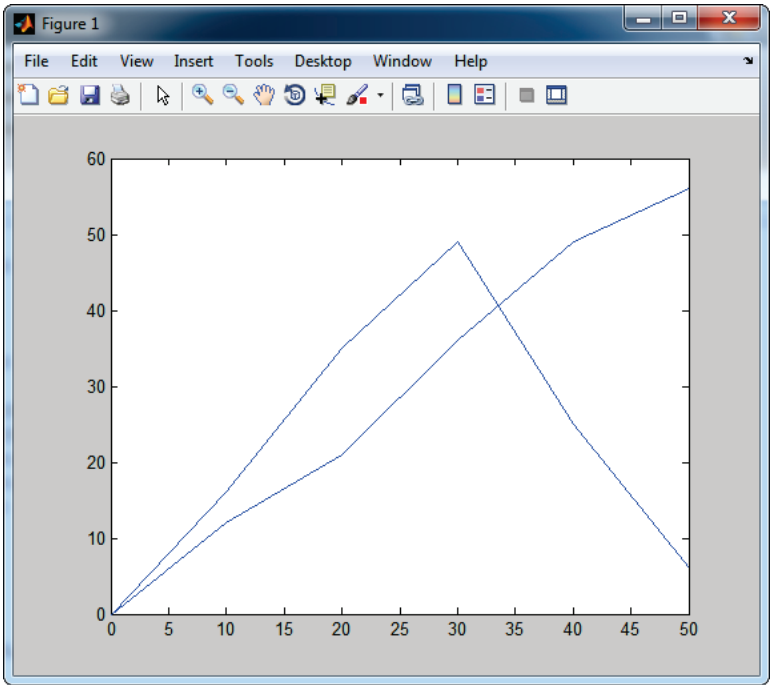


Figure 213 : tracé de deux courbes dans la même fenêtre graphique

La commande *grid on* permet d'afficher la grille

```
>>grid on
```

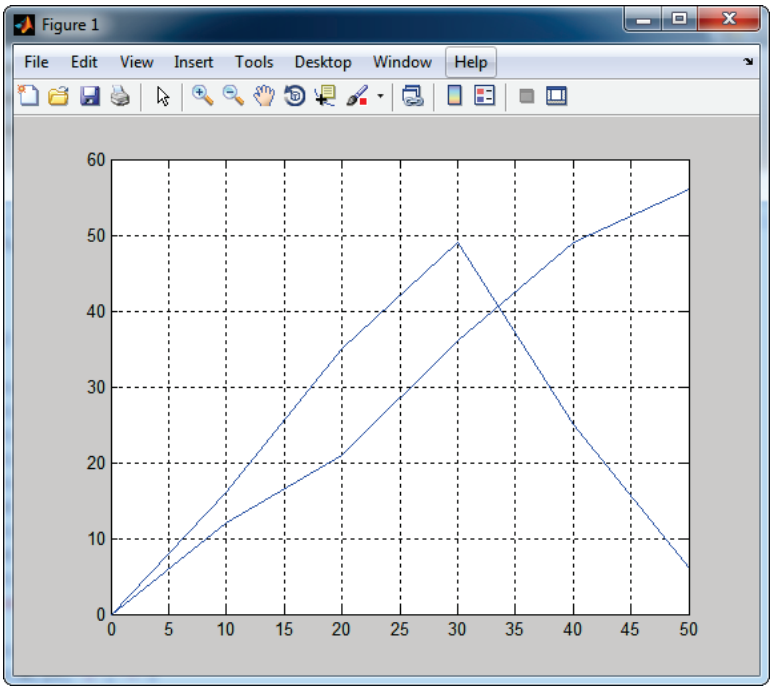


Figure 214 : affichage de la grille sur un graphique

La commande *grid off* permet de masquer la grille.

Si l'on souhaite ouvrir une nouvelle fenêtre graphique, il faut utiliser la commande *figure*

```
>> figure
```

Une fenêtre graphique vide nommée figure2 s'ouvre et il est possible travailler dans cette nouvelle fenêtre (Figure 215).

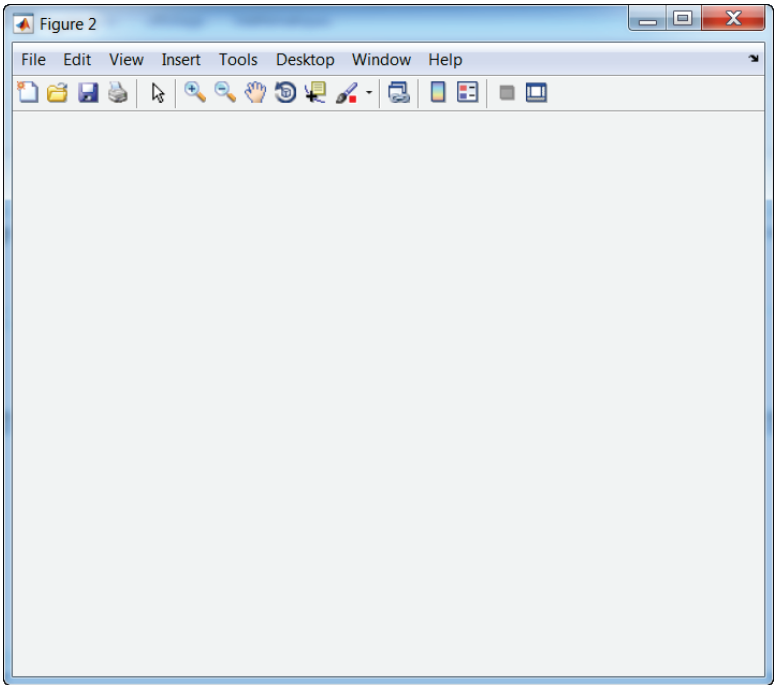


Figure 215 : ouverture d'une nouvelle fenêtre graphique

E. Mise en forme élémentaires des courbes

Des fonctions de mise en forme des courbes sont disponibles. La fonction *plot* accepte un troisième argument facultatif qui vous permet de spécifier une mise en forme pour les courbes.
Le tableau de la Figure 216 présente les codes que l'on peut saisir pour obtenir la mise en forme des courbes. Ces codes peuvent être combinés dans un ordre quelconque.

Couleur		Marqueur		Style de ligne	
b	Bleu	**.**	Point	**-**	Continu
g	Vert	**o**	Cercle	**--**	Trait
r	Rouge	**x**	Croix diagonale	**:**	Pointillés
c	Cyan	**+**	Croix verticale	**-.**	Trait-point
m	Magenta	*****	Etoile		
y	Jaune	**s**	Carré		**Remarque** : Si un style de marqueurs est spécifié sans aucun style de ligne aucune ligne ne sera tracée.
k	Noir	**d**	Losange		
w	blanc	**v**	Triangle vers le bas		
		^	Triangle vers le haut		
		<	Triangle à gauche		
		>	Triangle à droite		
		p	Pentagramme		
		h	hexagramme		

Figure 216 : codes de mise en forme des courbes

Vous pouvez faire quelques essais :

```
>>hold off
>>plot(x,y1,'r*:')
```

r pour rouge, * pour les marqueurs et **:** pour le style de ligne pointillés

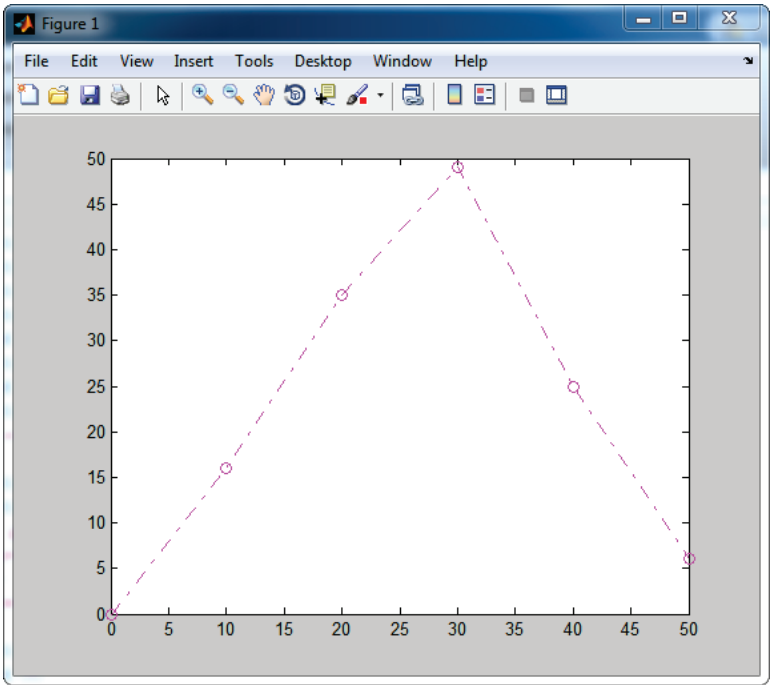


Figure 217 : mise en forme d'une courbe : modification de la couleur et du style de ligne

```
>>plot(x,y1,'mp')
```

Aucun style de ligne n'est spécifié, ce sont uniquement les marqueurs qui apparaissent.

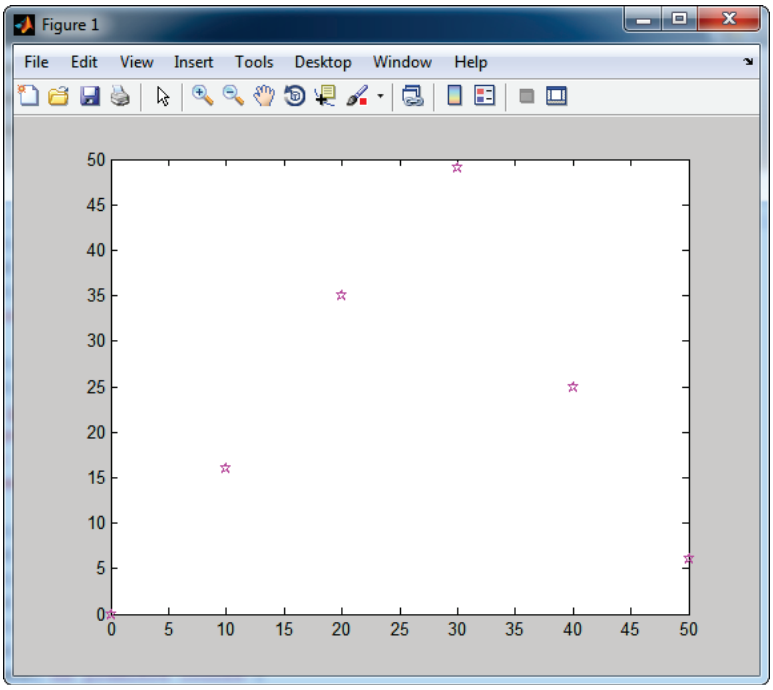


Figure 218 : mise en forme d'une courbe : ajout de marqueurs

Il est également possible de modifier l'épaisseur du trait avec l'argument *LineWidth*

```
>> plot(x,y1,'mo-.','LineWidth',3)
```

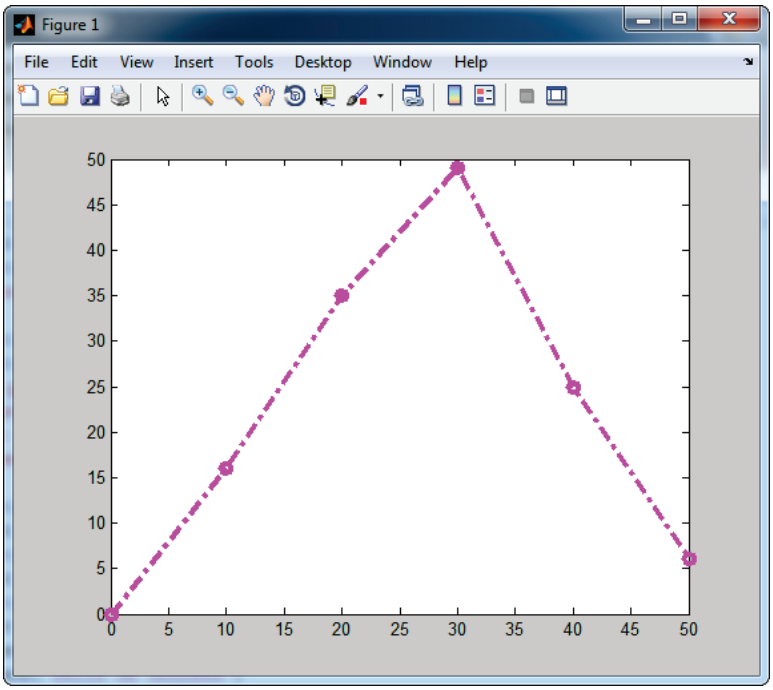


Figure 219 : mise en forme d'une courbe : choix de l'épaisseur de trait

F. Annotation des graphiques

Il est également possible d'annoter le graphique:

```
>> title('ma première courbe')          % affiche un titre au graphique
>> xlabel('vecteur x')                  % étiquette de l'axe des abscisses
>> ylabel('vecteur y1')                 % étiquette de l'axe des ordonnées
>> legend('y1 en fonction de x')        % ajout d'une légende
```

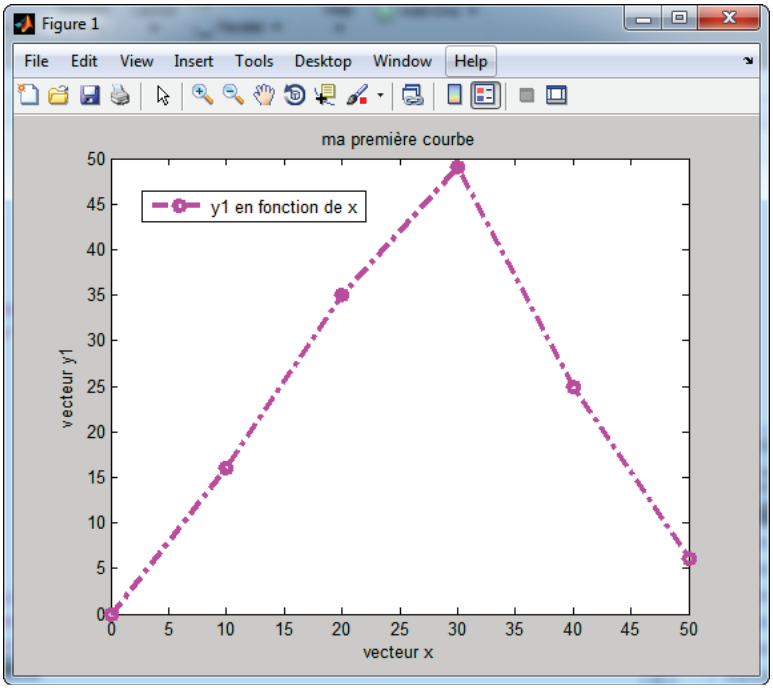


Figure 220 : mise en forme d'une courbe : légende et annotation des axes

La commande *axis* permet de définir manuellement les échelles des axes.

```
axis([0 100 -10 60])
```

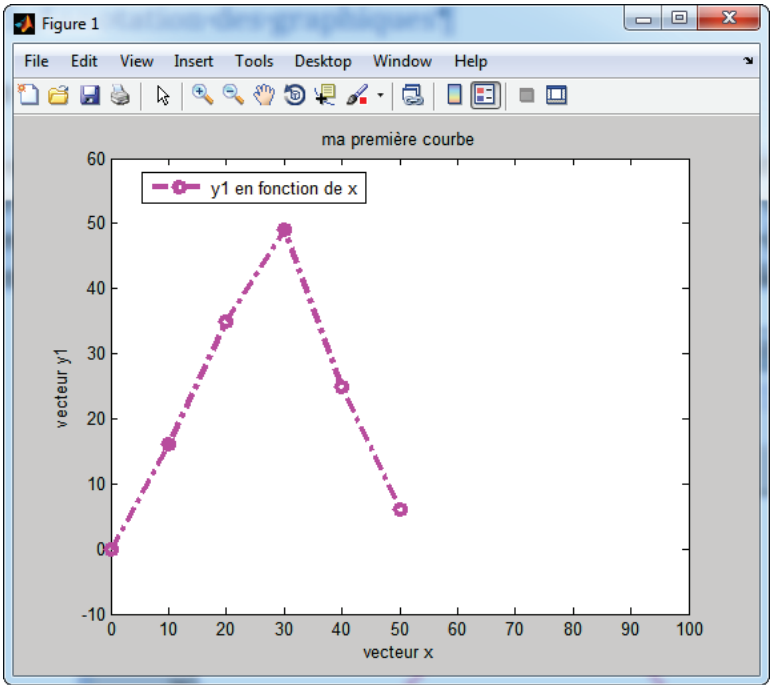
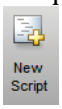


Figure 221 : mise en forme d'une courbe : modification des échelles des axes

G. Créer un script élémentaire

Un script est un programme écrit en langage **MATLAB**. On comprendra aisément qu'il est très utile de regrouper une suite de lignes de commandes dans un fichier appelé script. L'intérêt est de pouvoir le sauvegarder et exécuter plusieurs fois la même séquence.

Pour créer un script, cliquer sur **New script** 

dans la barre de commande.

Une fenêtre **Editor** apparaît dans laquelle il est possible de saisir des lignes de commandes.

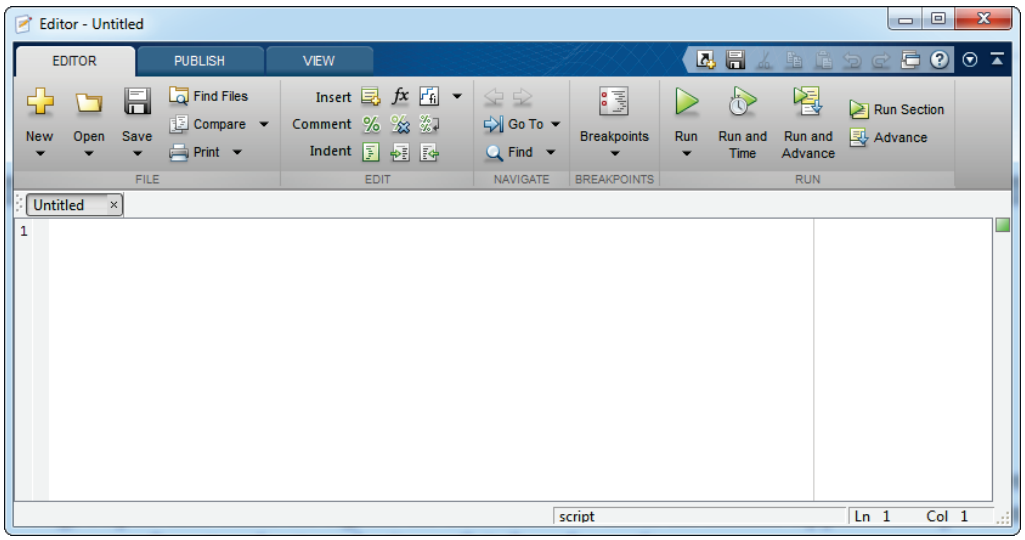


Figure 222 : fenêtre Editor d'édition des scripts

Saisir les lignes de commande et sauvegarder le fichier sous le nom ***plot_sinus_0.m*** (l'extension **.m** est l'extension des fichiers scripts de MATLAB, l'extension .m sera mise par défaut)).

Le signe **%** signal un commentaire dans un script.

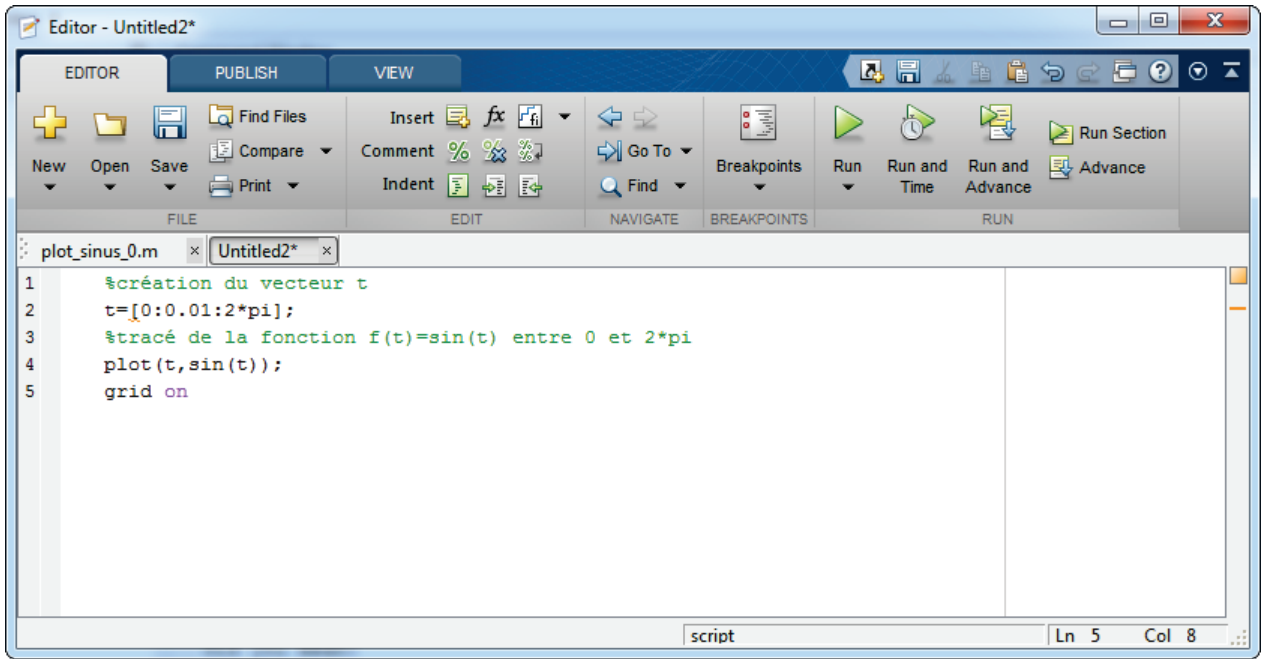


Figure 223 : premier script avec MATLAB

Exécuter le script en cliquant sur 

dans la barre de commande.

Si la fenêtre de la Figure 224 apparaît, l'emplacement du fichier n'appartient pas au « **path** » de MATLAB et MATLAB ne peut pas exécuter le script. Le logiciel vous propose :

- D'ajouter le dossier en question au « **path** » : **Add to Path** (recommandé)
- De déplacer ce fichier vers le dossier courant de travail de **MATLAB** : **Change FOLDER**

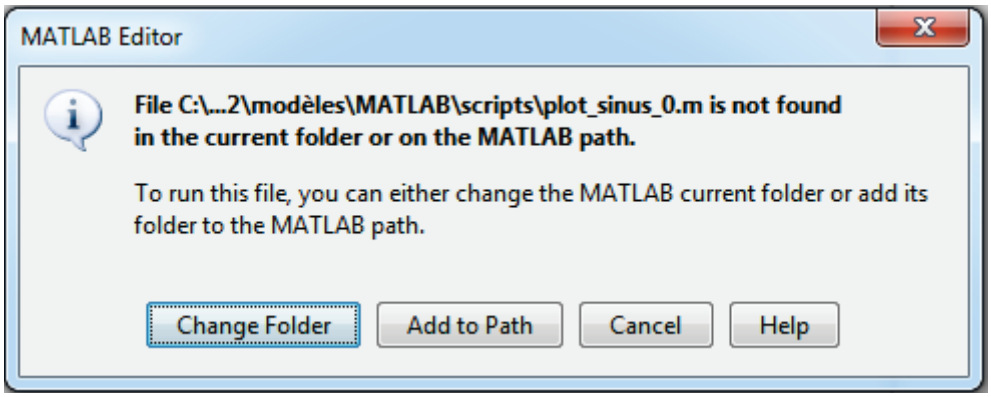
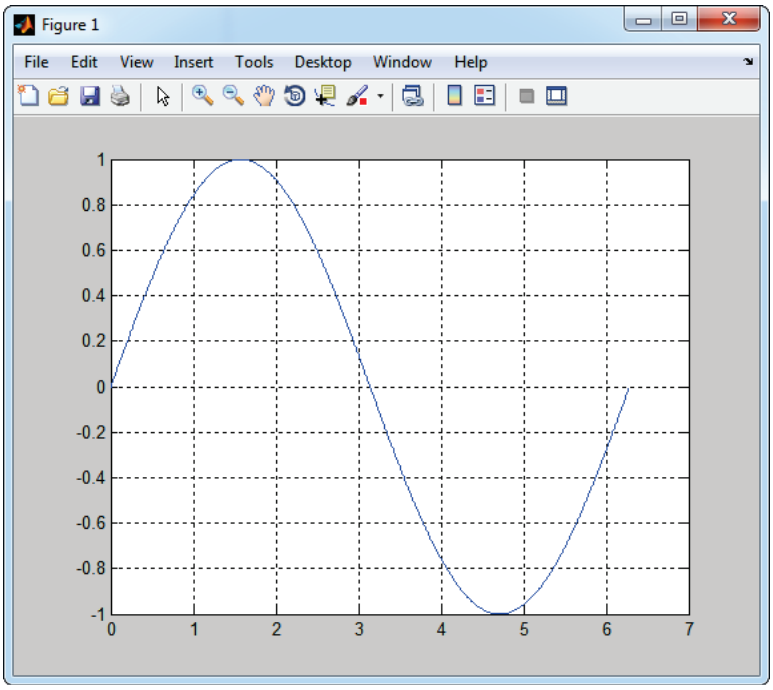


Figure 224 : fenêtre d'ajout automatique de dossier au « path » de MATLAB

On obtient la représentation de la fonction sinus en fonction du temps.



Il est possible d'utiliser des commandes de programmation classique comme une boucle **for**. Modifier votre script comme indiqué sur la Figure 225.

```
%création du vecteur t
t=[0:0.01:2*pi];
% la fonction hold all est identique à la fontion hold on mais permet
% de tracer chaque courbes avec une couleur différentes
hold all
% pour a prenant les valeurs comprises entre 1 et 4 avec un pas de 1

for a=[1:4]
%tracé de la fonction f(t)=sin(t) entre 0 et 2*pi
plot(t,sin(a*t),'LineWidth',2);
end
grid on
title('Fonction Sinus')
%MATLAB possède un interpréteur TeX de base, il est possible d'y faire
%figurer des caractères spéciaux ou des équations
xlabel('angle \theta en radians')
ylabel('f(\theta)=sin(\theta)')
axis([-1 7 -1.5 1.5])
```

Figure 225 : Tracé de plusieurs Sinus sur le même graphique

Exécuter le script et visualiser le résultat.
Si une erreur de syntaxe est détectée lors de l'exécution, le détail apparaît en rouge dans la fenêtre de commande.

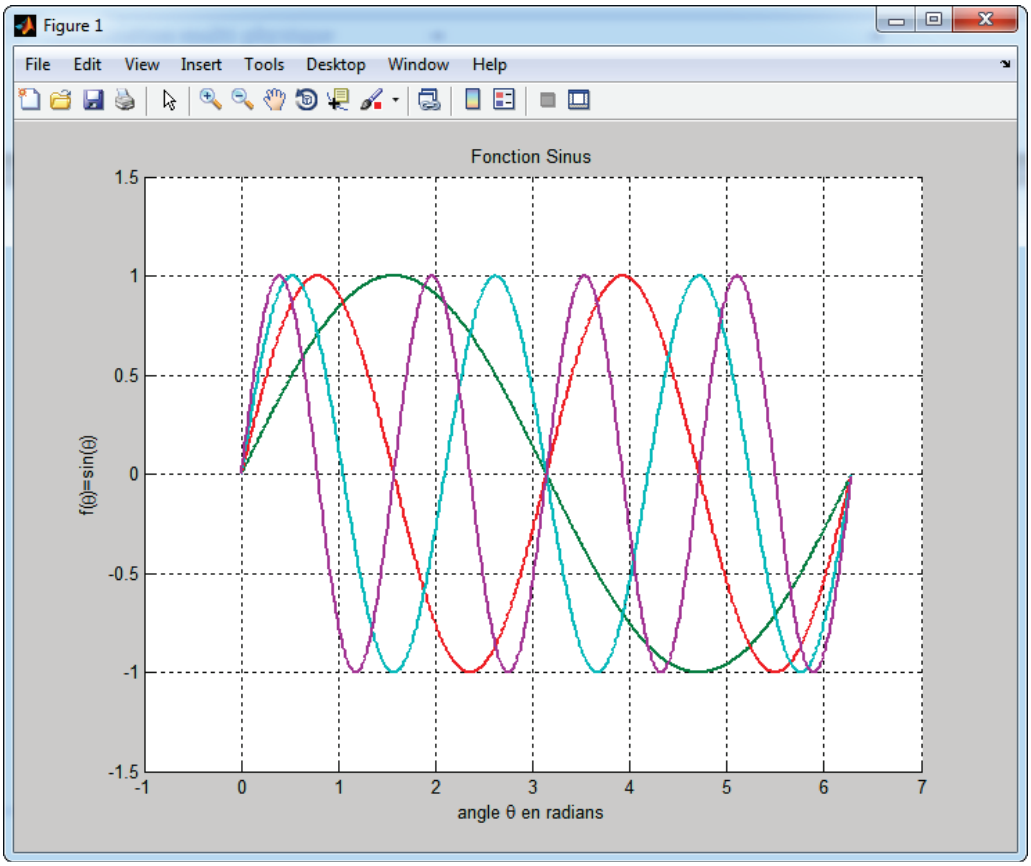


Figure 226 : résultats du script de tracé de plusieurs sinus

H. Les opérateurs de comparaison de MATLAB

Afin de réaliser des tests dans les scripts, **MATLAB** possède des opérateurs de comparaison et des opérateurs logiques.

Commandes utiles	
Opérateurs	Syntaxe MATLAB
Egal à	==
Différent de	~=
Supérieur à	>
Supérieur ou égal à	>=
Inférieur à	<
Inférieur ou égal à	<=
Ou(or)	|
Et(and)	&

Figure 227 : les opérateurs logiques et de comparaison de MATLAB

I. Les structure de boucles usuelles

Afin de réaliser des boucles dans les scripts, les trois solutions les plus utilisées sont :

- if – elseif – else
- for
- while

1. Syntaxe de la boucle if – elseif – else

```
if (test logique 1)
        action 1
elseif ( test logique 2)
        action 2
else
        action 3
```

Pour ce type de boucle, une seule action parmi *action1, action 2 et action 3* sera exécutée. Les tests sont effectués dans l'ordre et uniquement si cela est nécessaire. Ainsi si (*test logique 1*) est vrai, *action 1* sera exécutée et (test logique 2) ne sera pas évalué. Il peut y avoir un nombre quelconque d'instructions *elseif* ou aucune. Il peut y avoir une seule instruction *else* qui doit représenter la dernière condition.

2. Syntaxe de la boucle for

```
for variable=valeur :initiale :incrément :valeur_finale
        action
end
```

La boucle *for* exécute l'action un nombre défini de fois en faisant varier la valeur d'une variable.

3. Syntaxe de la boucle while

```
while condition
        action
end
```

L'action est exécutée tant que la condition évaluée est vraie. La sortie de la boucle s'opère lorsque la condition évaluée devient fausse.

II. Exemple d'exploitations

A. Interpolation d'une série de données

Lors de l'analyse de données expérimentales, il est souvent nécessaire d'interpoler une série de données. MATLAB possède des fonctionnalités qui permettent de réaliser des interpolations très simplement.
Considérons une série de points expérimentaux représentés par les deux vecteurs X=[0 10 20 30 40] et Y=[2 5 12 25 46].

L'objectif est de placer ces points sur un graphique et de tracer la courbe d'interpolation.

Dans la fenêtre de commande, **créer** les deux vecteurs X et Y et visualiser l'allure de la répartition en traçant les points expérimentaux obtenus

```
>>X=[0 10 20 30 40];
>>Y=[2 5 12 25 46];
>>plot(X,Y,'r.','MarkerSize',25);
```

la spécification du marqueur sans spécifier le style de ligne affiche les marqueurs uniquement

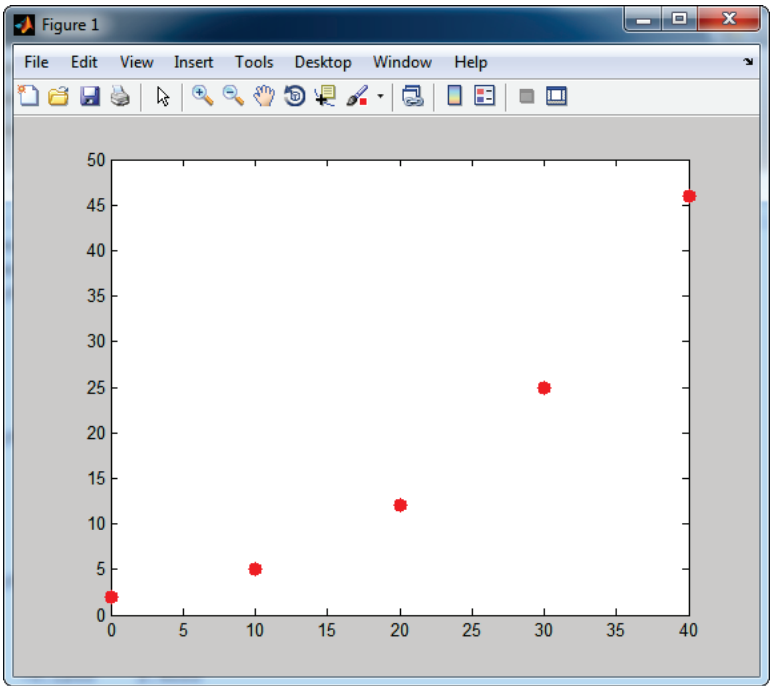


Figure 228 : repartition d'une série de points

L'allure de la répartition nous permet de permet de chercher une interpolation avec un polynôme de degré 2. La fonction *polyfit* donne automatique les coefficients du polynôme d'interpolation.

```
>> polyfit(X,Y,2)
ans =
   0.0300   -0.1200    2.4000
```

Le polynôme d'interpolation de degré 2 pour expression $p(x) = 0.03\,x^2 - 0.12\,x + 2.4$.

Tester le script de la Figure 229 pour comprendre l'intérêt d'utiliser un script pour automatiser ce type de tâche. Ce script permet de placer les différents points (ronds rouges) et de superposer la courbe d'interpolation.

```
1      % création des vecteurs X et Y
2 -    X=[0 10 20 30 40];
3 -    Y=[2 5 12 25 46];
4      %création du vecteur A qui contient les coefficients du polynôme
5      %d'interpolation de degré 2 qui interpole la série de donnée
6 -    A=polyfit(X,Y,2);
7      %Création d'un vecteur t contenant 100 composante prise entre X(0) et
8      %X(end), X(end) représente la dernière composante du vecteur X.
9 -    t=linspace(X(1),X(end),100);
10     %Création du vecteur U qui contient 100 éléments correspondants aux
11     %valeurs du polynôme d'interpolation pour les 100 éléments du vecteur t
12 -    U= polyval(A,t);
13     %tracé du polynôme d'interpolation
14 -    plot(t,U,'g','LineWidth',3);
15 -    hold on;
16     %tracé des points Y en fonction de X, la commande MarkerSize permet de
17     %spécifier la taille des marqueurs
18 -    plot(X,Y,'r.','MarkerSize', 25);
19 -    grid on;
```

Figure 229 : script d'interpolation d'une série de données

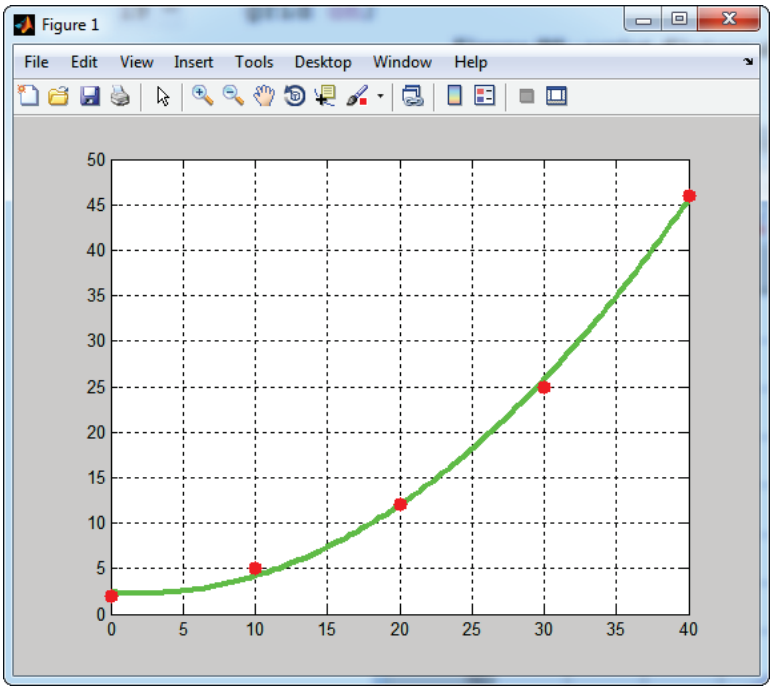


Figure 230 : graphique de l'interpolation d'une série de données

B. Le calcul symbolique avec MATLAB

En phase de modélisation, l'étudiant et l'ingénieur sont souvent amenés à résoudre des équations algébriques ou différentielles. Dans cette partie, nous verrons comment mener une démarche de calcul avec des variables symboliques, résoudre des équations, travailler avec la transformée de Laplace…

1. Résolution d'une équation algébrique

Considérons l'équation suivante : $x^2 + 2x - 3 = 0$
Nous souhaitons trouver les valeurs de x qui vérifient cette équation en créant un script.
Nous utiliserons la commande *syms* qui permet à MATLAB de traiter une variable comme une variable symbolique.

Saisir le script suivant ou ouvrir le script « **resolution_equa_algebrique.m** » :

```
1     %défintion de x comme une variable symbolique
2 -   syms x
3     %% %résolution d'une equation algébrique
4
5     %définition de l'équation à résoudre
6 -   equation= x^2+2*x-3==0;
7     %Résolution de l'équation
8 -   solution=solve(equation,x);
9
10 -  disp('La première racine est:'),disp (solution(1))
11 -  disp('La seconde racine est:'),disp (solution(2))
```

Figure 231 : script de résolution d'une équation du second degré

L'exécution du script permet d'obtenir le résultat suivant :

```
>>resolution_equa_algebrique
La première racine est:
-3

La seconde racine est:
1
```

MATLAB renvoie la valeur des deux racines de cette équation.

Dans le cas de racine complexe, la méthode de résolution est la même. Considérons l'équation suivante :
Considérons l'équation suivante : $x^2 + 2x + 5 = 0$

Modifier votre script ou ouvrir le script « « **resolution_equa_algebrique_complexe.m** » :

```
%définition de x comme une variable symbolique
syms x
%% %résolution d'une equation algébrique (racine complexes)

%définition de l'équation à résoudre
equation= x^2+2*x+5==0;
%Résolution de l'équation
solution=solve(equation,x);

disp('La première racine est:'),disp (solution(1))
disp('La seconde racine est:'),disp (solution(2))
```

Figure 232 : résolution d'une équation du second degré avec racines complexes

L'exécution du script permet d'obtenir le résultat suivant :

```
>> resolution_equa_algebrique_complexe
La première racine est:
- 1 - 2i

La seconde racine est:
- 1 + 2i
```

MATLAB renvoie la valeur des deux racines de cette équation sous la forme d'un nombre complexe.

2. Développer ou factoriser une expression

Il est souvent utile de changer la forme d'une expression en la factorisant ou en la développant. Pour utiliser les fonctions *expand* et *factor*, la variable utilisée dont être définie comme une variable symbolique.

Saisir le script suivant ou ouvrir le script « **developper_et_factoriser.m** » :

```
%définition de x comme une variable symbolique
syms x
%% développer une expression
expr1=(x-8)*(x+2)^2
expand(expr1)

%% factoriser une expression
expr2=x^3 - 4*x^2 - 28*x - 32
factor(expr2,x)
```

Figure 233 : développer et factoriser une expression

L'exécution du script permet d'obtenir le résultat suivant :

```
>> developper_et_factoriser

expr1 =
 (x + 2)^2*(x - 8)

 ans =

x^3 - 4*x^2 - 28*x – 32                    (développement de expr1)

 expr2 =

x^3 - 4*x^2 - 28*x - 32

 ans =

[ x - 8, x + 2, x + 2]                     (factorisation de expr2)
```

Ces commandes peuvent être utiles pour la manipulation des fonctions de transfert.

3. Dériver une fonction

Considérons la fonction : $f(t) = \sin(t) + 3\,t^3$.
Nous souhaitons calculer les dérivées successives de cette fonction par rapport à la variable t.

Saisir le script suivant ou ouvrir le script « **deriver_fonction.m** » :

```
1     %% dériver une fonction
2     %définition de t comme une variable symbolique
3 -   syms t
4     %définition de la fonction à dériver
5 -   f=sin(t)+3*t^3;
6
7     %calcul de la dérivée première
8 -   df=diff(f,t)
9
10    %calcul de la dérivée cinquième
11 -  d5f=diff(f,t,5)
```

Figure 234 : script permettant de dériver une fonction

L'exécution du script permet d'obtenir le résultat suivant :

```
>> >> deriver_fonction

df =

cos(t) + 9*t^2                    (calcul de la dérivée première de la fonction)

 d5f =

cos(t)                            (calcul de la dérivée cinquième de la fonction)
```

4. Intégrer une fonction

Considérons la fonction : $f(t) = \sin(t) + 3\,t^3$.

Nous souhaitons calculer la primitive de $f(t)$ et l'intégrale suivante : $I = \int_{\frac{-\pi}{3}}^{\pi} f(t)\,dt$

Saisir le script suivant ou ouvrir le script « **integrer_fonction.m** » :

```
%% Intégrer une fonction
%définition de t comme une variable symbolique
syms t
%% Intégration d'une fonction

%calcul de la primitive
int(f,t)

%calcul d'une intégrale définie
int(f,t,-pi/3,pi)
```

Figure 235 : script permettant d'intégrer une fonction et de calculer une intégrale définie

L'exécution du script permet d'obtenir le résultat suivant :

```
>> integrer_fonction

ans =

 (3*t^4)/4 - cos(t)                    calcul de la primitive de f(t)

 ans =

(20*pi^4)/27 + 3/2                     calcul de I = int_{-pi/3}^{pi} f(t) dt
```

5. Utiliser la transformée de Laplace

La commande *laplace* permet d'obtenir très facilement la transformée de Laplace d'une fonction.

Saisir le script suivant ou ouvrir le script « **transformee_laplace.m** » :

```
%% Transformée de Laplace d'une expression
syms t;
syms w;
syms a;
laplace(sin(w*t))
laplace(cos(w*t))
laplace(exp(-a*t)*cos(w*t))
laplace(exp(-a*t)*sin(w*t))
```

Figure 236 : script permettant de calculer la transformée de Laplace de fonctions

L'exécution du script permet d'obtenir le résultat suivant :

```
>> transformee_laplace

ans =
 w/(s^2 + w^2)                    calcul de la transformée de Laplace de sin(ω t)

ans =
 s/(s^2 + w^2)                    calcul de la transformée de Laplace de cos(ω t)

 ans =

(a + s)/((a + s)^2 + w^2)         calcul de la transformée de Laplace de e^(-a t) cos(ω t)

 ans =

w/((a + s)^2 + w^2)               calcul de la transformée de Laplace de e^(-a t) sin(ω t)
```

6. Utiliser la transformée inverse de Laplace

La commande *ilaplace* permet d'obtenir très facilement la transformée inverse de Laplace d'une fonction.

Saisir le script suivant ou ouvrir le script « **transformee_inverse_laplace.**m » :

```
1     %% Transformée  inverse de Laplace d'une expression
2 -   syms t;
3 -   syms w;
4 -   syms a;
5
6     %% Transformée inverse de Laplace d'une expression
7 -   syms s
8 -   ilaplace(s/(s^2 + w^2))
9 -   ilaplace(3/(2+s)-5/(3+s)^2)
```

Figure 237 : script permettant d'obtenir la transformée inverse de Laplace d'une fonction

L'exécution du script permet d'obtenir le résultat suivant :

```
>> transformee_inverse_laplace

ans =
cos(t*w)                          calcul de la transformée inverse de Laplace de s/(s^2 + ω^2)
ans =
3*exp(-2*t) - 5*t*exp(-3*t)       calcul de la transformée inverse de Laplace de 3/(2+s) - 5/(3s^2)
```

7. Décomposition en éléments simples

La commande *residue* permet d'obtenir la décomposition en éléments simples d'une fraction rationnelle.

Considérons la focntion $H(s)=\frac{4s+5}{s^2+6s+8}$ à décomposer en éléments simples.

Saisir le script suivant ou ouvrir le script « **decomposition_elements_simples.m** » :

```
%% Décomposition en éléments simples
%définition de la fraction rationnelle a/b
%numérateur
a=[4 5];
%dénominateur%
b=[1 6 8];

%décomposition en éléments simples
[r p k]=residue(a,b)
```

Figure 238 : script permettant de décomposer en éléments simples une fraction rationnelle

L'exécution du script permet d'obtenir le résultat suivant :

```
decomposition_elements_simples

r =

   5.5000
  -1.5000

p =

   -4
   -2

k =

[]
```

Le variable **r** contient les coefficients du numérateur de chaque élément simple.
Le variable **p** contient les valeurs qui annulent le dénominateur de chacun des éléments simples.
La variable **k** contient les coefficients du polynôme qui s'ajouterait éventuellement aux éléments simples pour assurer l'égalité avec la fraction rationnelle initiale.

Le résultat est donc pour l'exemple traité :

$$H(s)=\frac{4s+5}{s^2+6s+8}=\frac{5.5}{p+4}-\frac{1.5}{p+2}$$

1. Résolution d'une équation différentielle d'ordre 1

Il est très facile de résoudre des équations différentielles linéaires.
Prenons par exemple la forme classique de l'équation différentielle linéaire du premier ordre :

$$T\frac{df(t)}{dt}+f(t)=K$$

MATLAB peut résoudre cette équation en utilisant le calcul symbolique.

Définition des variables et de la fonction symbolique :

```
>> syms K;syms T;syms t;syms f(t);
```

Définition des dérivées successives de f(t)

```
>> D1f=diff(f,1); D2f=diff(f,2);
```

Définition de l'équation différentielle du premier ordre

```
>> equ1=T*D1f+f(t)==K
```

Résolution de l'équation sans spécification des conditions initiales :

```
>> dsolve(equ1)
ans =

K - C1*exp(-t/T)
```

La solution $f(t)=K-C_1\,e^{-\frac{t}{T}}$ est donnée en fonction d'une constante C_1 que MATLAB ne peut pas déterminer car la condition initiale n'est pas définie.

Il est possible de résoudre cette équation en spécifiant la condition initiale :

```
>> dsolve(equ1,f(0)==0)
ans =

K - K*exp(-t/T)
```

La solution est donnée en tenant compte de la condition initiale : $f(t)=K\left(1-e^{-\frac{t}{T}}\right)$

Vous pouvez ouvrir le script « **resolution_equation_differentielle_ordre_1.m** » qui regroupe l'ensemble des commandes utilisées pour résoudre une équation différentielle d'ordre 1.

1. Résolution d'une équation différentielle d'ordre 2

Prenons maintenant l'exemple de deux équations différentielles linéaires du second ordre :

$$5\frac{d^2f(t)}{dt^2}+3\frac{df(t)}{dt}+f(t)=2 \quad \text{avec} \quad f(0)=0 \text{ et } \frac{df(0)}{dt}=0 \quad \text{(équation 1)}$$

$$5\frac{d^2f(t)}{dt^2}+\frac{df(t)}{dt}+f(t)=2 \quad \text{avec} \quad f(0)=0 \text{ et } \frac{df(0)}{dt}=0 \quad \text{(équation 2)}$$

MATLAB peut résoudre ces équations et tracer les fonctions solutions.

Saisir le script suivant ou ouvrir le script « **resolution_equation_differentielle_ordre_2.m** » :

```
%définition des variables et de la fonction symbolique
syms t;
syms f(t);

%% définition des dérivées successive de f(t)
D1f=diff(f,1);
D2f=diff(f,2);

%% Résolution d'une équation du second ordre:exemple 1
equ1=5*D2f+3*D1f+f==2
sol1=dsolve(equ1,f(0)==0,D1f(0)==0)

%représentation graphique de la solution
figure;
ezplot(sol1,[0,40,0,3])
legend('solution de l''equation 1')
grid on

%Résolution d'une équation du second ordre:exemple 2
equ2=5*D2f+1*D1f+f==2
sol2=dsolve(equ2,f(0)==0,D1f(0)==0)

%représentation graphique de la solution
figure;
ezplot(sol2,[0,70,0,4])
legend('solution de l''equation 2')

grid on;
```

Figure 239 : script permettant la résolution d'une équation différentielle du second ordre

Exécuter le script.

```
>> resolution_equation_differentielle_ordre_2

equ1(t) =

f(t) + 3*diff(f(t), t) + 5*diff(f(t), t, t) == 2

 sol1 =

2 - (6*11^(1/2)*exp(-(3*t)/10)*sin((11^(1/2)*t)/10))/11 - 2*exp(-(3*t)/10)*cos((11^(1/2)*t)/10)

equ2(t) =

f(t) + diff(f(t), t) + 5*diff(f(t), t, t) == 2

sol2 =

2 - (2*19^(1/2)*exp(-t/10)*sin((19^(1/2)*t)/10))/19 - 2*exp(-t/10)*cos((19^(1/2)*t)/10)
```

Les solutions sol1 et sol 2 sont exprimées sous forme symbolique et sont tracées à l'aide de la commande *ezplot* (Figure 240 et Figure 241)

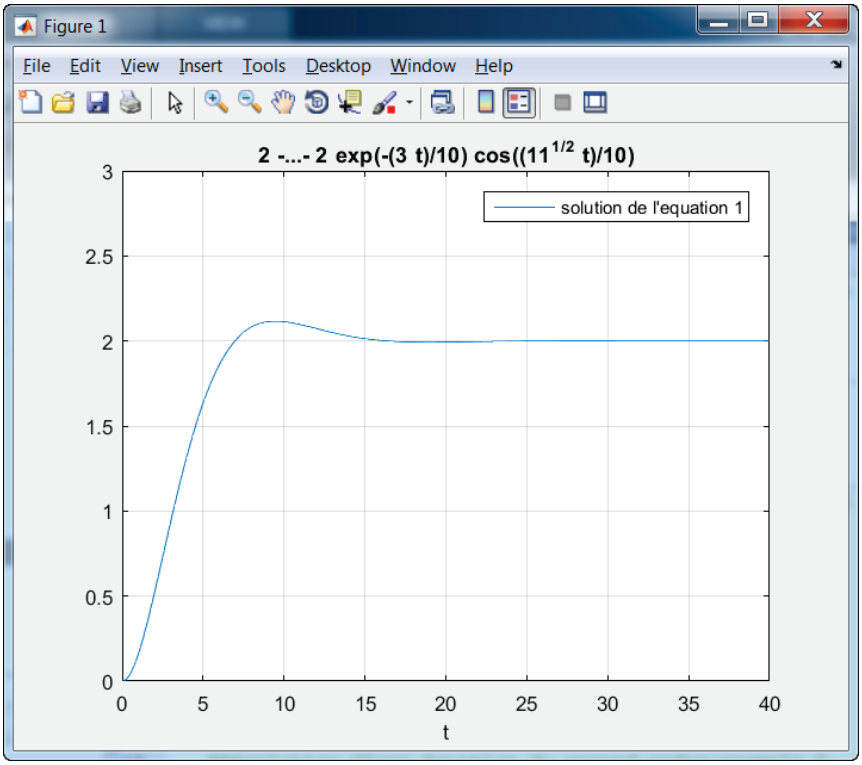


Figure 240 : représentation de la solution 1 de l'équation différentielle d'ordre 2

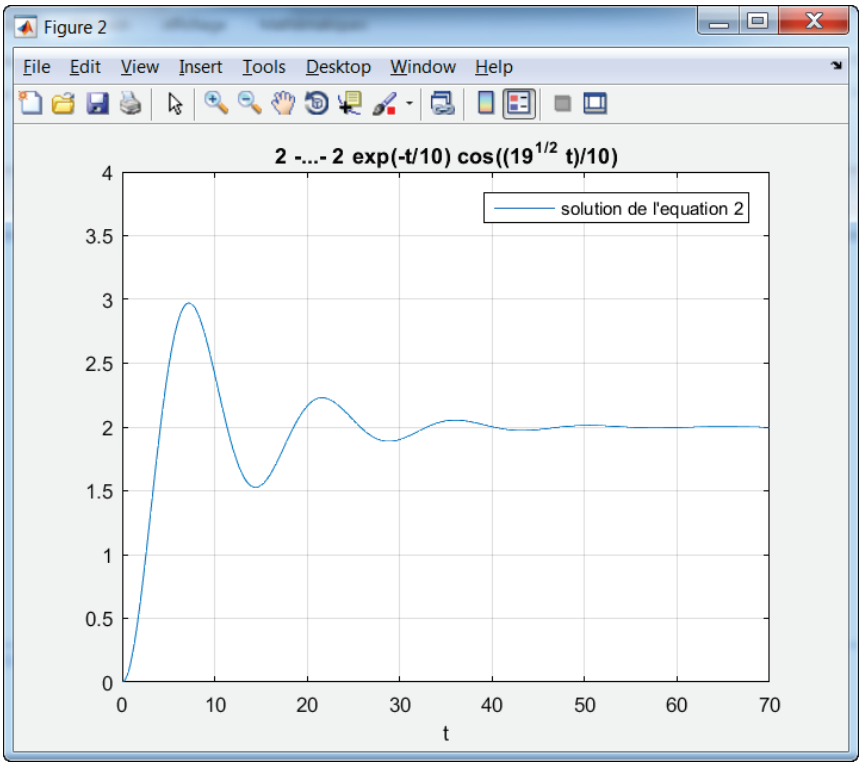


Figure 241 : représentation de la solution 2 de l'équation différentielle d'ordre 2

C. Manipulation des fonctions de transfert

MATLAB possède de nombreuses commandes qui permettent de manipuler les fonctions de transferts, tracer les réponses temporelles et fréquentielles des systèmes.

1. Création d'une fonction de transfert

Considérons deux fonctions de transfert :

$$\begin{cases} H(p) = \dfrac{4s+1}{2s^2+3s+6} \\ G(p) = \dfrac{100(s+1)(s+2))}{(s+0.1)(s+4)(s+10)} \end{cases}$$

Pour créer une fonction de transfert, plusieurs méthodes sont possibles en fonction de la forme selon laquelle elle est exprimée.

Pour $H(p)$, l'expression est développée, le plus simple est de saisir directement les coefficients du numérateur et du dénominateur en utilisant la commande *tf*

```
>> H=tf([4 1],[2 3 6])
H =
    4 s + 1
 ---------------
 2 s^2 + 3 s + 6
Continuous-time transfer function.
```

Pour $G(p)$, l'expression est factorisée, le plus simple est de la saisir directement sous cette forme pour éviter d'avoir à faire un développement

```
>> s=tf('s')                    indique à MATLAB que la variable s sera considérée comme la variable de Laplace
s =
 s
Continuous-time transfer function.

>> G=100*((s+1)*(s+2))/((s+0.1)*(s+4)*(s+10))
G =
       100 s^2 + 300 s + 200
 ---------------------------------
 s^3 + 14.1 s^2 + 41.4 s + 4
 Continuous-time transfer function.
```

La commande commande *zpk* permet de saisir une fonction de transfert en indiquant les pôles et les zéros.

Il aurait été également très simple de définir $G(p)$ en utilisant cette commande

```
>>G=zpk([-1,-2],[-0.1,-4,-10],100)
G =
   100 (s+1) (s+2)
 --------------------
 (s+0.1) (s+4) (s+10)
Continuous-time zero/pole/gain model.
```

2. Opérations sur les fonctions de transfert

Pour obtenir les pôles d'une fonction de transfert on utilise la commande *pole.*

```
>> pole(G)
ans =
 -10.0000
  -4.0000
  -0.1000
```

Pour obtenir les zéros d'une fonction de transfert on utilise la commande *zero.*

```
>> zero(G)
ans =
   -2
   -1
```

Pour calculer la fonction de transfert équivalente à deux fonctions de transfert en série on utilise la commande *series.*

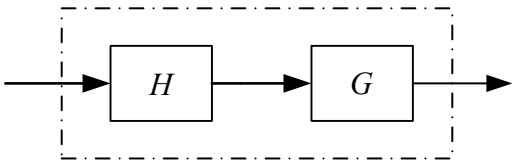


Figure 242 : fonctions de transfert en série

```
>> series(H,G)
ans =
       200 (s+1) (s+2) (s+0.25)
 -------------------------------------
 (s+0.1) (s+4) (s+10) (s^2 + 1.5s + 3)
Continuous-time zero/pole/gain model.
```

Cette opération est équivalente à la commande

```
>>H*G
```

Pour calculer la fonction de transfert équivalente à deux fonctions de transfert en parallèle, on utilise la commande *parallel*

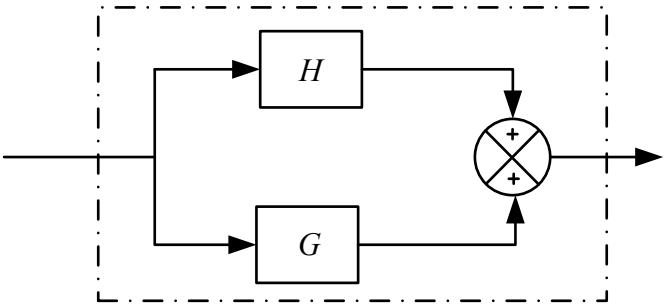


Figure 243 : fonction de transfert en parallèle

```
>> parallel(H,G)
ans =
  102 (s+1.515) (s+1.334) (s^2 + 1.844s + 2.92)
 ---------------------------------------------
    (s+4) (s+10) (s+0.1) (s^2 + 1.5s + 3)
Continuous-time zero/pole/gain model.
```

Pour calculer la fonction de transfert en boucle fermée, on utilise la commande *feedback*.

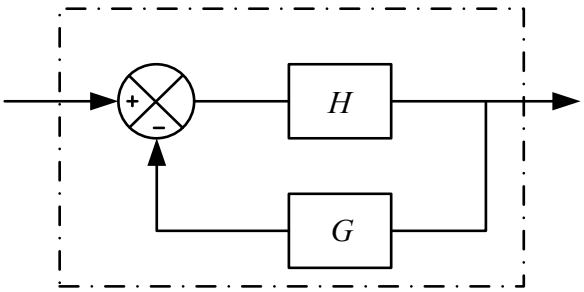


Figure 244 : fonction de transfert en boucle fermée

```
>> feedback(H,G)
ans =
          2 (s+0.25) (s+0.1) (s+4) (s+10)
 ------------------------------------------------------
 (s+0.2104) (s^2 + 3.028s + 2.392) (s^2 + 12.36s + 222.5)
Continuous-time zero/pole/gain model.
```

Pour inverser une fonction de transfert on peut utiliser la commande *inv.*

```
>> inv(H)
ans =
  2 s^2 + 3 s + 6
 ---------------
    4 s + 1
Continuous-time transfer function.
```

Exemple de fonction de transfert quelconque.

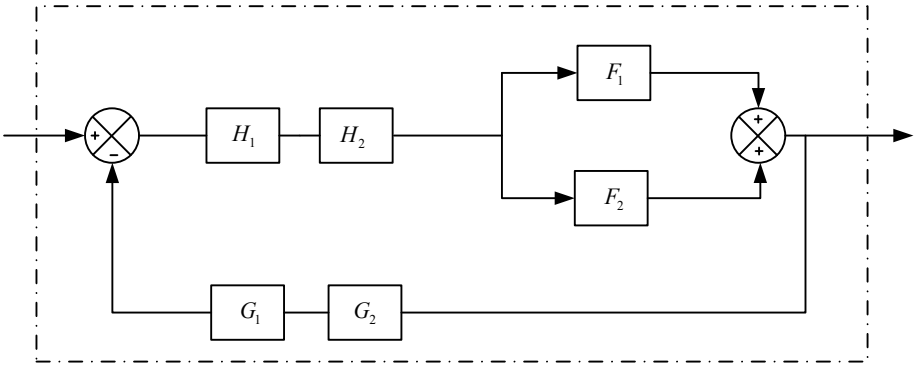


Figure 245 : fonction de transfert d'un système quelconque

On peut obtenir directement la fonction de transfert global d'un tel système en tapant la commande

```
>>feedback((H1*H2)*parallel(F1,F2),G1*G2)
```

Il faudrait préalablement saisir les valeurs de chacune de ces fonctions.

Il est également très simple de construire la fonction de transfert d'un correcteur PID en utilisant la fonction *pid.*

```
>>pid(50,0.1,70)                    Les trois arguments représentent les coefficients Kp, Ki et Kd du correcteur : pid(Kp,Ki,Kd)
ans =
                1
 Kp + Ki * --- + Kd * s
                s
 with Kp = 50, Ki = 0.1, Kd = 70
Continuous-time PID controller in parallel form.
```

3. Tracer les réponses temporelles d'un système

Pour obtenir la réponse indicielle d'un système, on peut utiliser la commande *step*

```
>>step(H) ; grid on
```

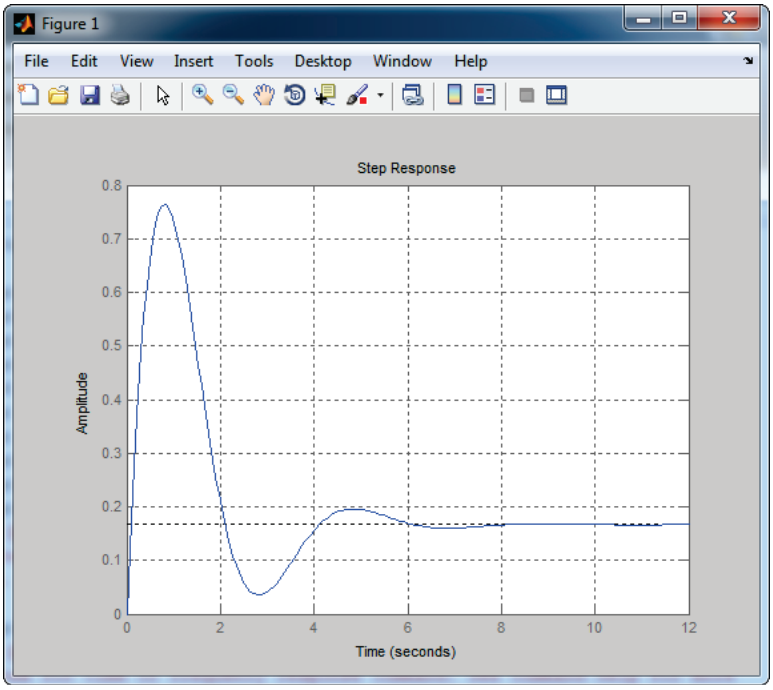


Figure 246 : réponse indicielle d'un système

Pour obtenir la réponse impulsionnelle d'un système, on peut utiliser la commande *impulse*

```
>>impulse(H) ; grid on
```

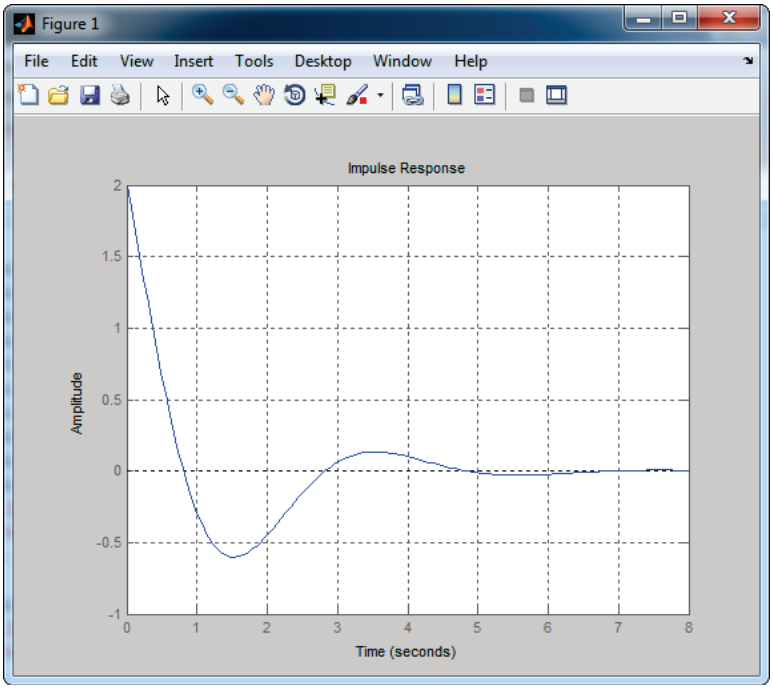


Figure 247 : réponse impulsionnelle d'un système

4. Tracer les réponses fréquentielles d'un système

Pour obtenir le diagramme de Bode d'un système, on peut utiliser la commande *bode*

```
>>bode(H) ; grid on
```

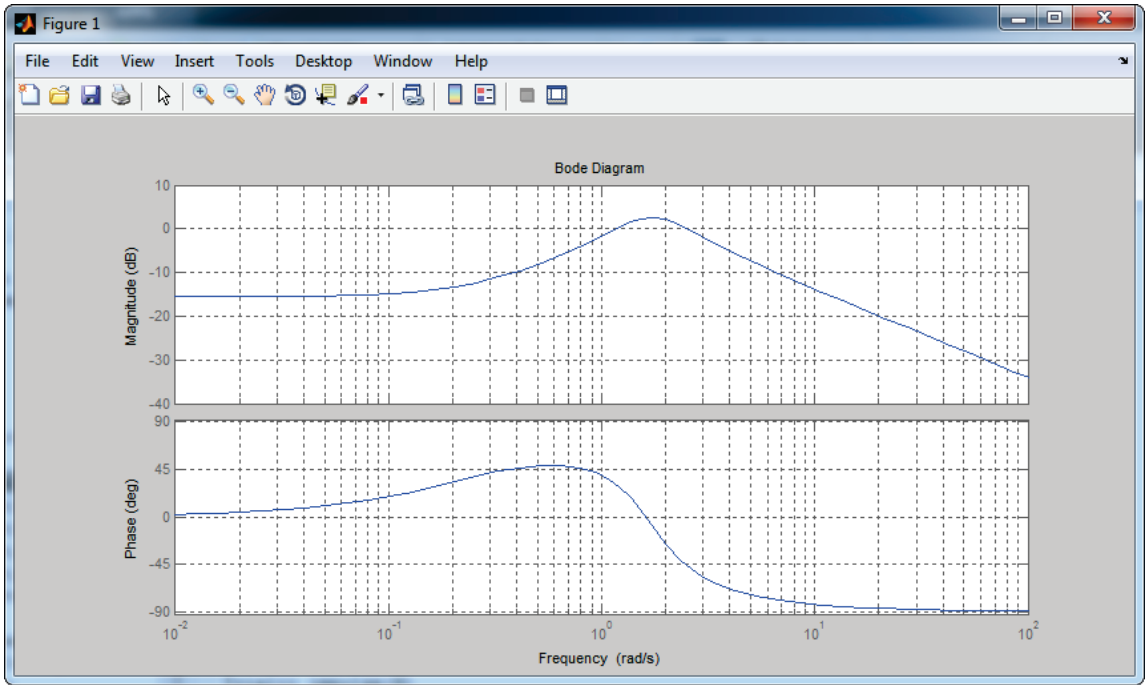


Figure 248 : diagramme de Bode d'un système

Pour obtenir le diagramme de Black-Nichols d'un système, on peut utiliser la commande *nichols*

```
>>nichols(H) ; grid on
```

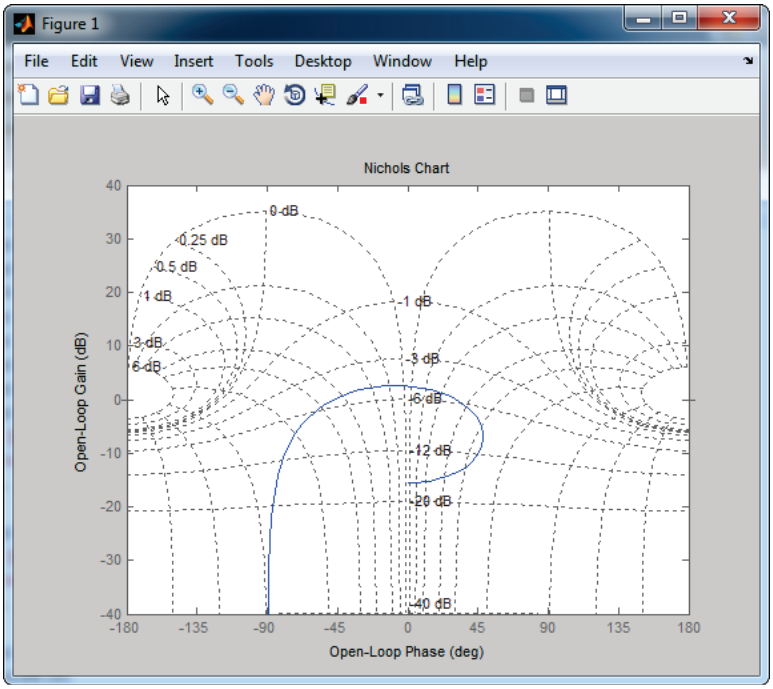


Figure 249 : diagramme de Black-Nichols d'un système

Pour obtenir le diagramme de Nyquist d'un système, on peut utiliser la commande *nyquist*

```
>>nyquist(H) ; grid on
```

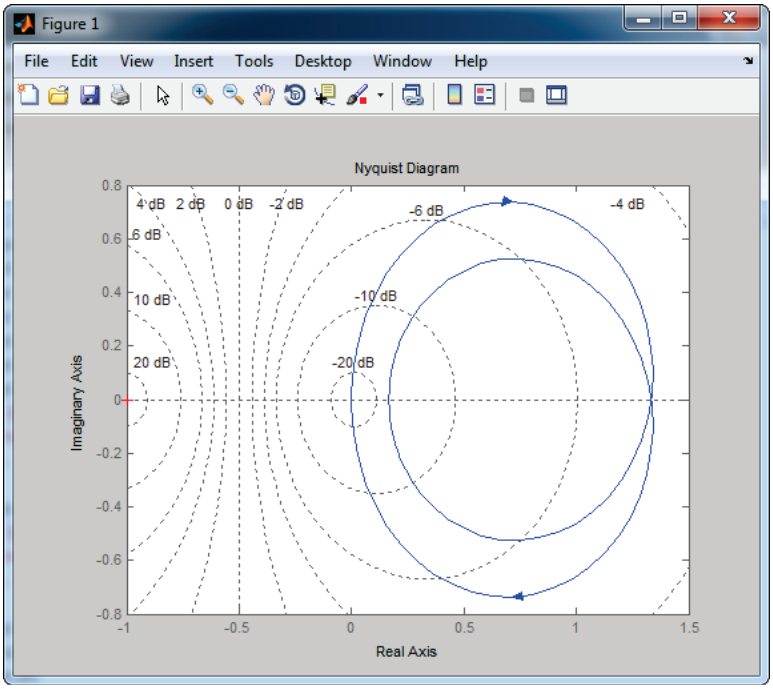


Figure 250 : diagramme de Nyquist d'un système

On peut également tracer deux diagrammes de Bode sur le même graphique.

```
>>bode(H,G) ; grid on
```

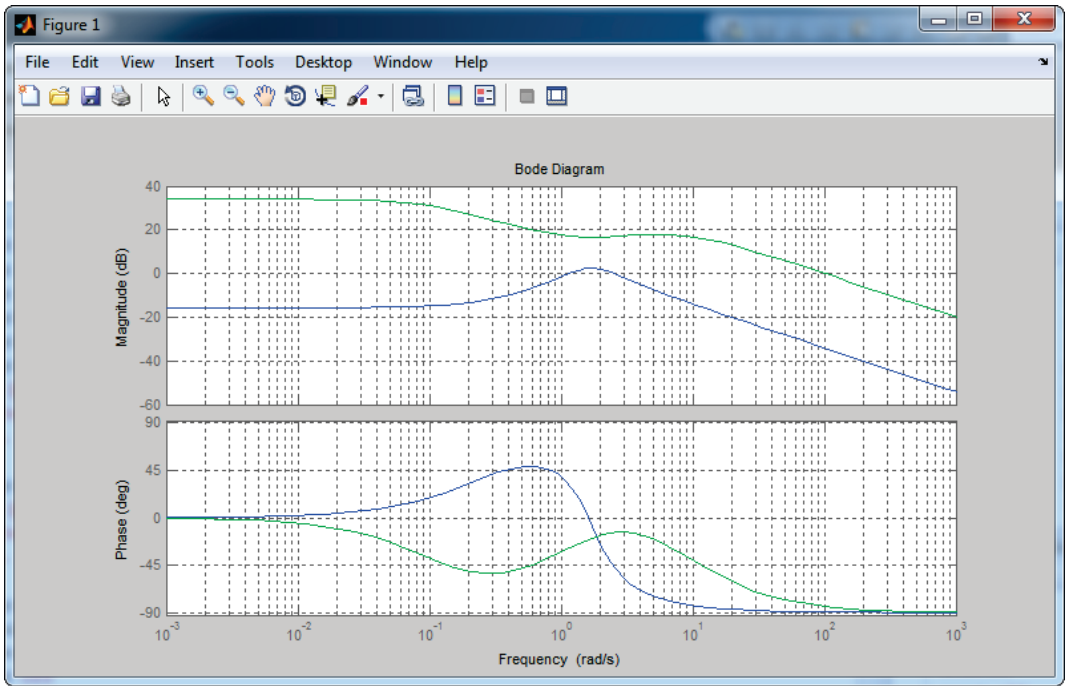


Figure 251 : tracé de deux diagrammes de Bode sur le même graphique

5. Evaluer les marges de gain et de phase

Il est possible d'obtenir automatiquement les marges de gain et de phase d'une fonction de transfert en bloucle ouverte avec la commande *margin* qui affiche un diagramme de Bode en donnant les informations numériques et graphiques sur les marges de gain et de phase.

```
>>margin(H)
```

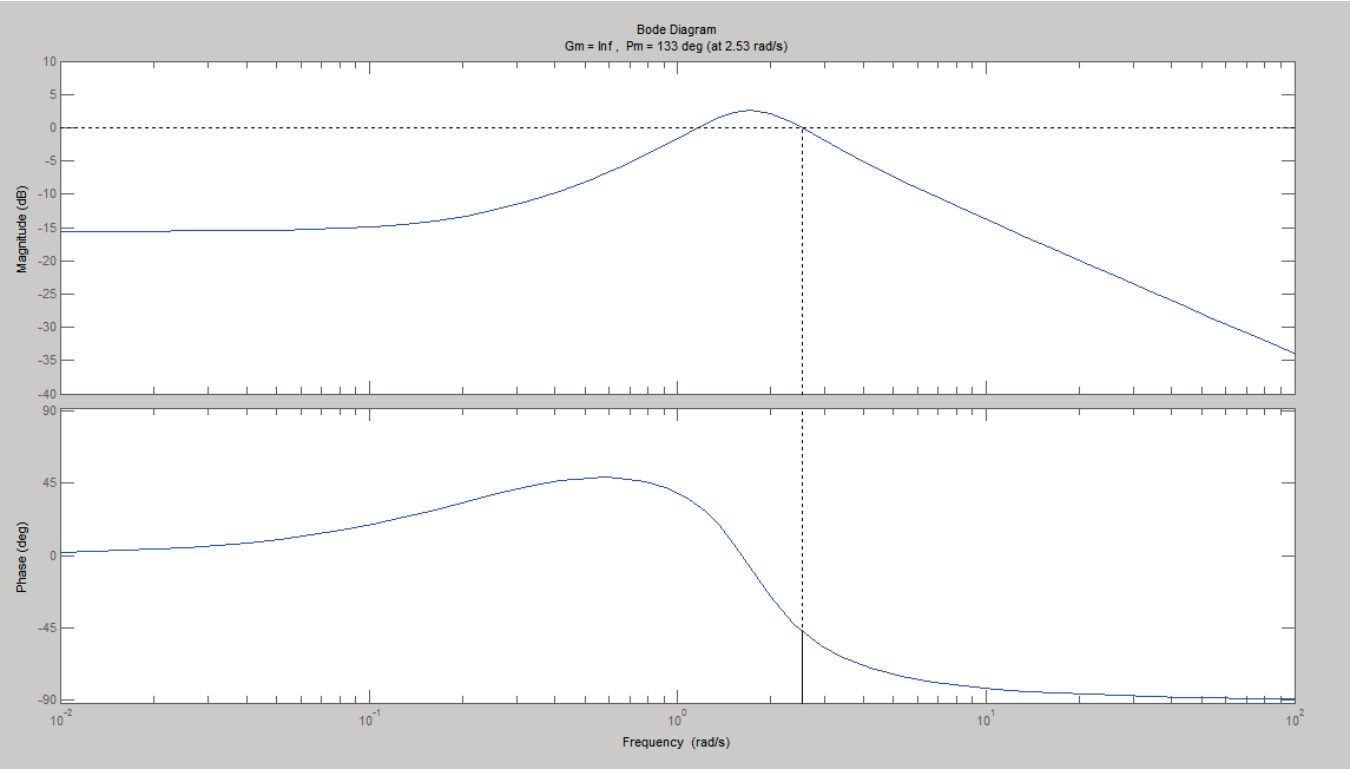


Figure 252 : diagramme de Bode avec indication des marges de gain et de phase

Il est également possible de stocker dans des variables les valeurs des marges de gain et de phase et les différentes pulsations auxquelles elles sont évaluées.

```
>>[gm,pm,wcg,wcp]=margin(H)
gm =
  Inf

pm =
 132.6226

wcg =
  NaN

wcp =

   2.5255
```

6. Tableau récapitulatif des commandes utiles sur les fonctions de transfert

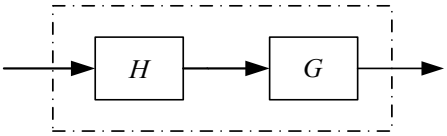
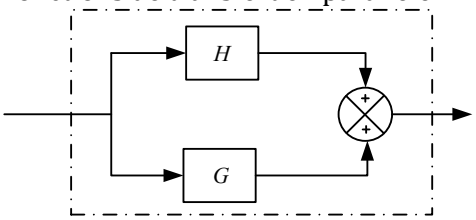
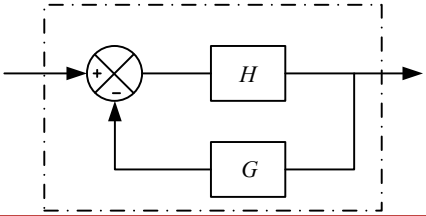
Commandes utiles	
Fonctions	Commandes
Création d'une variable symbolique	syms x
Résolution d'une équation algébrique	solve(equation,x)
Factoriser une expression	factor (x^3-4*x^2-28*x-32)
Développer une expression	expand ((x-8)*(x+2)^2)
Dériver une fonction	diff (f,t)
Intégrer une fonction	int (f,t)
Résolution d'une équation différentielle	dsolve(equation)
Calculer une intégrale définie	int(f,t,-pi/3,pi)
Calculer la transformée de Laplace d'une fonction	laplace(sin(w+t))
Calculer la transformée inverse de Laplace d'une fonction	ilaplace(3/(2+s)-5/(3*s)^2)
Décomposer en éléments simples une fraction rationnelle	[r p k]=residue (a,b)
Création d'une fonction de transfert à partir de la définition des coefficients des polynômes	H=tf([4 1],[2 3 6])
Création d'une fonction de transfert à partir de la connaissance du gain, des pôles et des zéros	G=zpk([-1,-2],[-0.1,-4,-10],100)
Indiquer que s est la variable de Laplace	s=tf('s')
Création directe d'une fonction de transfert après utilisation de la commande « s=tf('s') »	G=100*((s+1)*(s+2))/((s+0.1)*(s+4)*(s+10))
Création de la fonction de transfert d'un correcteur PID	pid(Kp,Ki,Kd)
Fonctions de transfert en série 	series(H,G)
Fonctions de transfert en parallèle 	parallel(H,G)
Fonction de transfert en boucle fermée 	feedback(H,G)
Inverser une fonction de transfert	inv(H)
Obtenir les pôles d'une fonction de transfert	pole(H)
Obtenir les zéros d'une fonction de transfert	zero(H)
Réponse indicielle	step(H)
Réponse impulsionnelle	impulse(H)

Diagramme de bode	bode(H)
Diagramme de Black-Nichols	nichols(H)
Diagramme de Nyquist	nyquist(H)
Tracé multiple de diagrammes	bode(H,G) ou nichols(H,G)…
Diagramme de bode avec marge de gain et de phase	margin(H)
Récupérer dans des variables les valeurs des marges de gain et de phase et les pulsations correspondantes	[gm,pm,wcg,wcp]=margin(H)

Pour obtenir des informations complémentaires sur une commande, vous pouvez utiliser l'aide de **MATLAB** en utilisant la commande doc.

```
>>doc bode        permet d'ouvrir la fenêtre d'aide de la commande « bode »
```

Chapitre 4 : Prise en main de Simulink

I. Introduction

Ce chapitre a pour objectif de présenter au travers d'un exemple une modélisation effectuée avec **Simulink** et de montrer les communications possibles entre l'environnement **MATLAB** et l'environnement **Simulink**.

II. Régulation en température d'un four

L'étude porte sur l'asservissement en température d'un four industriel.

Une résistance chauffante permet de chauffer l'enceinte du four. Un capteur informe la carte de commande de la température à l'intérieur du four. Cette mesure est comparée à la consigne de température pour former un écart. Un correcteur proportionnel et intégral impose une tension de commande à la résistance.

A. Ouverture du modèle

Ouvrir le fichier ***four_0.slx*** pour visualiser le schéma bloc de la Figure 253 et explorer les différents blocs du modèle.

Ce modèle **Simulink** représente l'asservissement en température d'un four et les éléments modélisés par les différents blocs sont indiqués sur la Figure 253.

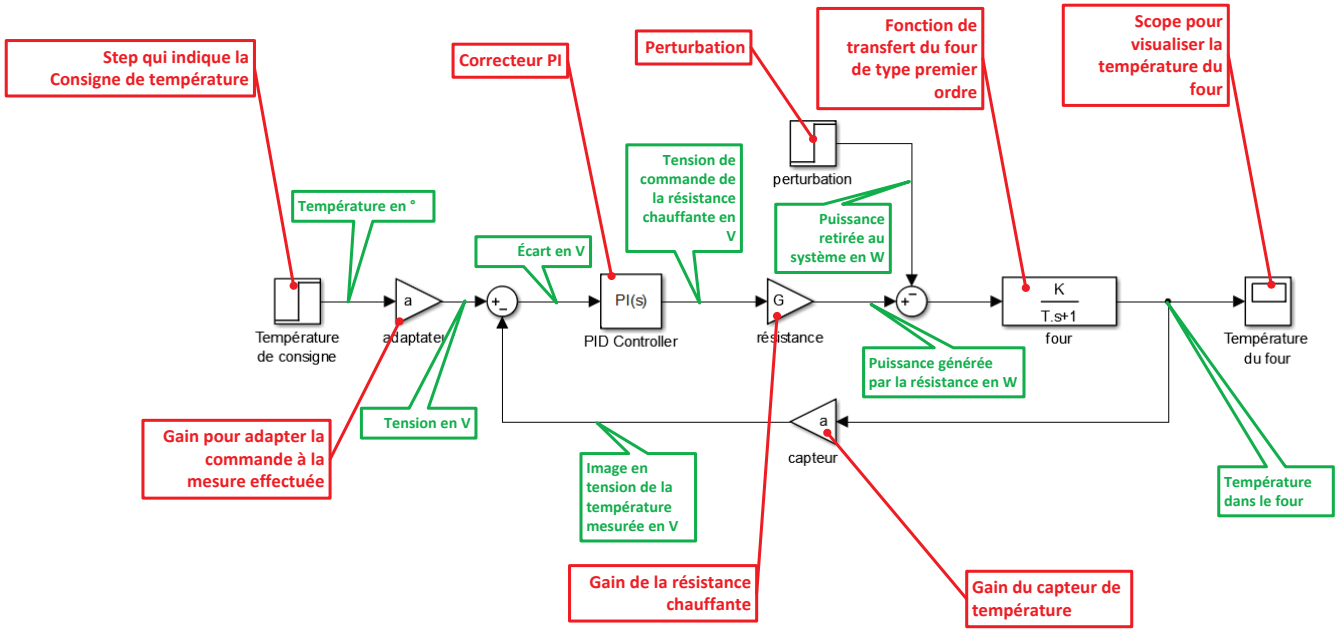


Figure 253 : modélisation de l'asservissement en température d'un four

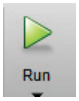
Ce modèle ne peut pas être exécuté pour l'instant dans la mesure où toutes les grandeurs physiques du modèle sont représentées par des variables qu'il faudra définir avant de lancer la simulation.
Il est très utile de regrouper la valeur de toutes les grandeurs physiques du système dans un script unique. L'exécution du script permettra la création de toutes les variables dans le **Workspace** de **MATLAB**. Il sera alors possible de lancer la simulation dans Simulink.

B. Ouverture du script contenant la définition des variables

Ouvrir le script ***parametres_four.m***, en cliquant sur **Open script** 


```
1     %gain du capteur
2 -   a=0.5;
3     %tension maxi aux bornes de la résistance
4 -   Vsat=100;
5     %gain de la resistance chauffante
6 -   G=0.8;
7     %paramètres de la fonction de transfert du four
8 -   K=15;
9 -   T=2500;
10    %paramètres du correcteur
11 -  Kp=10;
12 -  Ki=0.01;
```

Figure 254 : script contenant les paramètres de la simulation

Exécuter le script, en cliquant sur **Run** 

dans la barre de commande.

On constate que les variables ont été créés et sont visible dans le Workspace de **MATLAB**

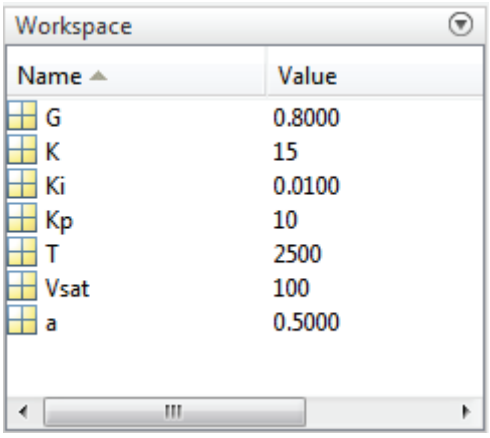


Figure 255 : visualisation des variables créées dans le Workspace

C. Lancement de la simulation

Lancer la simulation du modèle Simulink et visualiser la variation de la température à l'intérieur du four

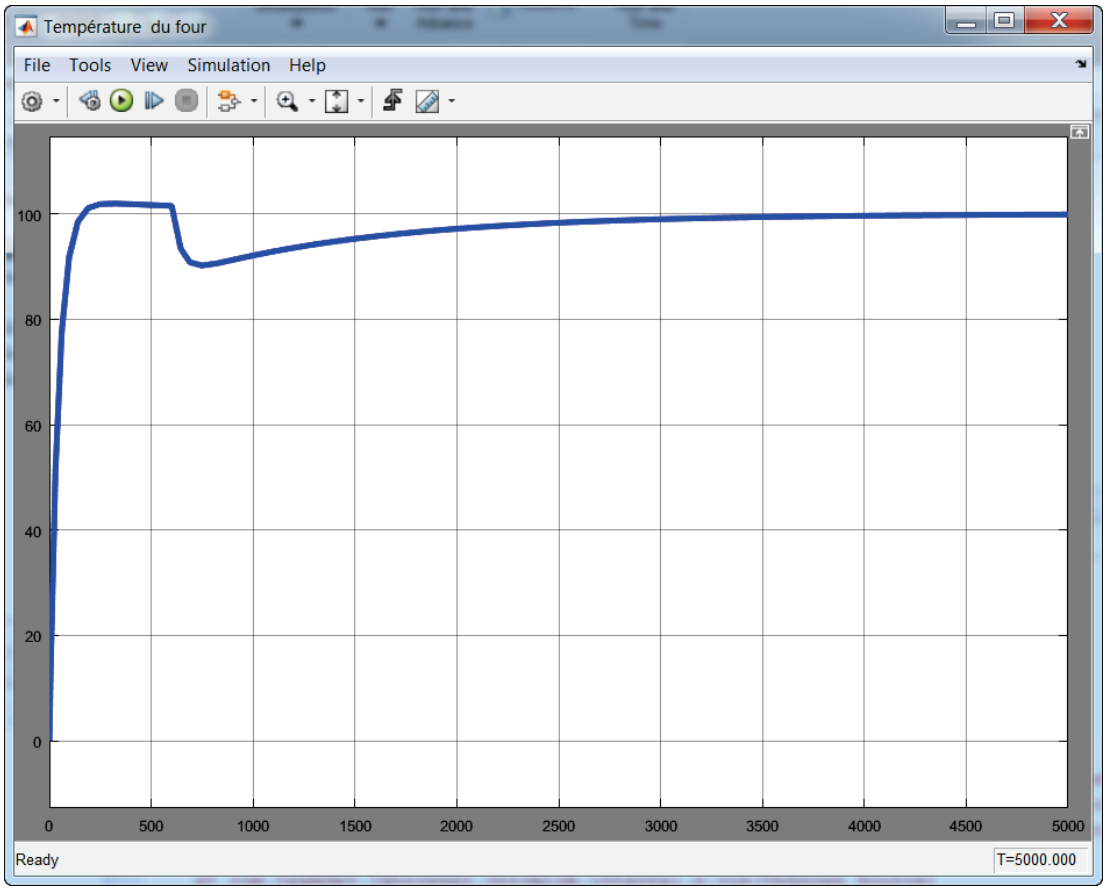


Figure 256 : réponse en température du four

Une consigne de 100° est imposée à l'entrée du modèle. Une perturbation est visible à l'instant t=600 s et se caractérise par une chute de la température. Le correcteur proportionnel et intégral permet de rétablir la température du four au niveau de la consigne.

D. Tracer un diagramme de Bode avec Simulink

Il existe de nombreuses méthodes pour tracer un diagramme de Bode à partir d'un schéma bloc réalisé avec Simulink. La plus simple consiste à placer des **points de linéarisation** sur le schéma bloc et d'utiliser le bloc **Bode Plot** de la bibliothèque **Simulink Control Design**. Il existe différents types de points de linéarisation, nous allons voir comment les choisir pour obtenir un diagramme de Bode en boucle ouverte et en boucle fermée.
Avant de commencer il est préférable de donner un nom aux signaux du schéma bloc qui interviendront dans les fonctions de transfert à tracer.
Nommer les signaux conformément à la Figure 257 :

- Entrée
- Sortie
- Ecart
- Retour

Pour nommer un signal **cliquer avec le bouton droit** sur le signal puis choisir **Properties**. La boite de dialogue **Signal Properties** s'ouvre. Le nom du signal est à spécifier dans **Signal Name**.

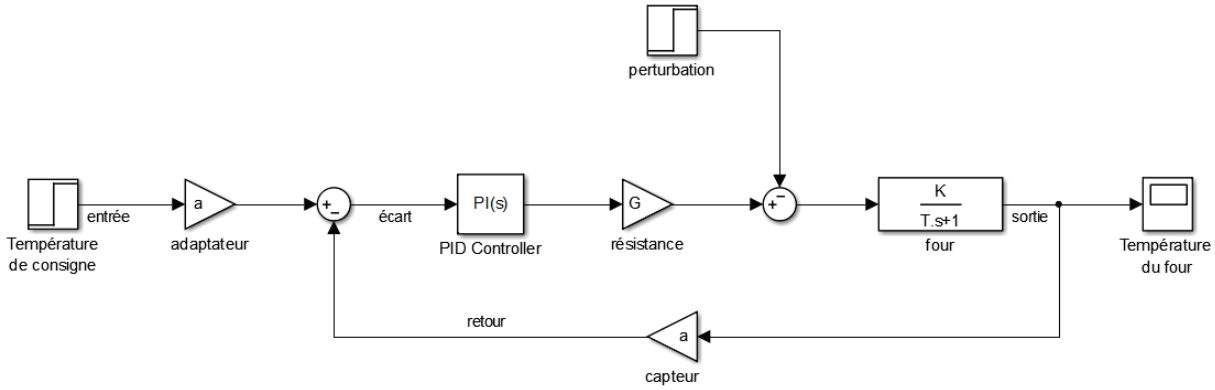


Figure 257 : nommer un signal Simulink

1. Tracer un digramme de Bode en boucle ouverte

Pour tracer un diagramme de Bode en boucle ouverte, il faut placer deux points de linéarisation sur les signaux d'entrée et de sortie de la boucle ouverte. Pour cela il faudra choisir des points de linéarisation de type :

- **Open loop input** (pour l'entrée de la boucle ouverte)
- **Open loop output** (pour la sortie de la boucle ouverte)

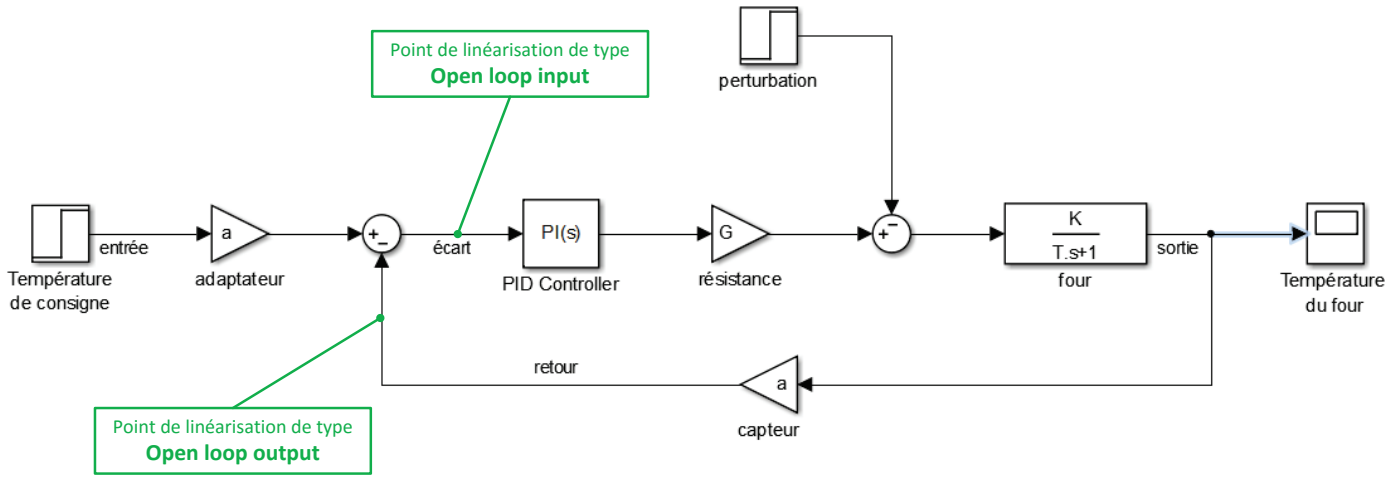


Figure 258 : placement des points de linéarisation pour tracer un diagramme de Bode en boucle ouverte

Pour placer un point de linéarisation de type **open loop input** sur le signal « **écart** », il faut **cliquer avec le bouton droit** sur le signal « **écart** », puis choisir **Linear Analysis point/open loop Input.**

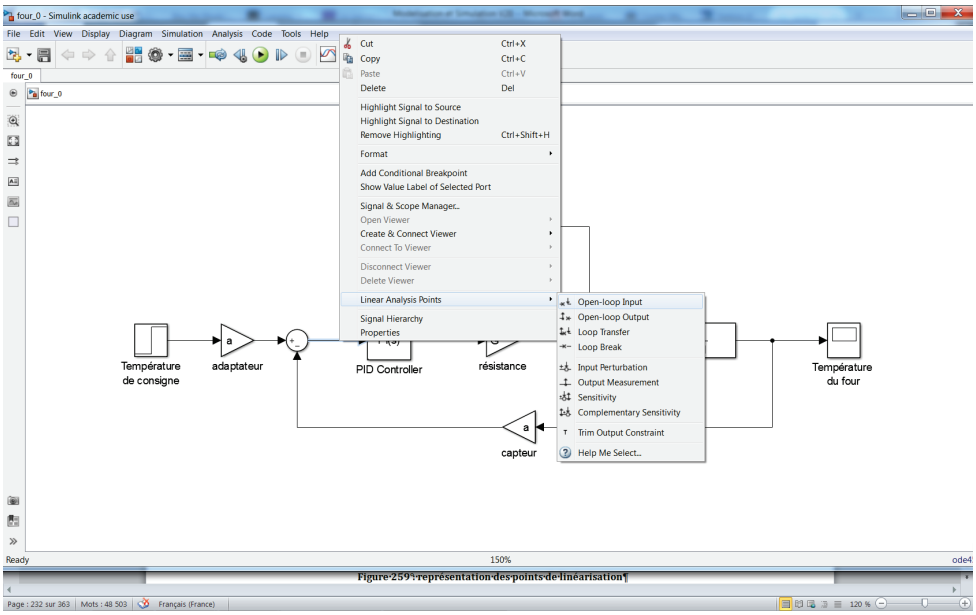


Figure 259 : placement d'un point de linéarisation de type « Open loop input »

Pour placer un point de linéarisation de type **open loop output** sur le signal « **retour** », il faut **cliquer avec le bouton droit** sur le signal « **retour** », puis choisir **Linear Analysis point/open loop Output.**

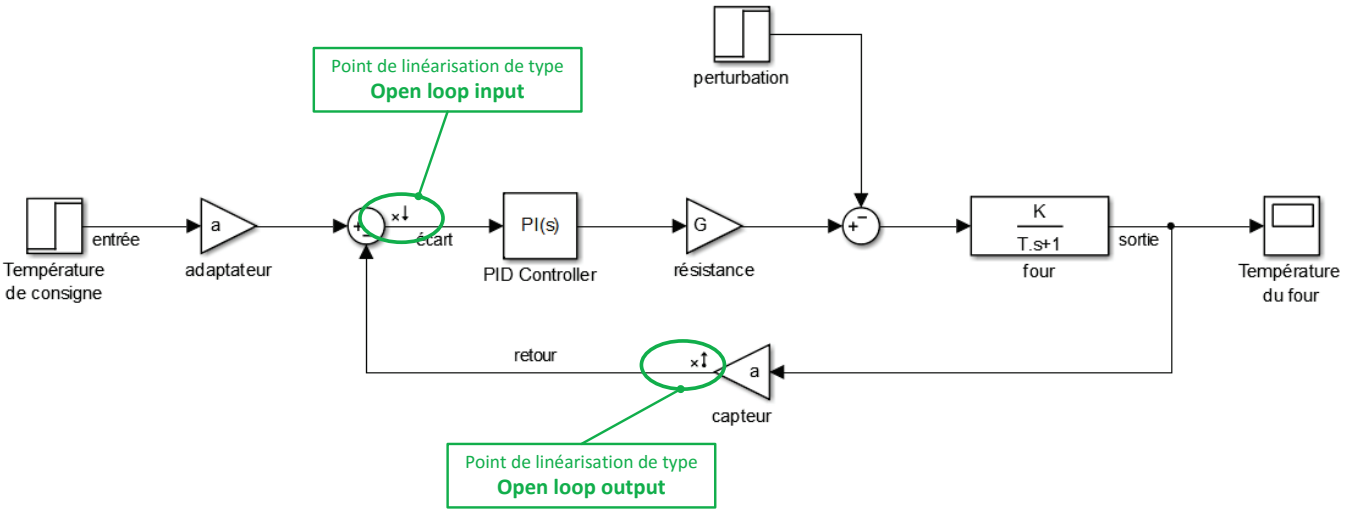
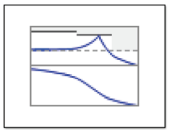


Figure 260 : représentation des points de linéarisation

Fonction du composant	Représentation	Bibliothèque
Tracer d'un diagramme de Bode	 Bode Plot	Simulink/Linear Analysis Plot

Insérer dans votre modèle un bloc **Bode Plot**, renommer ce bloc « **boucle ouverte** ».

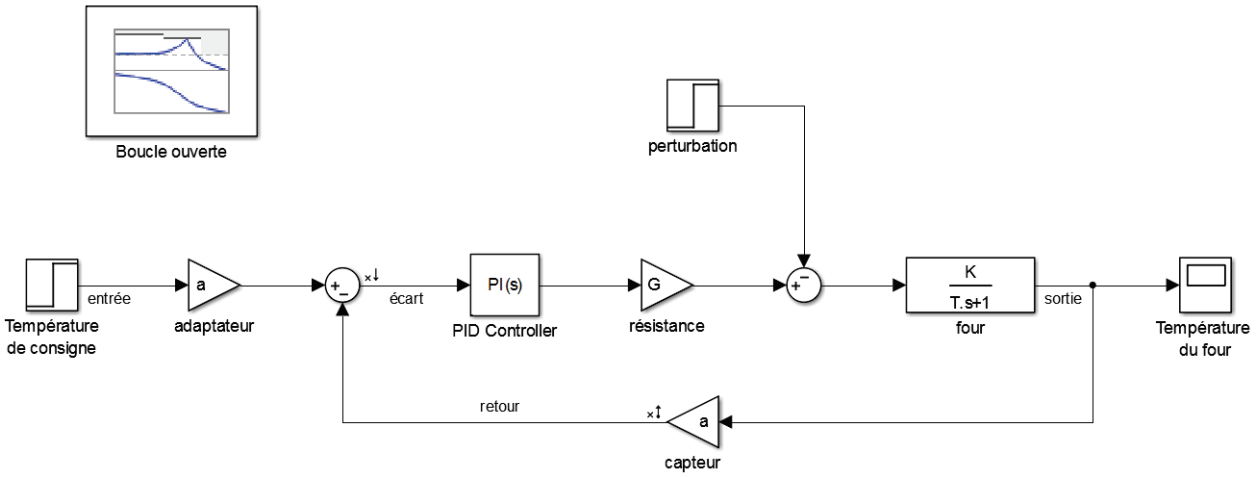


Figure 261 : insertion d'un bloc Bode Plot

Double cliquer sur le bloc pour l'ouvrir.

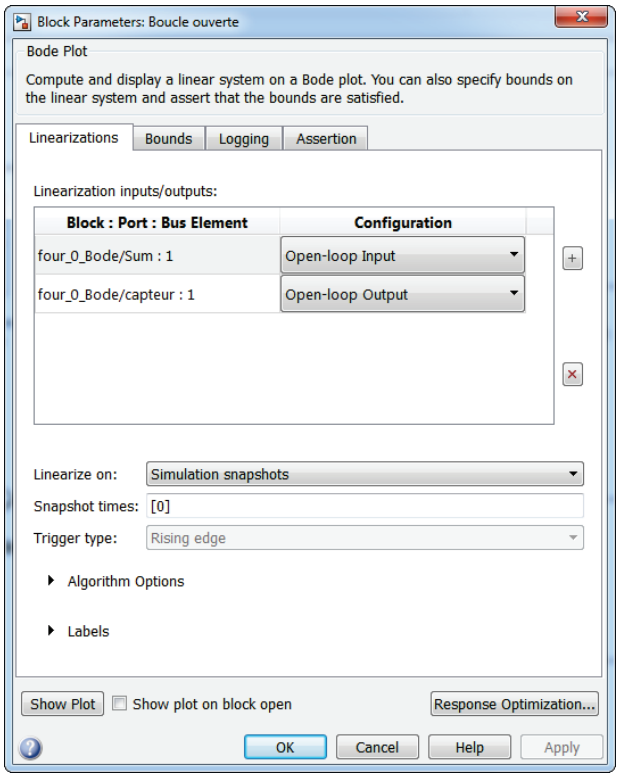


Figure 262 : fenêtre de paramétrage du bloc Bode Plot

Par défaut les deux points de linéarisation créés apparaissent.
Cliquer sur **Show Plot** pour faire apparaître la fenêtre de tracé.

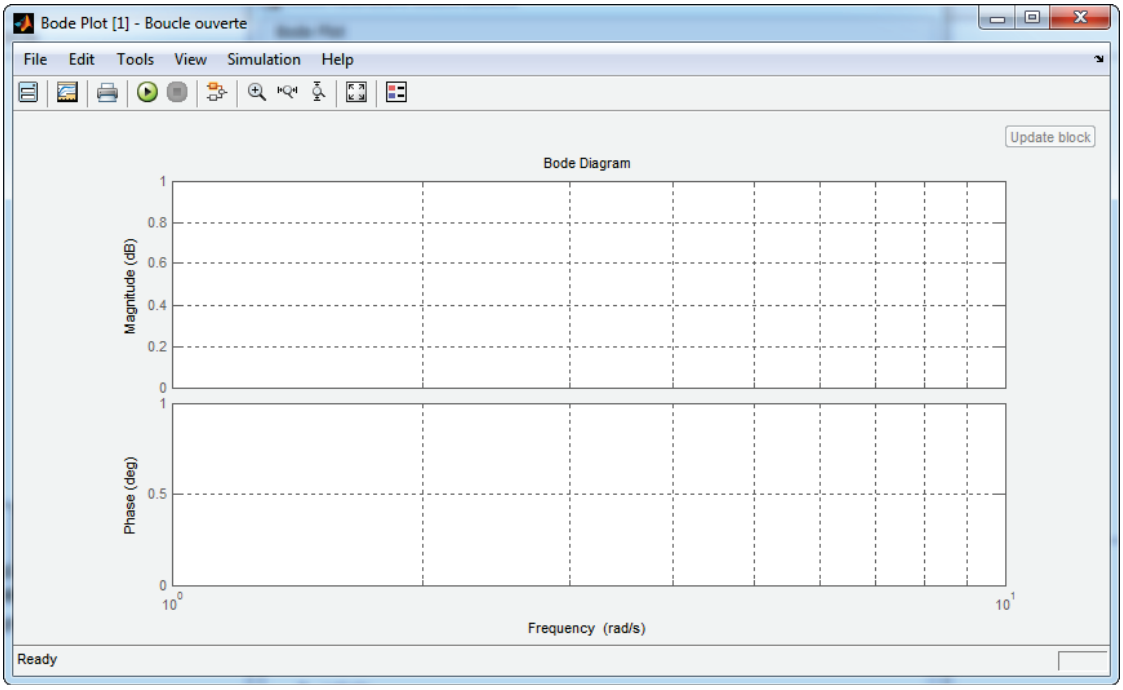


Figure 263 : fenêtre de tracé du diagramme de Bode

Cliquer sur Run  pour linéariser le modèle. Le diagramme de Bode se trace automatiquement.

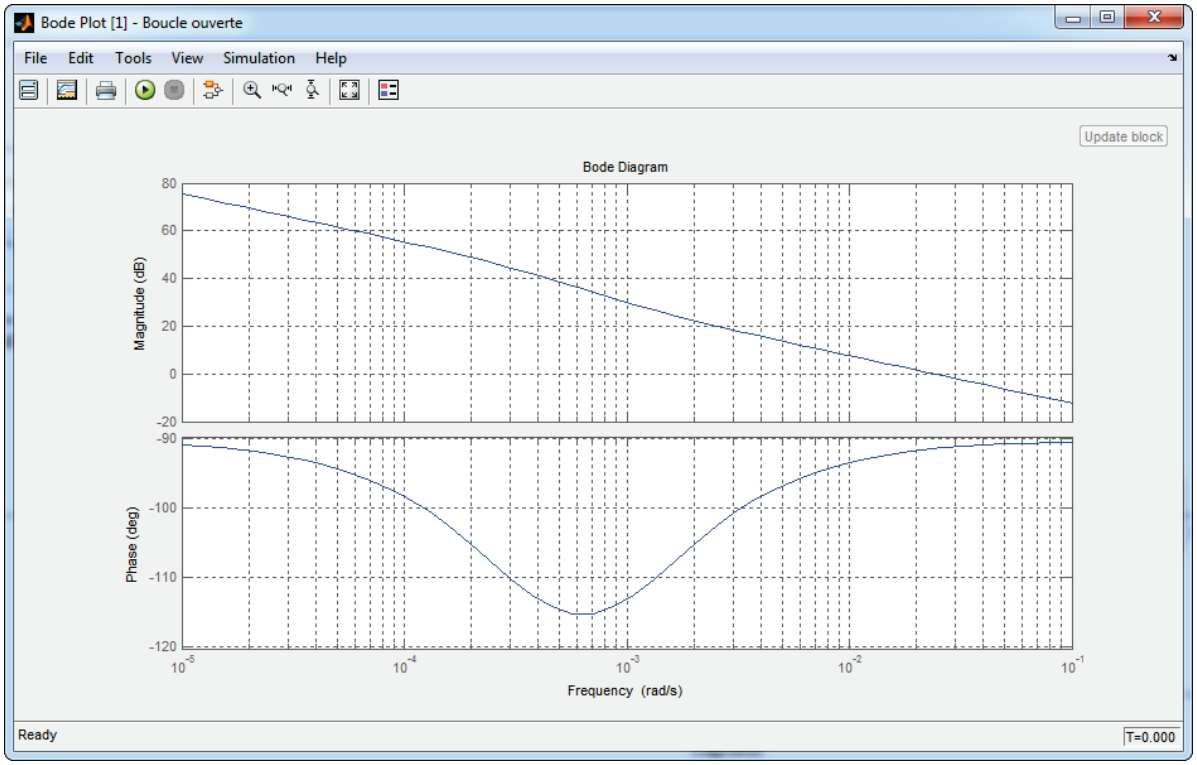


Figure 264 : diagramme de Bode de la boucle ouverte

2. Tracer un diagramme de Bode en boucle fermée

Pour tracer un diagramme de Bode en boucle fermée, il faut placer deux points de linéarisation sur les signaux d'entrée et de sortie de la boucle fermée. Pour cela il faudra choisir des points de linéarisation de type :

- **Input Perturbation** (pour l'entrée de la boucle fermée)
- **Output Measurement** (pour la sortie de la boucle fermée)

Pour placer un point de linéarisation de type **Input Perturbation** sur le signal « **entrée** », il faut **cliquer avec le bouton droit** sur le signal « **entrée** », puis choisir **Linear Analysis point/Input Perturbation.**

Pour placer un point de linéarisation de type **Output Measurement** sur le signal « **sortie** », il faut **cliquer avec le bouton droit** sur le signal « **sortie** », puis choisir **Linear Analysis point/Output Measurement.**

Les différents points de linéarisation apparaissent sur la Figure 265.

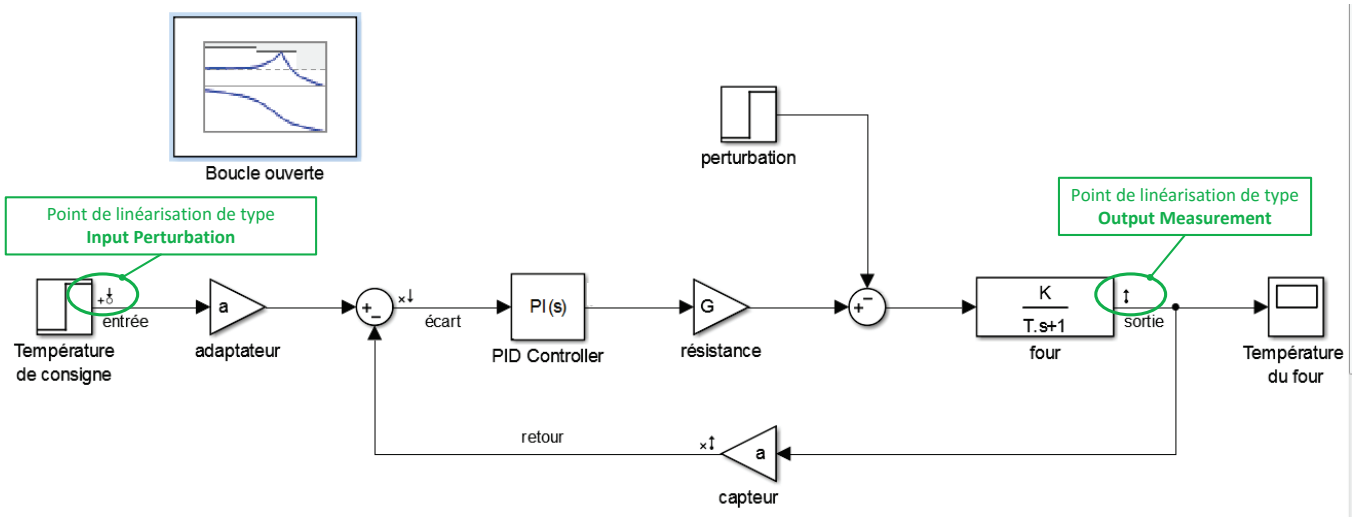


Figure 265: représentation des points de linéarisation

Insérer dans votre modèle un second bloc **Bode Plot**, **renommer** ce bloc « **boucle fermée** ».

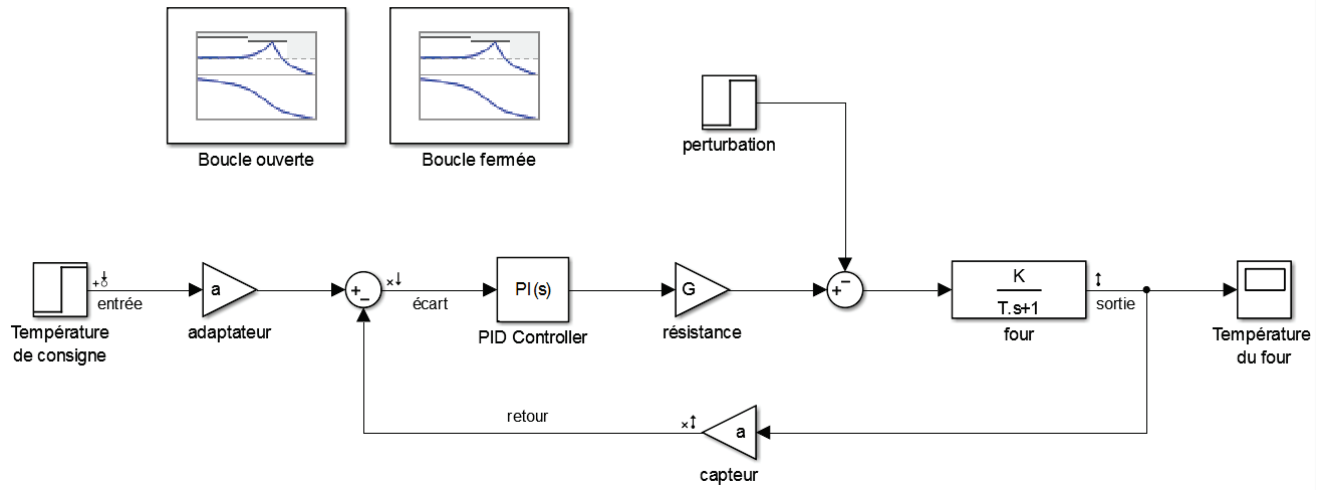


Figure 266 : insertion d'un second bloc Bode plot

Double cliquer sur le bloc pour l'ouvrir.

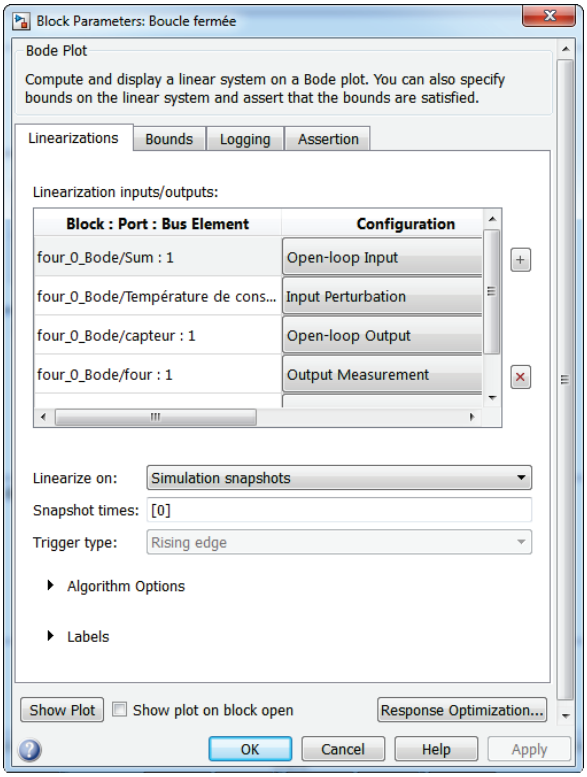


Figure 267: fenêtre de paramétrage du bloc Bode Plot

Par défaut les quatre points de linéarisation créés apparaissent.
Il suffit de garder les deux points de linéarisation correspondant à la boucle fermée et de supprimer les deux points de linéarisation correspondant à la boucle ouverte.
Pour cela, sélectionner les points de linéarisation à supprimer et utiliser le bouton **Delete Selected Linearization I/Os**  pour obtenir la configuration de Figure 268**.**

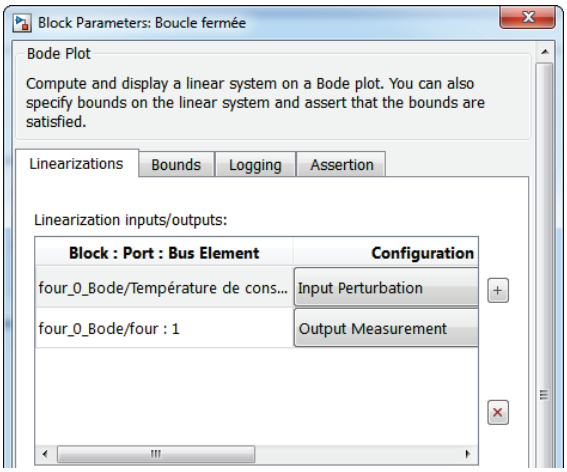


Figure 268 : paramétrage des points de linéarisation pour la boucle fermée

Cliquer sur **Show Plot** pour faire apparaître la fenêtre de tracé.

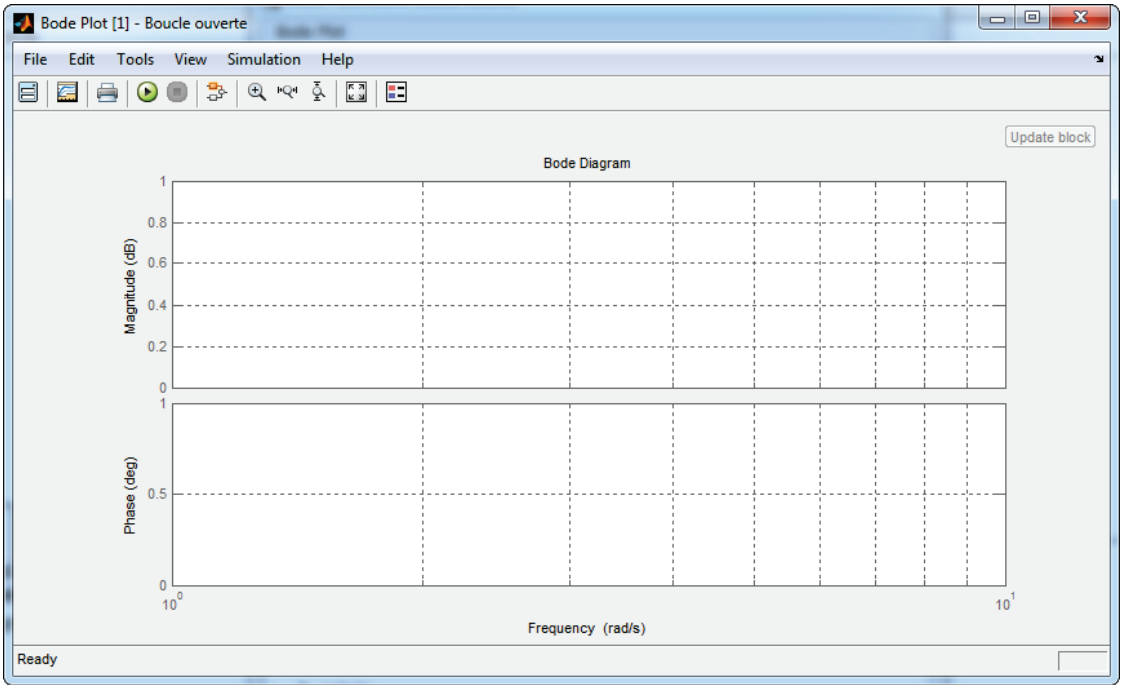


Figure 269 : fenêtre de tracé du diagramme de Bode

Cliquer sur **Run**  pour linéariser le modèle. Le diagramme de Bode se trace automatiquement.

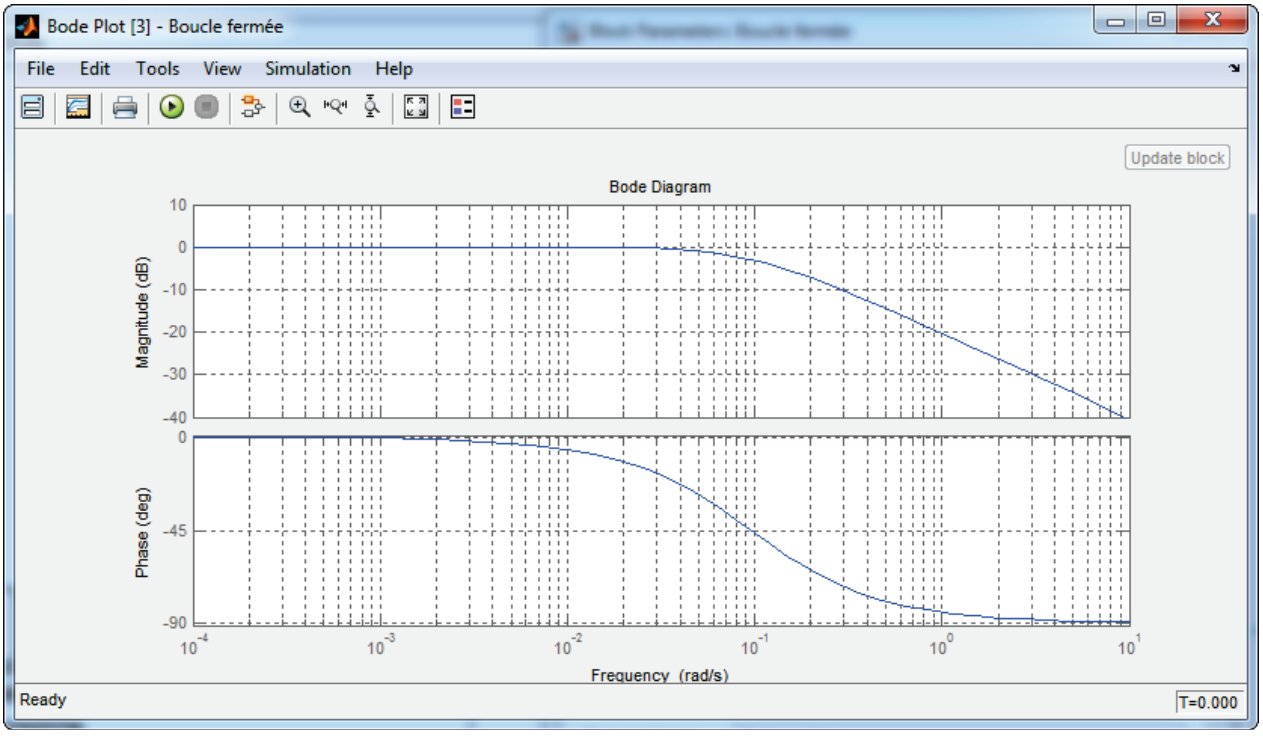
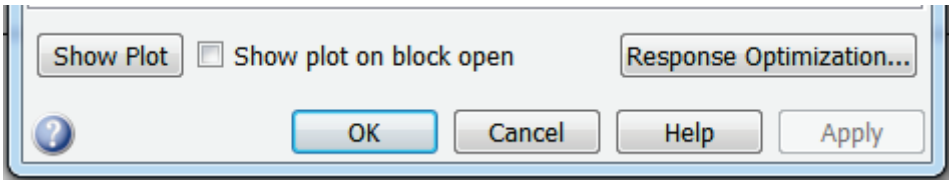


Figure 270 : diagramme de Bode de la boucle fermée

Si l'on souhaite ouvrir la fenêtre graphique du diagramme de Bode par un double clic sur le bloc **Bode Plot**, il suffit de cocher la case **Show plot on block open.**



Nous disposons alors du modèle Simulink que l'on peut modifier et visualiser très rapidement l'allure des diagrammes de Bode.

E. Tracer d'un diagramme de Black-Nichols

La méthode est rigoureusement la même que pour tracer un diagramme de Bode, en utilisant le bloc **Nichols Plot**.

Le fichier contenant le modèle complet est disponible sous le nom ***four_0_Bode_Nichols.slx***

F. Ajout et paramétrage d'une saturation

Nous allons ajouter un saturateur pour tenir compte de la limite de 100 V représentant la tension maximale de commande de la résistance.
Les non-linéarités (seuil, hysteresis…) de la bibliothèque Simulink se trouvent dans **Simulink/Discontinuities.**

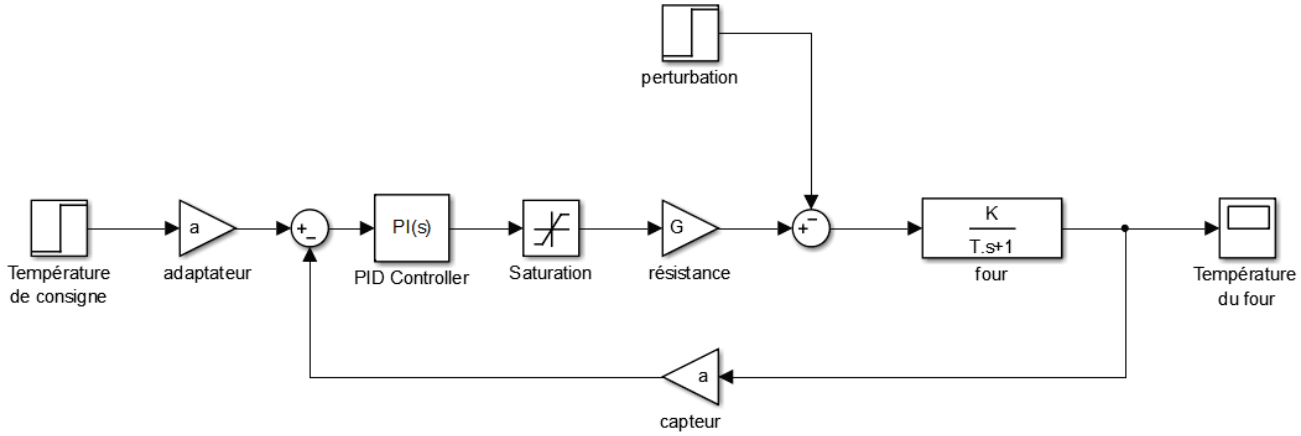
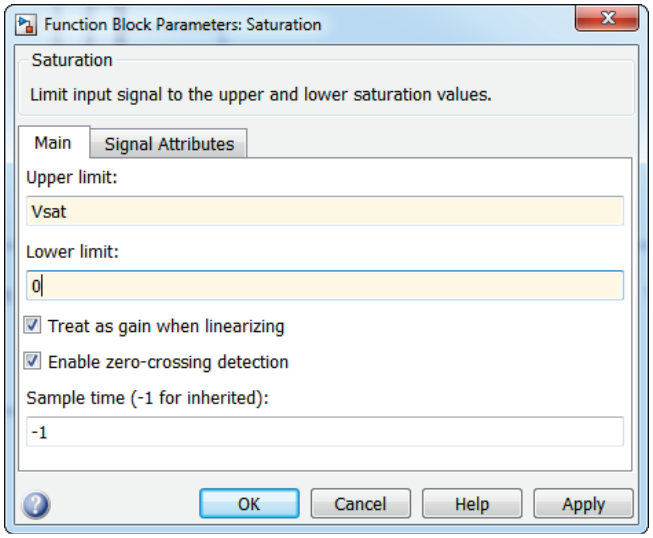


Figure 271 : asservissement en température d'un four avec saturation

Paramétrage

Saturation  Simulink/Discontinuities

Ce composant permet de limiter la sortie du bloc entre deux valeurs mini et maxi de saturation
Ici la valeur maximale est donnée par la variable Vsat, la valeur mini étant laissée à 0.



Lancer la simulation et visualiser la réponse obtenue avec la saturation.

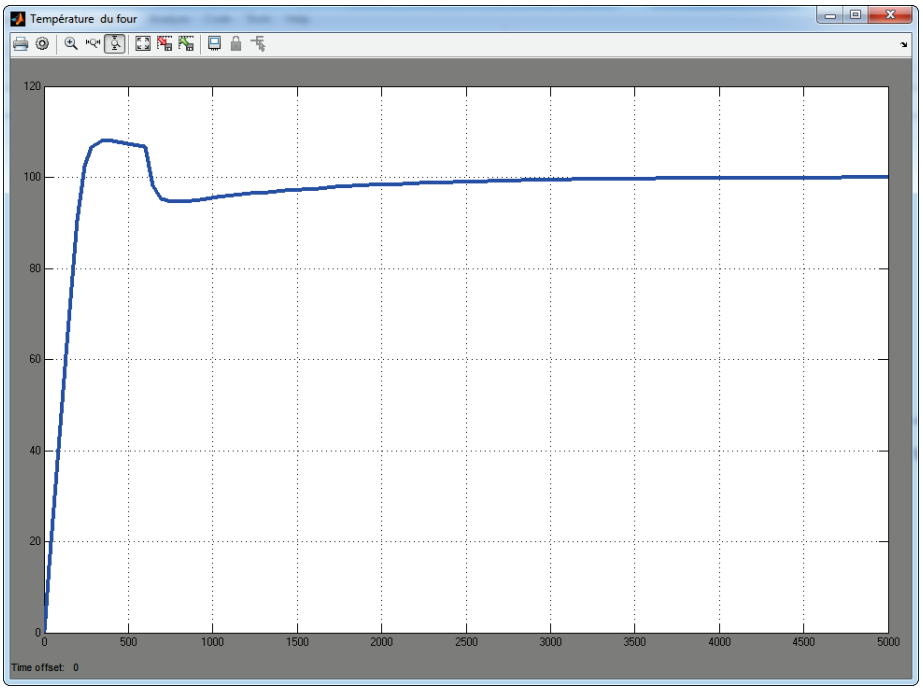
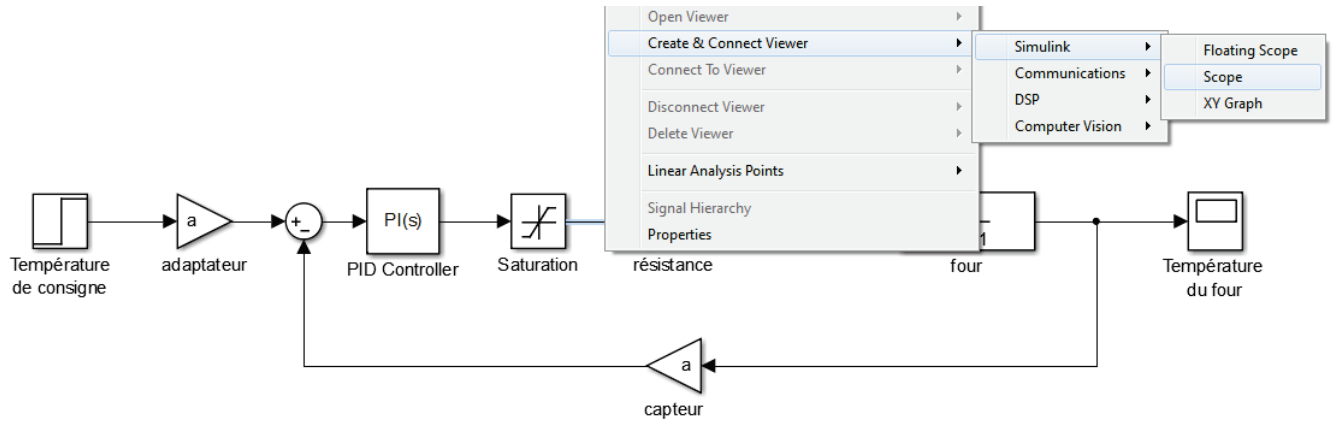


Figure 272 : réponse en température de four avec saturation

Nous constatons que la saturation ralentit le système, le temps de réponse à 5% est plus élevé.
Pour visualiser la tension en sortie de saturateur il est possible de placer un scope classique. Cependant afin de visualiser des signaux sans avoir à surcharger la modélisation MATLAB propose d'autre mode de visualisation en plaçant un scope directement sur un signal.

Pour cela **cliquer droit** sur le fil du signal puis sélectionner **Create&connect Viewer/Simulink/Scope**



Un petit scope apparaît sur le signal en sortie de saturateur.

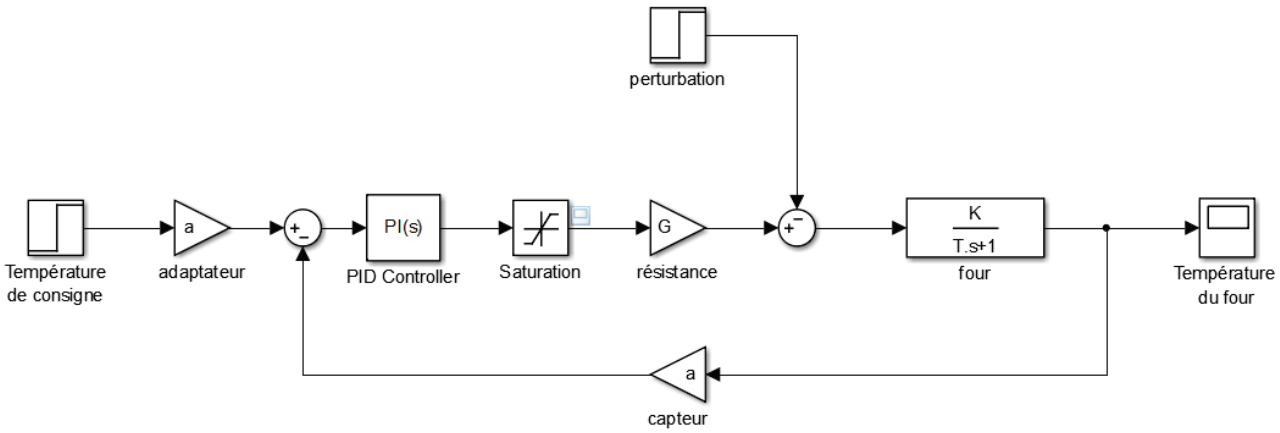


Figure 273 : mise en place d'un scope sur un signal

Relancer la simulation et double cliquer sur le scope du saturateur pour visualiser la tension en sortie du bloc.

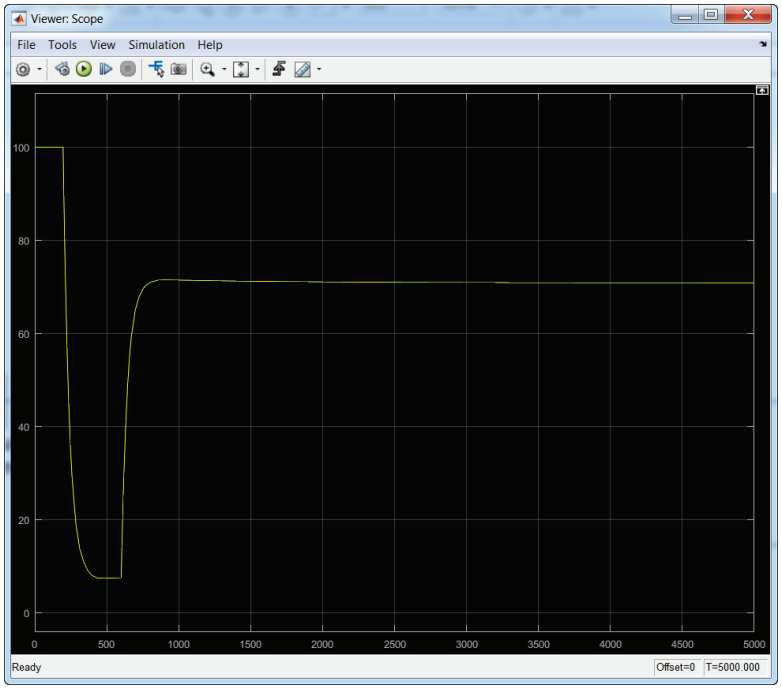


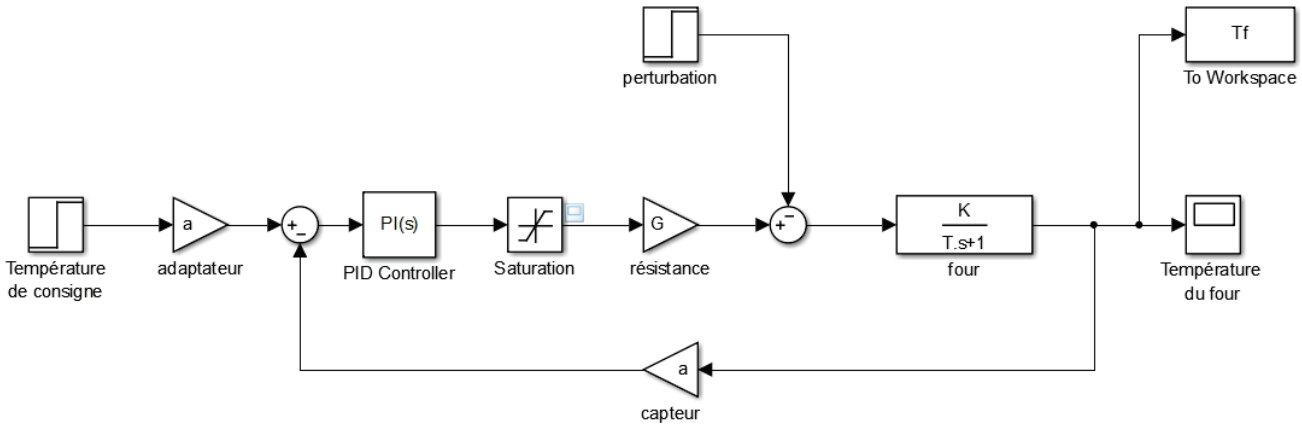
Figure 274 : visualisation de la tension en sortie de saturateur

Nous constatons que la tension sature à 100 V durant environ 200 s ce qui explique le ralentissement du système.

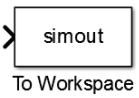
G. Exportation des variables de la simulation vers le Workspace

Il est très souvent utile d'exporter des variables qui contiennent le résultat de simulation vers le **Workspace**. Cela permet d'utiliser toutes les fonctions de **MATLAB** pour la présentation des résultats.

Nous souhaitons tracer un réseau de courbes permettant de voir l'influence du gain **Kcor** du correcteur proportionnel sur la température du four. Il faudra pour cela créer une variable **Tf** correspondant à la température du four et exporter cette variable vers le Workspace à l'aide d'un bloc **To Workspace** que l'on trouve dans la bibliothèque **Simulink/Sinks**.

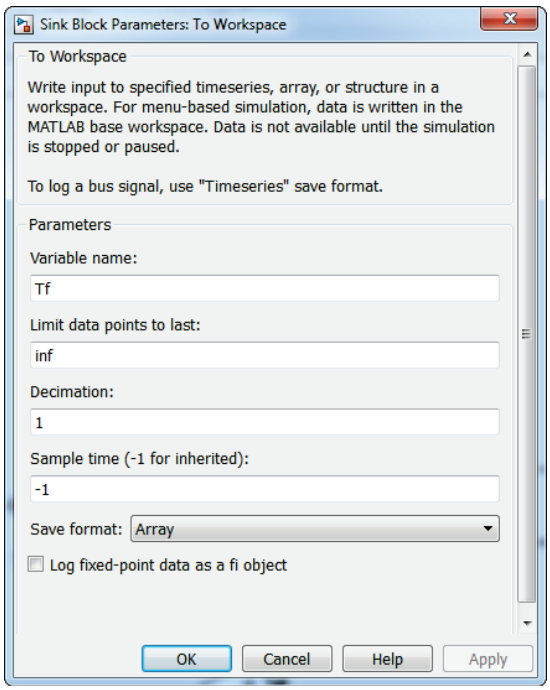


Exportation de variables vers le Workspace



Simulink/Sinks

Ce composant permet de créer une variable pour un signal **Simulink** et de l'exporter vers le Workspace de **MATLAB**. Il suffit de spécifier le nom de la variable (ici **Tf**) et son format. Pour exporter la variable sous la forme d'un simple vecteur il faut choisir le format **Array**.



Pour pouvoir tracer ensuite la variable **Tf** en fonction du temps, il faudra un second vecteur représentant les abscisses et qui sera constitué des différentes valeurs du temps correspondants aux pas de calcul du solveur. Cette variable est générée automatiquement par Simulink et exporter vers le Workspace. Cette variable est nommée par défaut **tout**.

Pour modifier le nom de cette variable **tout**, **cliquer** sur la configuration du solveur de Simulink  et choisir l'onglet **Data Import/Export** dans la partie gauche de la fenêtre. Remplacer **tout** par **t**.

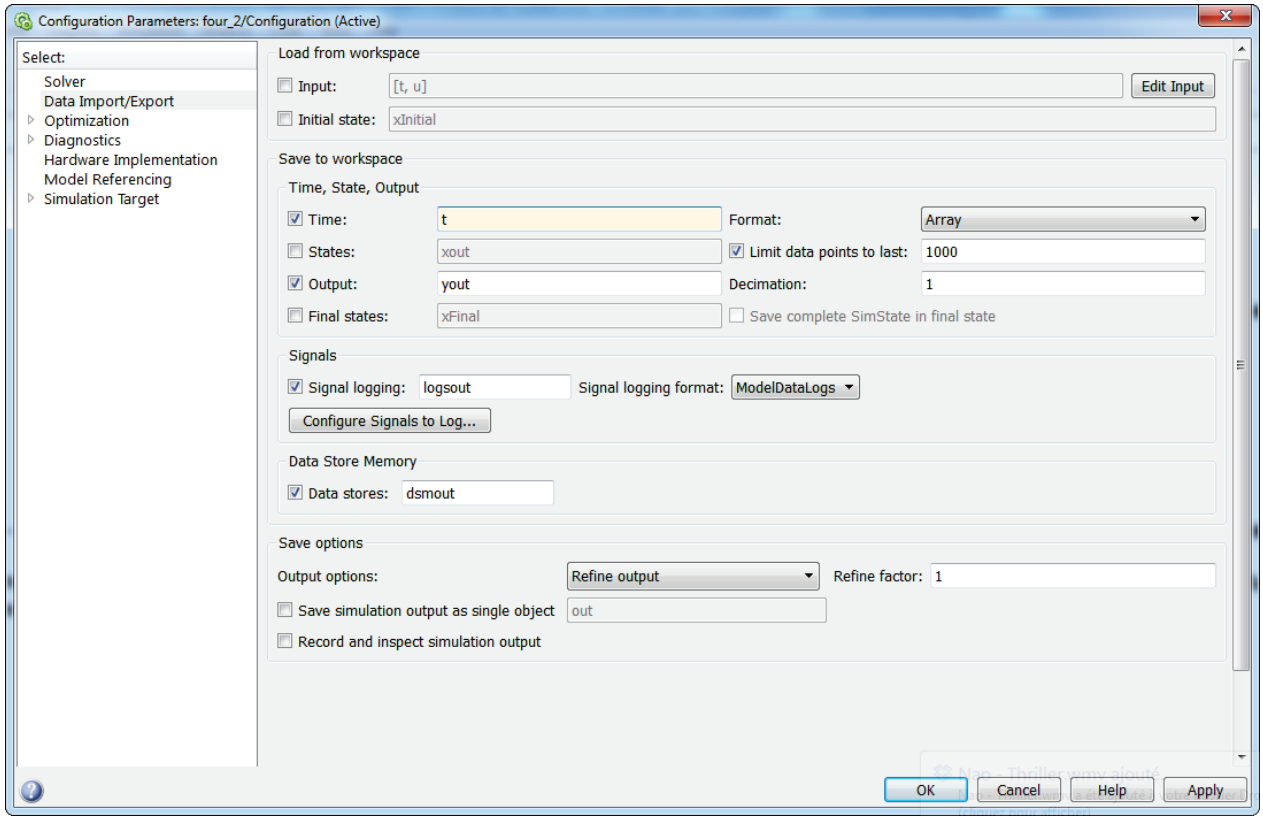


Figure 275 : fenêtre de paramétrage de l'exportation des données vers le Workspace

Enregistrer le modèle sous le nom ***four_expor_name.slx.***

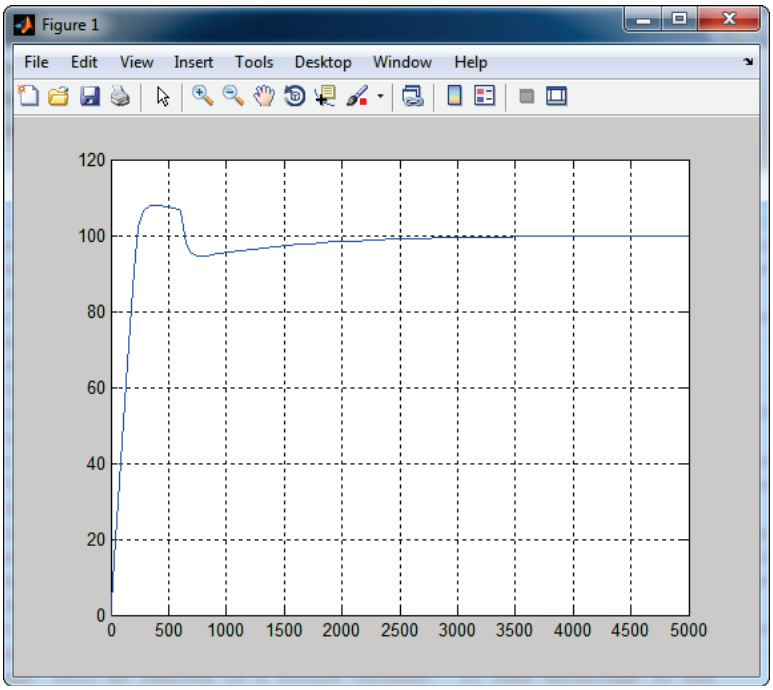
Si nécessaire, le modèle complet est disponible dans le fichier ***four_export.slx.***

Lancer la simulation.

Observer le Workspace de **MATLAB** et vérifier la création des variables t et Tf.

Dans la fenêtre de commande de MATLAB taper la commande suivante :
>>plot(t,Tf) ;grid on ;

La courbe représentant la température en fonction du temps apparaît.



1. Ecriture d'un script pour tracer une série de courbes

La commande ***sim*** permet de lancer une simulation d'un modèle Simulink à partie d'une ligne de commande. Nous allons l'utiliser pour écrire un script permettant de visualiser l'influence du gain du correcteur proportionnel.

Taper le script suivant**, enregistrer le** et **exécuter le**.

```
1 -    hold all;
2 -    grid on;
3      % pour Kp variant de 1 à 10 avec un pas de 1 (valeur par défaut du pas)
4      % exécute la simulation du fichier simulink four_import
5      % trace la température en fonction du temps
6 -    for Kp=1:10
7 -        sim('four_export');
8 -        plot(t,Tf,'LineWidth',2);
9 -    end
10
```

Figure 276 : script pour tracer une série de courbes

Le script est disponible dans le fichier ***script_courbes_Kcor.slx.***

Vous devez obtenir le résultat suivant.

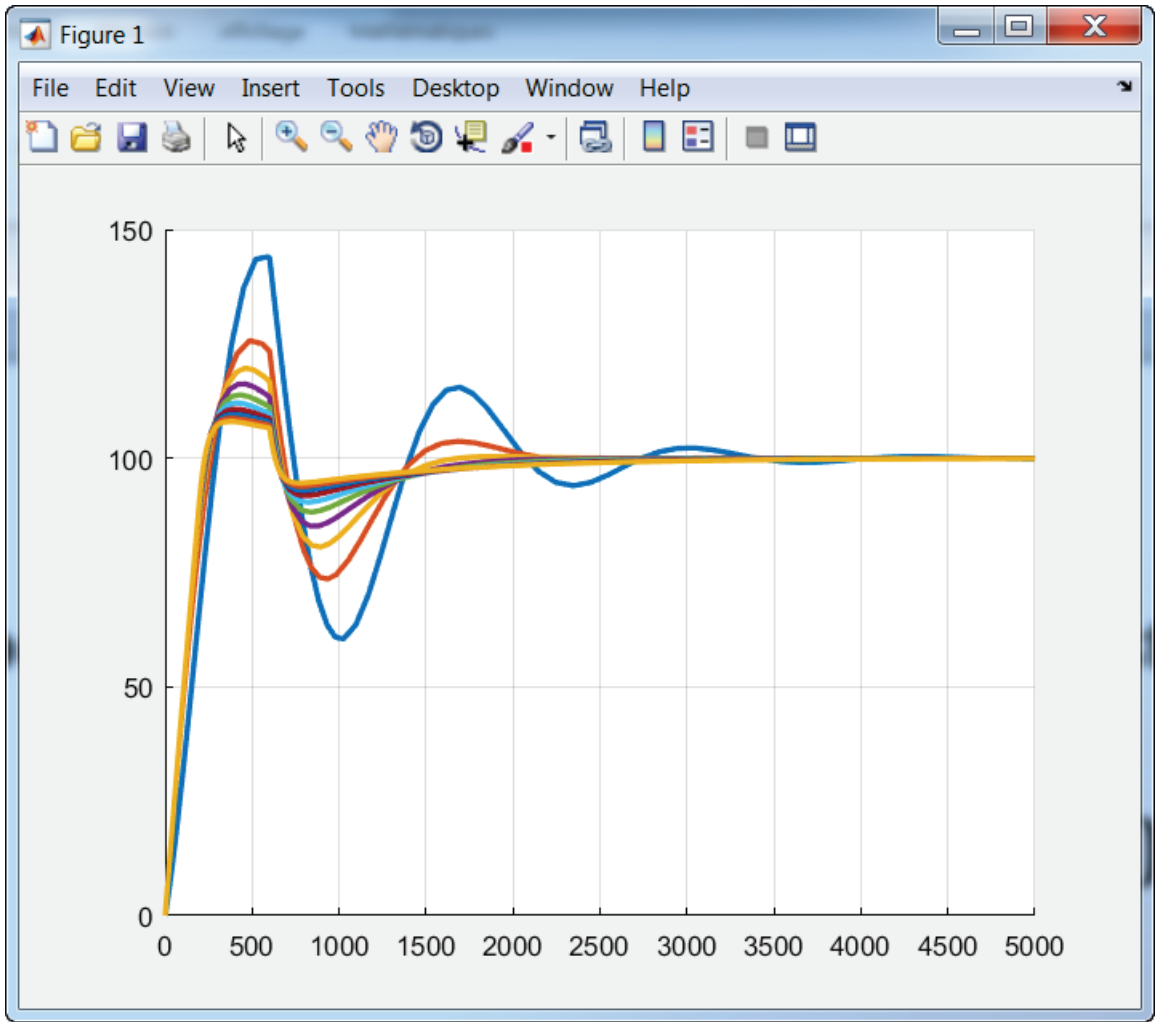


Figure 277 : influence de la correction proportionnelle sur la température du four

Chapitre 5 : Prise en main de Stateflow

I. Introduction à Stateflow

Stateflow est le module de MATLAB permettant de modéliser les comportements séquentiels des systèmes sous la forme de machines à états.
Une machine à état est constituée « d'états » qui représentent les différents modes du système et de « transitions » qui traduisent les conditions de passage d'un état à un autre. Lors de l'activation d'un état, des actions peuvent être effectuées par le système, elles sont alors indiquées dans l'état correspondant.

A. Modélisation d'une machine à état avec Stateflow

Nous allons créer un diagramme d'état élémentaire pour modéliser la logique de la boucle de cap du pilote hydraulique de bateau.
Le système reçoit deux variables d'entrée :

- **mes_cap** : cap réel du bateau mesuré à l'aide d'un compas
- **cons_cap** : consigne de cap du bateau

Le système renvoie une variable de sortie. :

- **cons_barre** : consigne angulaire de la barre du bateau

Le graphe d'état comporte 3 états distincts :

Etat « Pause »
Le système est dans cet état si l'erreur de cap est inférieure à 0.5°. Le pilote hydraulique maintient la consigne de barre à 0° tant que l'erreur de cap ne dépasse pas 1°. En cas de dépassement de ce seuil, le système entre dans l'état « Temporisation ».

Etat « Temporisation »
Le système entre dans cet état si l'erreur de cap a dépassé le seuil de 1°. Si l'erreur de cap se maintient pendant 20s au-dessus de 1°, le système passe dans l'état « correction de cap ». Si durant ces 20s, l'erreur de cap repasse sous le seuil de 0.1°, le système retourne dans l'état « Pause ».

Etat «Correction de cap »
Le pilote hydraulique corrige le cap du bateau, la consigne de barre est obtenue à partir de l'erreur de cap. Le système sort de l'état « Correction de cap » et entre dans l'état « Pause » quand l'erreur de cap redevient inférieure à 0.05°.

B. Construction du diagramme d'état

1. Ouverture du modèle

Ouvrir le modèle ***pilote_hydraulique_stateflow_0.slx***.
Ce modèle contient la modélisation du pilote hydraulique de bateau à l'exception du diagramme d'état qui gère la boucle de cap.

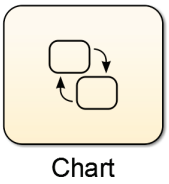
Double cliquer sur le sous-système « chaine d'information » et observer le diagramme à compléter sur la Figure 278.

Figure 278 : boucle de cap sans le diagramme d'état

On observe sur le modèle que la boucle de cap ne comporte pas de logique de commande, nous allons modéliser cette logique de commande à l'aide d'un diagramme d'états.

2. Insertion d'un « chart »

Le composant qui permet de construire un diagramme d'état s'appelle un « chart ». C'est lui qui va contenir les états et les transitions.
Insérer un nouveau « chart » à partir de la bibliothèque **Stateflow**.

Fonction du composant	Représentation	Bibliothèque
Chart		Statflow

Positionner et **redimensionner** le chart pour obtenir la configuration de la Figure 279.

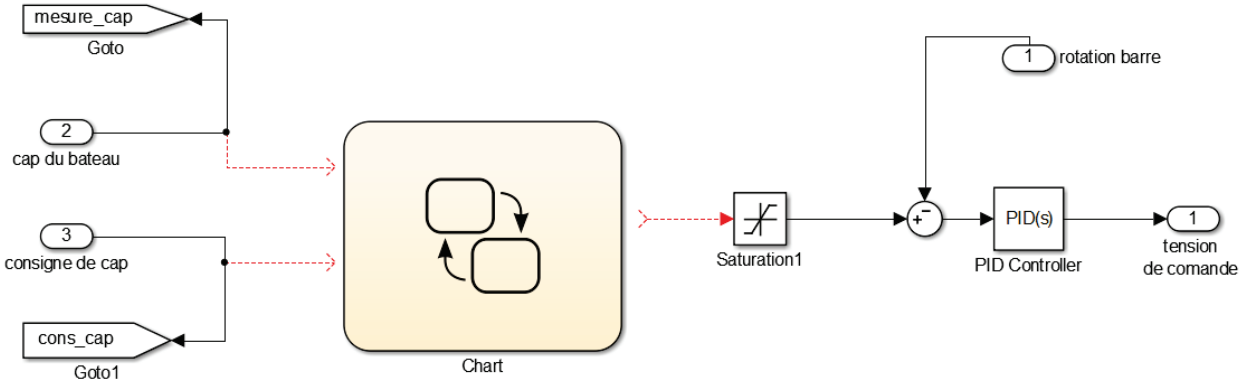


Figure 279 : positionnement d'un chart dans un modèle Simulink

On remarque que pour l'instant le « chart » ne possède aucune entrée/sortie. Elles seront définies ultérieurement.

C. Création d'un diagramme d'état élémentaire

Double cliquer sur le « chart » pour ouvrir l'environnement **Stateflow**.
La barre de commande **Stateflow** apparait à gauche de la fenêtre avec en particulier les commandes qui permettent de créer un état et une « transition par défaut ».

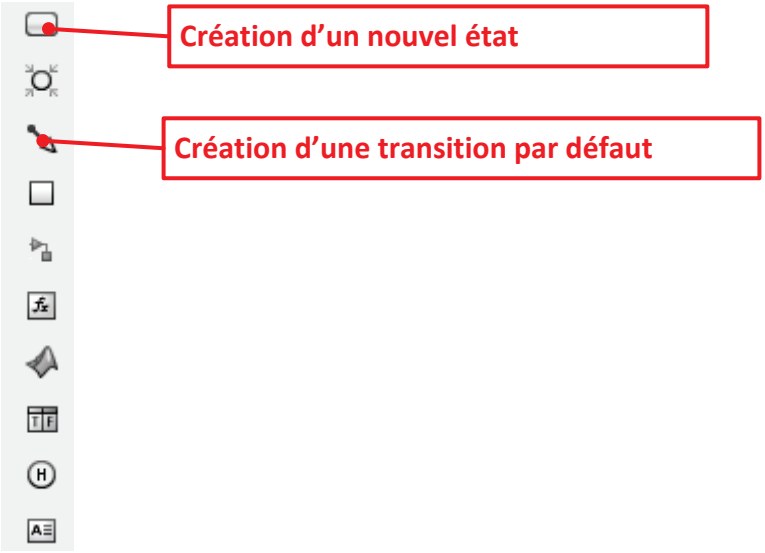


Figure 280 : commande de création d'un état et d'une « transition par défaut »

1. Création des états

Il est maintenant possible de créer les états pour programmer la logique du système.
A l'aide de la commande de création d'un nouvel état, placer et nommer les trois états conformément à la Figure 281. Pour cela **cliquer** avec le bouton gauche de la souris sur la commande de création d'un nouvel état et faire glisser l'état dans la fenêtre de travail.

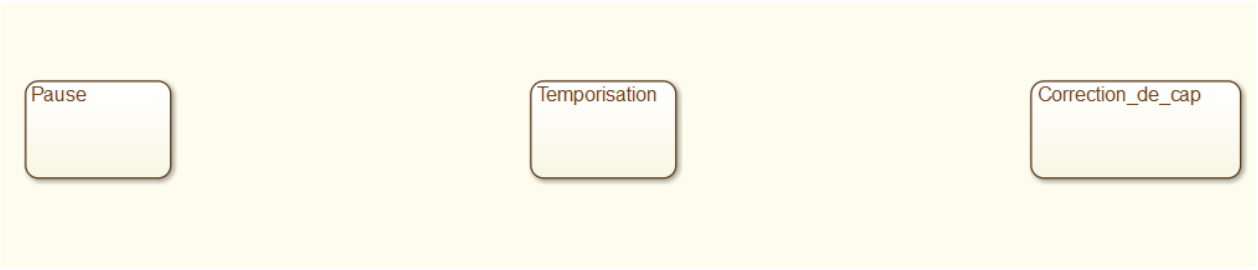


Figure 281 : création des états du système

Dans cet exemple, nous allons commencer par travailler avec des états **exclusifs**, c'est-à-dire qu'à chaque pas de temps, un seul état pourra être actif. Nous verrons par la suite qu'il est également possible de créer des états **parallèles** qui pourront être activés simultanément.

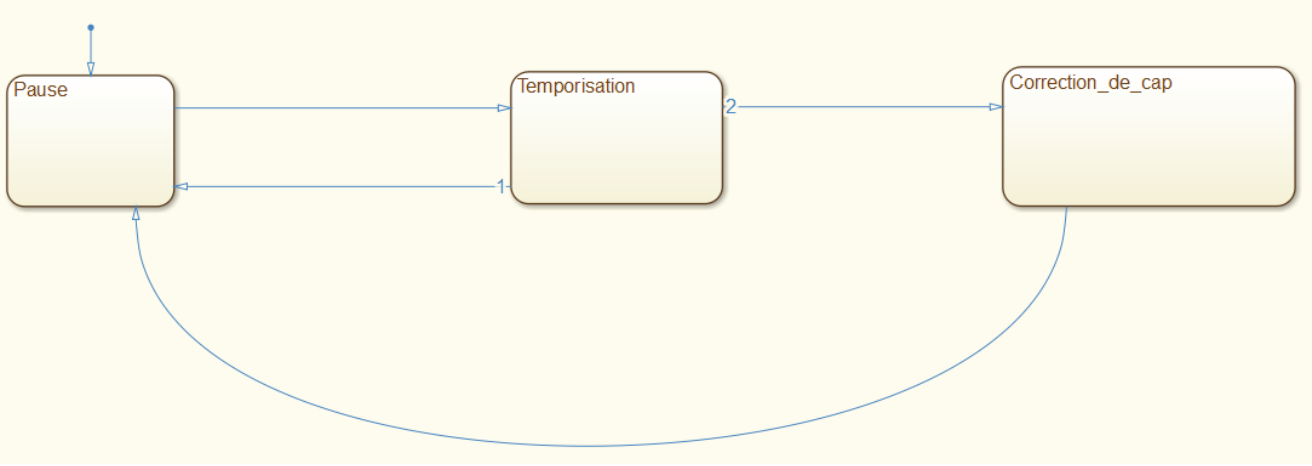
2. Création d'une transition par défaut

Placer une transition par défaut sur l'état « Pause ». La transition par défaut permet d'activer un état en début de simulation. La présence d'une **transition par défaut** dans le diagramme est **indispensable**.

Figure 282 : création d'une transition par défaut

3. Création des transitions

Pour créer une transition, déplacez le curseur de souris sur le bord de l'état de départ de la transition. Lorsqu'il prend la forme d'une croix, cliquer avec le bouton gauche et glisser avec le curseur jusqu'à l'état destination de la transition.
Créer les transitions pour obtenir la configuration de la Figure 283
Les transitions sont munies de poignées invisibles qui permettent de leur donner la forme souhaitée. Pour les déformer, cliquer avec le bouton gauche de la souris sur la transition et utiliser le glisser déposer.



.

Figure 283 : création des transitions entre les états

4. Création des actions dans les états

Il faut maintenant affecter à chacun des états les actions à imposer au système.
Pour cela il faut utiliser un mot clé d'action qui déterminera le moment où l'action sera exécutée.

Les mots clés usuels sont :

- **entry** (ou **en**) : l'action sera exécutée uniquement lors de l'activation de l'état
- **exit** (ou **ex**) : l'action sera exécutée uniquement lors de la désactivation de l'état
- **during** (ou **du**) : l'action sera exécutée à chaque pas de temps tant que l'état est actif.

Compléter les actions des états conformément à la Figure 284.

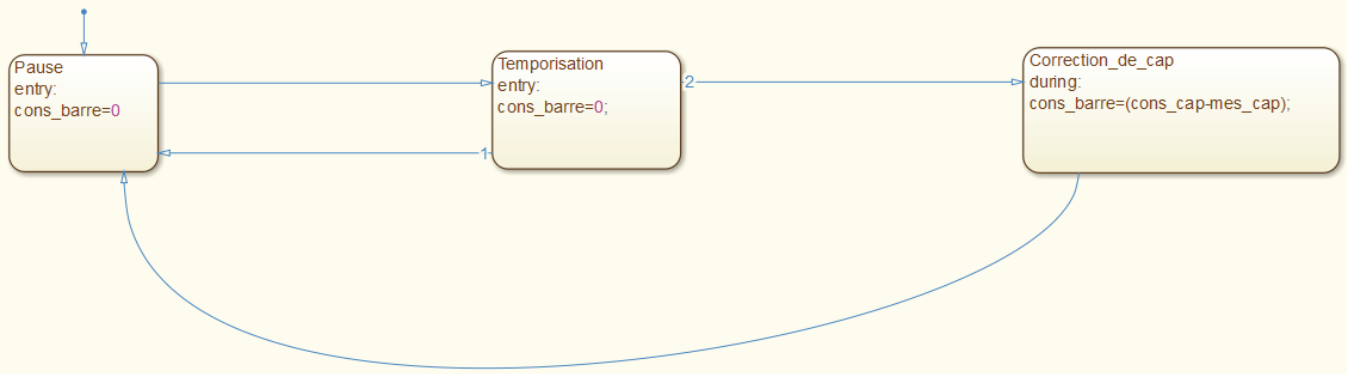


Figure 284 : affectation des actions dans les états

Pour l'état « **Pause** » et « **Temporisation** », l'utilisation du mot clé **entry** impose l'action cons_cap=0, uniquement à l'activation de l'état. La variable cons_barre reste à 0 et ne varie pas quand ces états sont actifs. Par contre lorsque le système est dans l'état « **Correction_de_cap** », la variable cons_barre doit être modifiée continuellement lors de l'évolution du cap du bateau d'où la nécessité d'utiliser le mot clé **during**.

5. Création des étiquettes de transitions

Il faut maintenant définir les conditions qui permettront de passer d'un état à un autre. Les étiquettes de transitions représentent des conditions logiques qui doivent être vérifiées pour permettre le passage d'un état à un autre.
Les informations présentes dans l'étiquette de transition peuvent être de différentes natures.

- Un commentaire : **/*commentaire*/**
- Une condition : **[condition]**

Pour créer une étiquette de transition, cliquer gauche sur la transition, puis cliquer gauche à nouveau sur le point d'interrogation qui apparait. Vous pouvez alors saisir l'étiquette de transition. Pour déplacer une étiquette de transition, utiliser un glisser déposer avec le bouton gauche de la souris.

Créer les étiquettes des transitions conformément à la Figure 285.

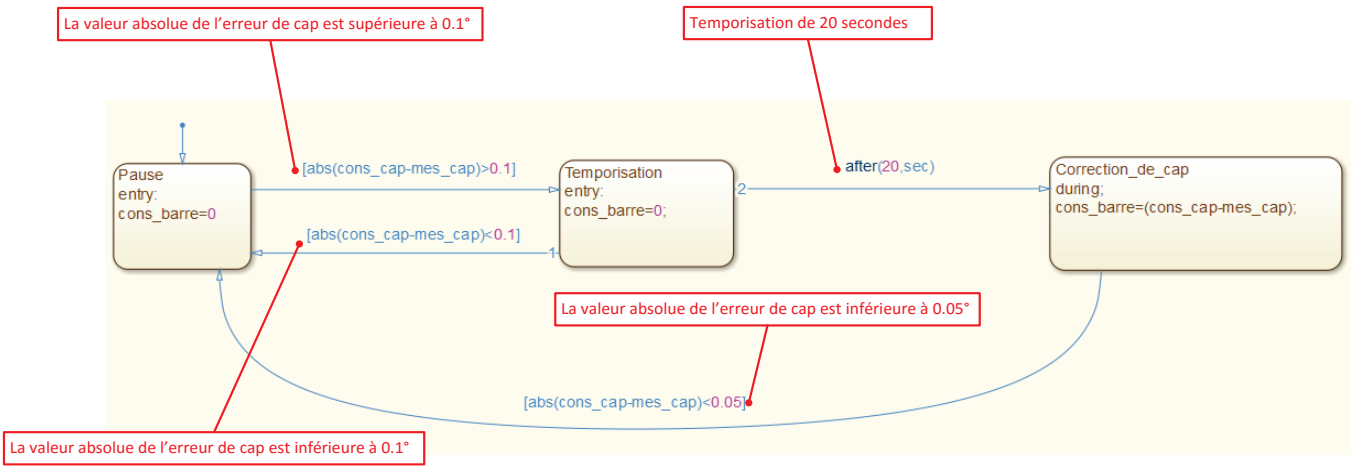


Figure 285 : écriture des étiquettes de transitions dans un diagramme d'états

Les étiquettes de transitions offrent d'autres possibilités qui sont décrites page 266.

6. Définitions des variables d'entrée et de sortie du diagramme d'état

Si nous retournons dans le sous-système représentant la chaîne d'information, nous remarquons que le chart ne dispose pas d'entrée et de sortie pour être connecté avec le système.

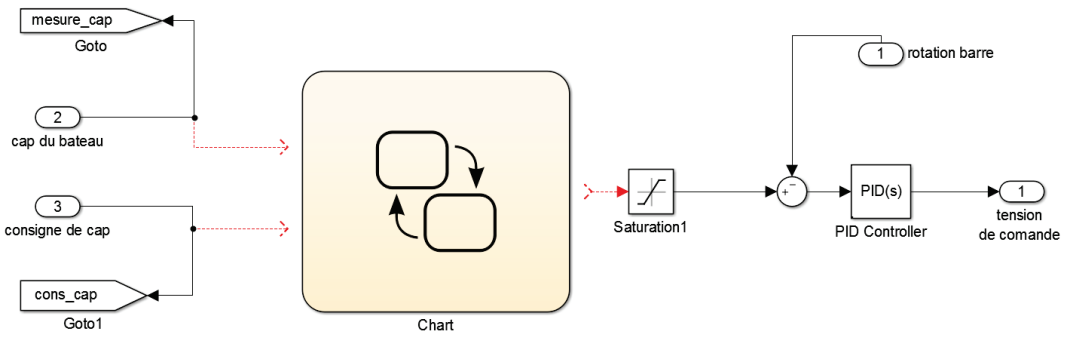


Figure 286 : chart non connecté avec le modèle

Il existe plusieurs moyens de créer les variables d'entrée et de sortie. Le plus simple est de lancer la simulation, **Stateflow** proposera automatiquement une affectation pour les entrées et les sorties du chart en fonction de la logique qui vient d'être programmée.

Lancer la simulation du modèle.
Un message d'erreur apparaît indiquant que la simulation est impossible et la fenêtre du **Symbol Wizard** s'ouvre en proposant une affectation des variables par défaut.
Vérifier que les affectations proposées par défaut correspondent bien à la logique de commande du système. Si nécessaire, il est possible de modifier les affectations à l'aide des menus déroulants.
Cocher la case View created data/events in Model Explorer puis cliquer sur OK.

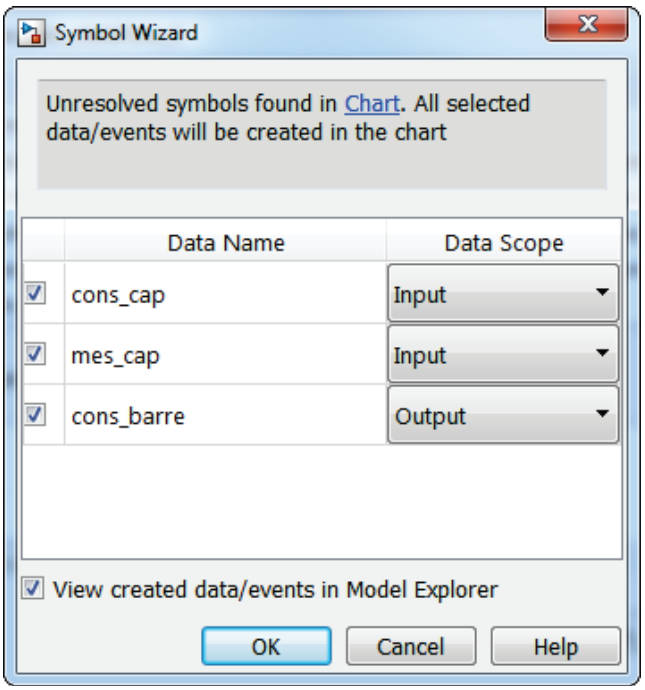


Figure 287 : affectation des variables du diagramme à l'aide du Symbol Wizard

A la fermeture de la fenêtre du **Symbol Wizard**, la fenêtre du **Model Explorer** s'ouvre. Le **Model Explorer** permet de visualiser pour l'ensemble du modèle toutes les variables qui ont été créées. L'arborescence située sur la partie gauche de la fenêtre permet de naviguer dans le système.

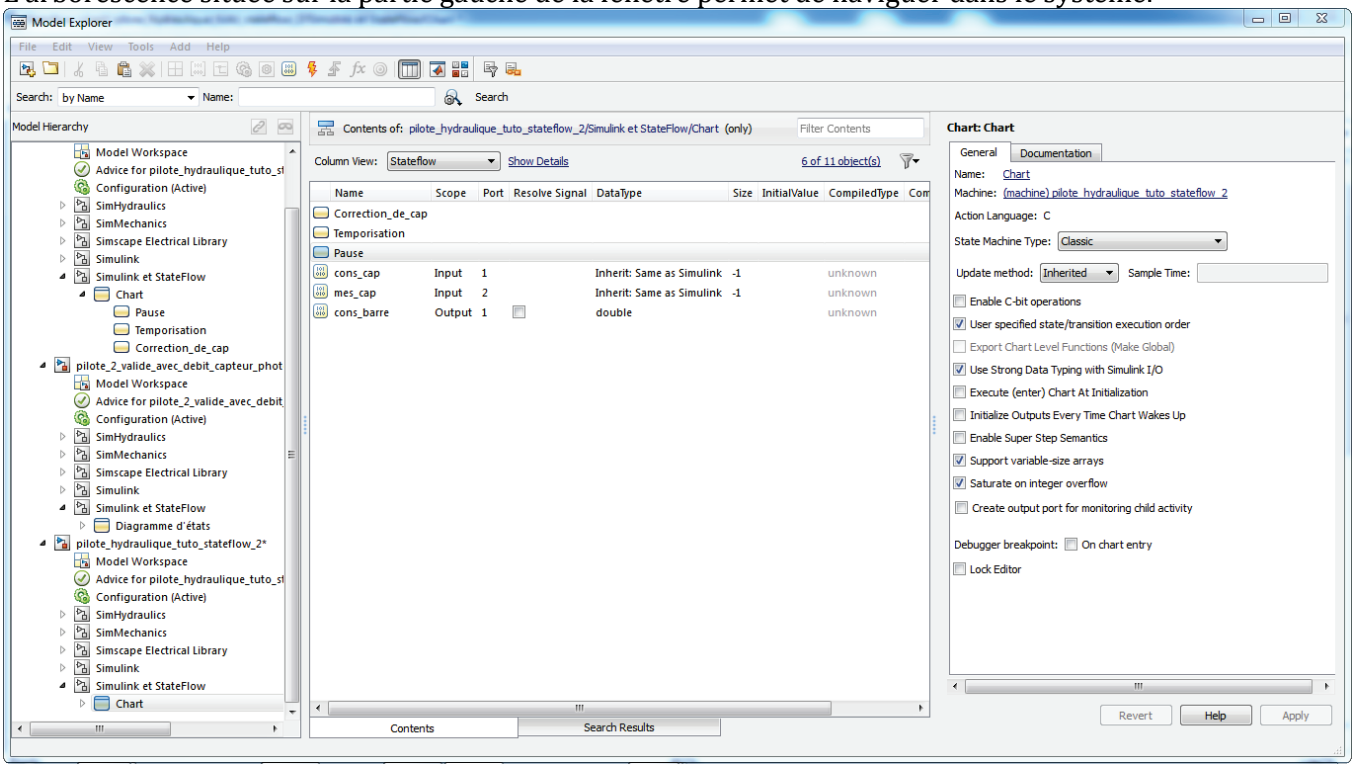


Figure 288 : visualisation des variables dans le Model Explorer

Il est possible de modifier les caractéristiques des différentes variables. Vous pouvez revenir à tout moment dans le **Model Explorer** pour modifier les caractéristiques d'une variable en cliquant sur  dans la barre de commande principale.

Il est également possible d'ajouter des variables manuellement à partir de la barre de commande en utilisant la commande **Chart/Add Inputs & Outputs/Data Inputs from Simulink** pour ajouter une **entrée** et la commande **Chart/Add Inputs & Outputs/Data Outputs from Simulink** pour ajouter une **sortie**.

Revenir dans le sous-système représentant la chaîne d'information pour visualiser les connexions qui apparaissent sur le Chart.

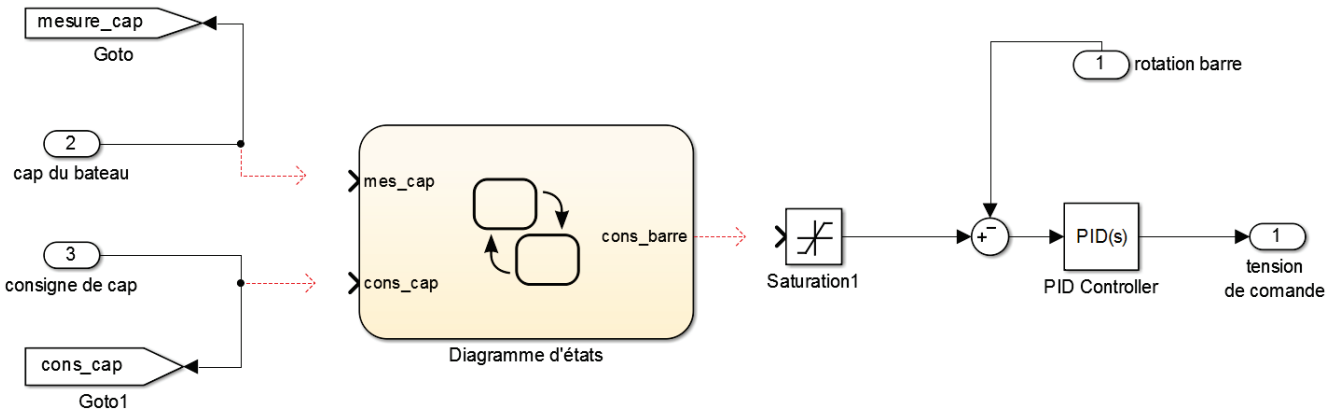


Figure 289 : chart avant connexion avec le système

Les connexions étant créées automatiquement, elles peuvent se positionner différemment sur le chart. Vérifiez que la consigne de cap sera bien reliée à cons_cap et que la mesure de cap sera bien reliée à mes_cap.
Si les connexions sont inversées, ouvrir le **Model Explorer** et indiquer les bons numéros de port comme indiqué sur la Figure 290.

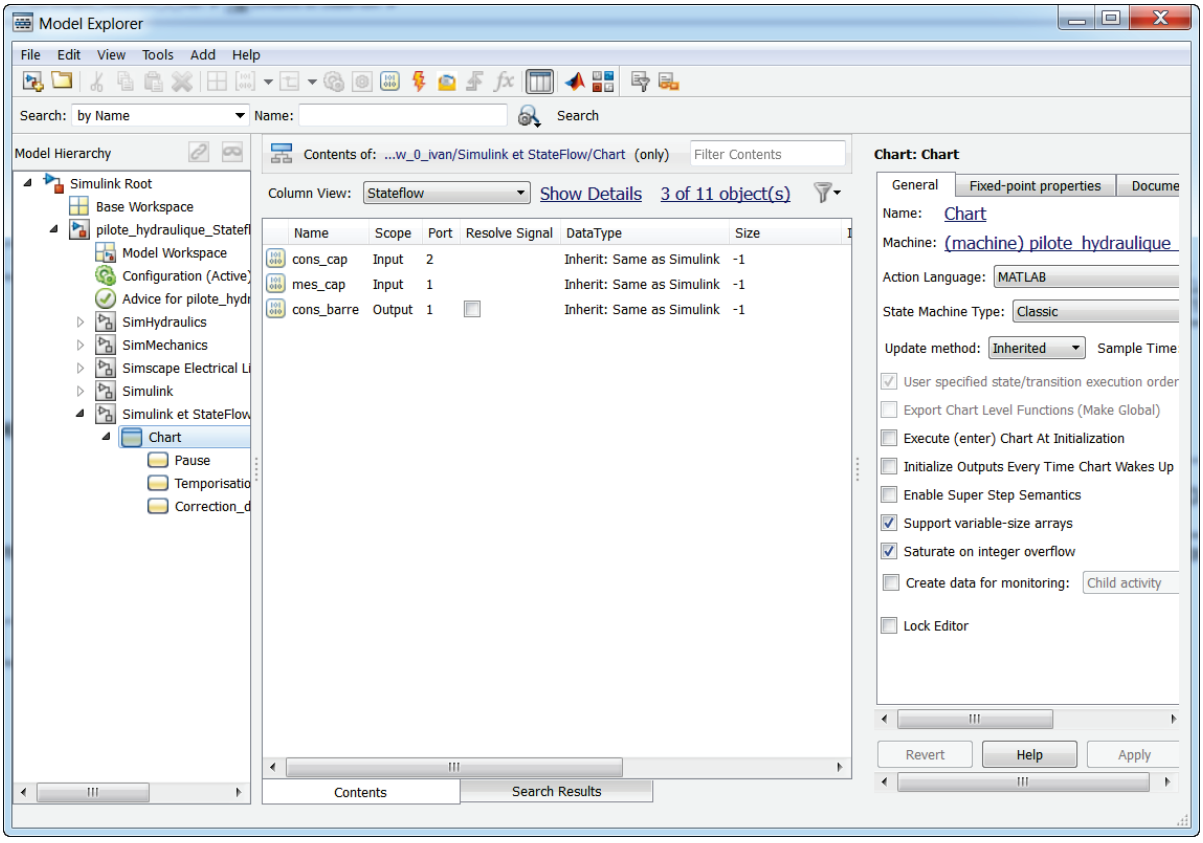


Figure 290 : modification des numéros de port d'un chart

Connecter le chart avec le schéma bloc Simulink conformément à la Figure 291.

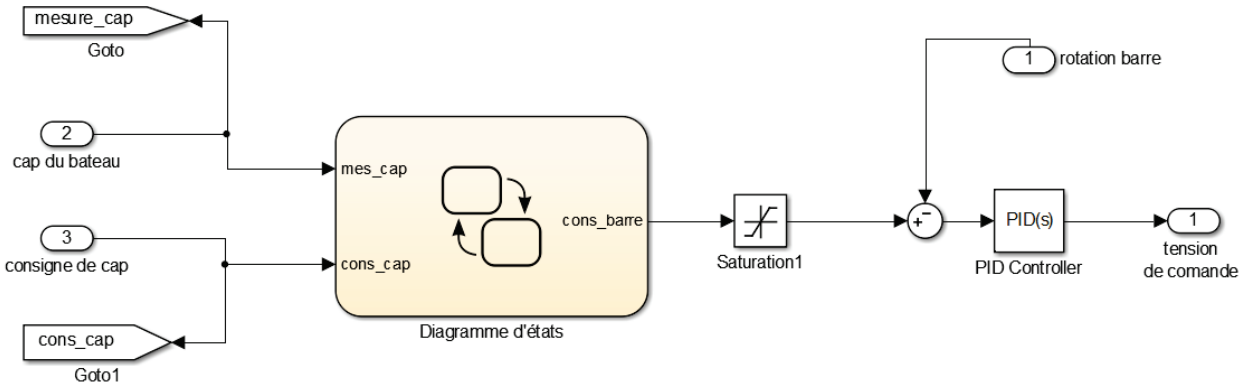


Figure 291 : connexion du chart avec le schéma bloc Simulink

7. Simulation du diagramme d'états

Lancer la simulation et observer le suivi du cap en visualisant le scope donnant la consigne de cap et le cap effectif suivi par le bateau.

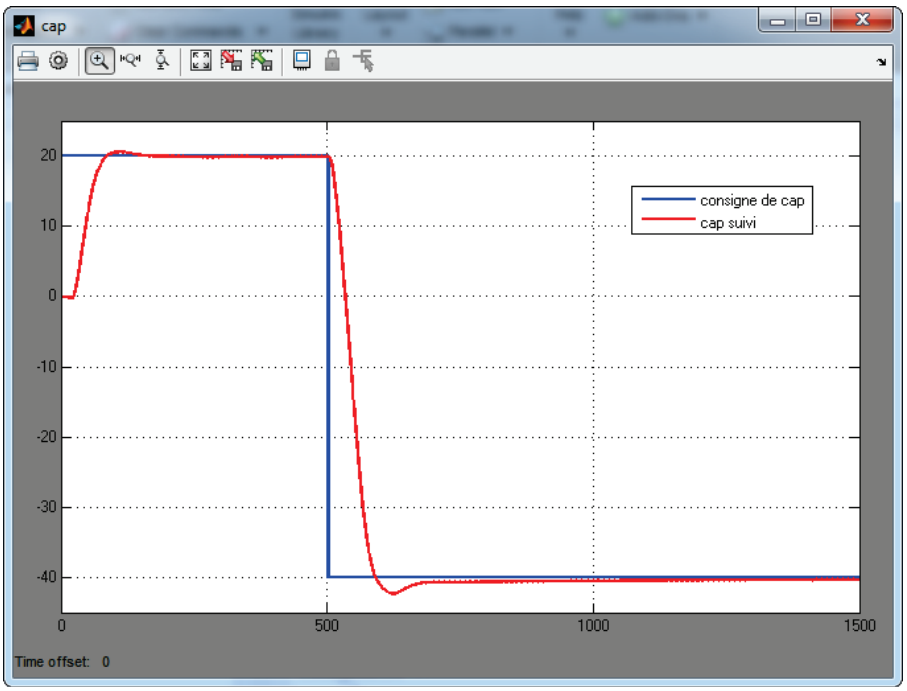


Figure 292 : résultat de la simulation

D. Architecture des machines à états

1. La hiérarchie des états

Le diagramme d'état précédemment établi comporte 3 états de même niveau. Aucune hiérarchie n'est présente entre ces états. Afin de pouvoir modéliser des comportements logiques plus complexe **Stateflow** utilise le concept de **super-état**.
Un super état est un état qui peut contenir d'autres états de niveaux hiérarchiques inférieurs appelés **sous-états**.

- **Super-état** : état parent contenant d'autres états
- **Sous-états** : état enfant contenu dans un état parent. Les sous-états ne peuvent être actifs uniquement si l'état parent est actif.

L'utilisation de super-états et de sous-états permet d'améliorer la lisibilité des modèles. Il n'y a pas de limitation dans le nombre de niveaux hiérarchiques que l'on peut construire dans un diagramme.

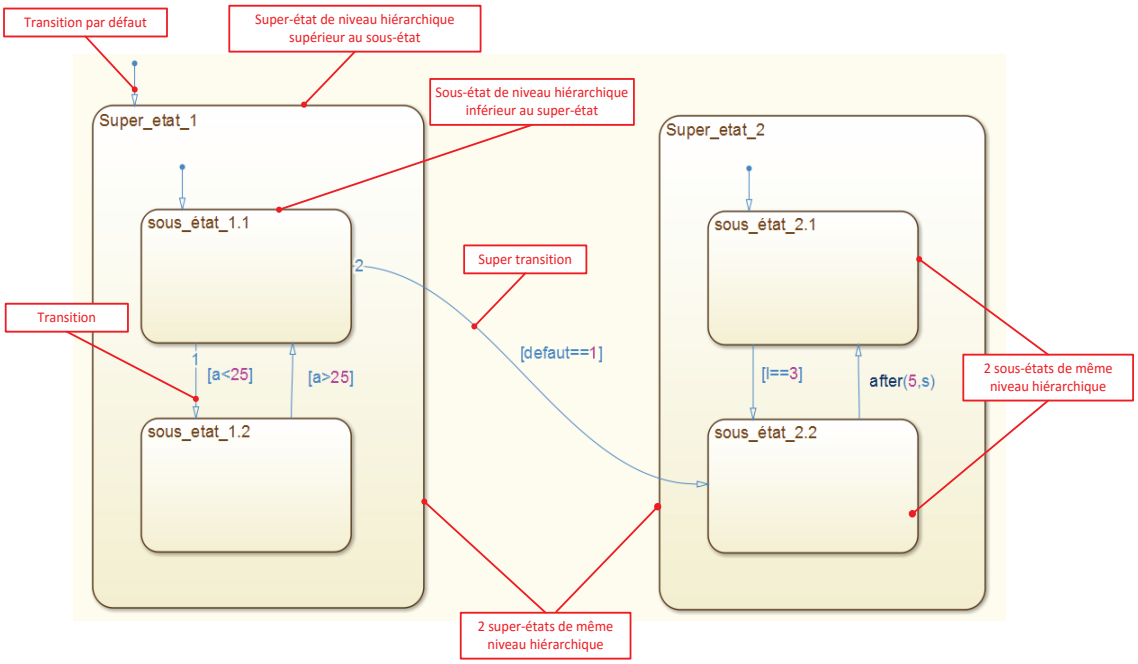


Figure 293 : représentation de super-état et de sous-état

2. Les priorités de test des transitions

Ces architectures plus complexes nécessite des précisions sur les règles d'évolution et sur la priorité de test des différentes transitions.

On définit :

- les **transitions internes** : une transition qui ne traverse pas les bordures de l'état ou qui est englobée par l'état
- les **transitions externes** : une transition qui traverse les frontières de l'état

Règles de test des transitions:

- les transitions des super-états sont testées avant celles des sous-états
- les transitions externes sont testées avant les transitions internes

3. Etats parallèles

A chaque niveau de la hiérarchie, les sous-états peuvent être **parallèles** ou **exclusifs.**

- **Si le sous-état est exclusif** : un seul et unique sous-état ne peut être actif au sein d'un même niveau hiérarchique (à condition que le super-état parent soit actif)
- **Si l'état est parallèle** : tous les sous-états d'un même niveau hiérarchique sont actifs simultanément (à condition que le super-état parent soit actif). Il n'est pas possible de définir une transition vers ou depuis un état parallèle, puisque son activation sera conditionnée exclusivement par l'activation de son état parent.

Dans **Stateflow**, les états parallèles s'affichent avec des bords en pointillés.

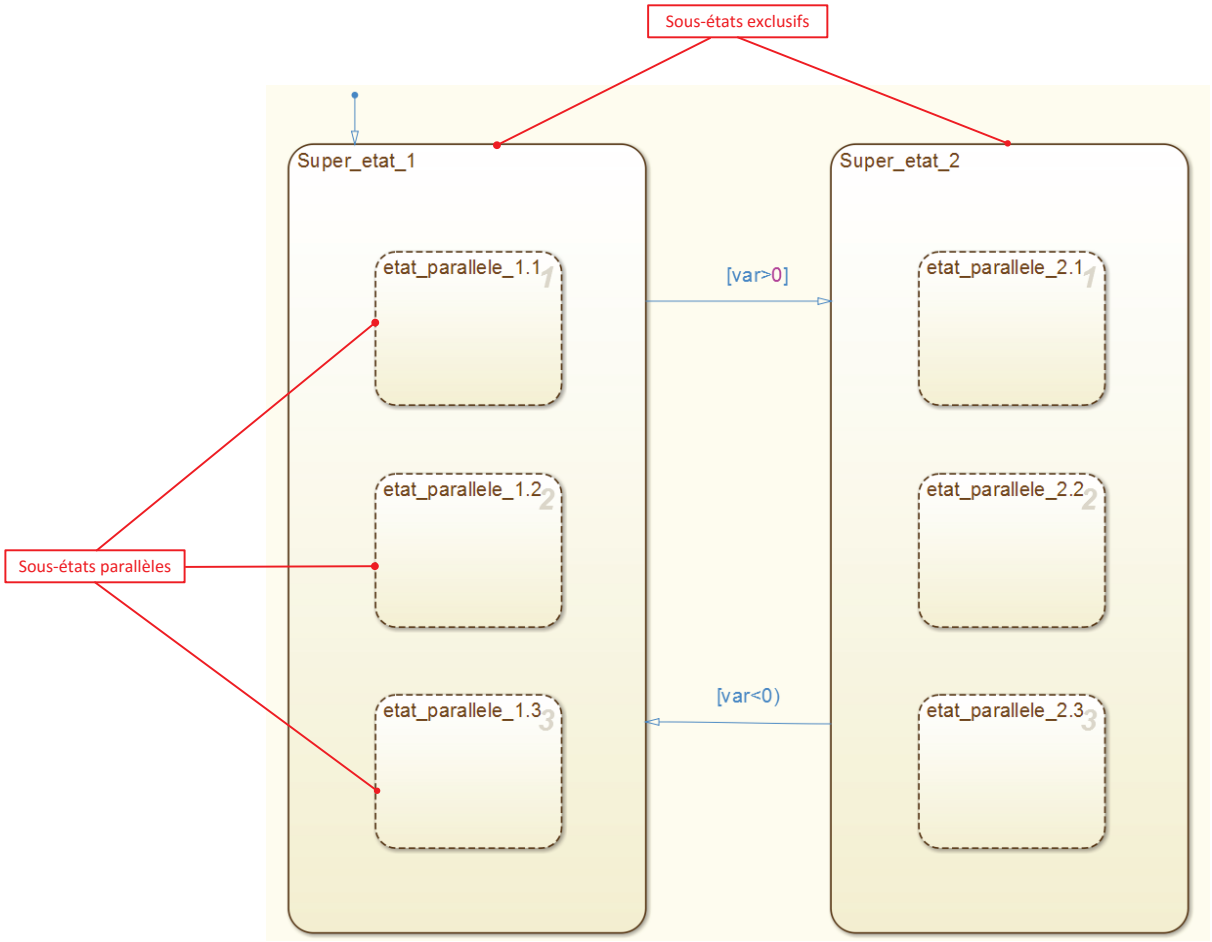


Figure 294 : représentation de sous-états parallèles

E. Ajout de niveaux hiérarchique et d'états parallèles dans un diagramme d'état

Nous allons reprendre le diagramme d'état de la commande de cap du pilote hydraulique de bateau en ajoutant de nouvelles fonctionnalités.
Lorsque le pilote est en mode « correction_de_cap », un voyant rouge doit s'allumer pour signaler aux utilisateurs que le pilote automatique consomme de l'énergie.
L'allumage de ce voyant sera géré dans l'état « Gestion_voyant ». Une variable « voyant_rouge » sera à 1 lorsque le voyant sera allumé et à 0 lorsque le voyant sera éteint.

Créer les super-états « **Suivi_de_cap** », « **Gestion_voyant** » et « **Commande_de_cap** » en respectant la hiérarchie de la Figure 295. La procédure de création d'un super-état est la même que la procédure de création d'un état. Il faut être attentif à positionner les contours de l'état parent autour des états enfants. **StateFlow** prend en compte automatiquement la hiérarchie entre les états à partir de leur positionnement dans la fenêtre graphique. Si un conflit empêche **Stateflow** d'établir une hiérarchie sans ambiguïté entre les états, les états qui posent un problème apparaissent en rouge.

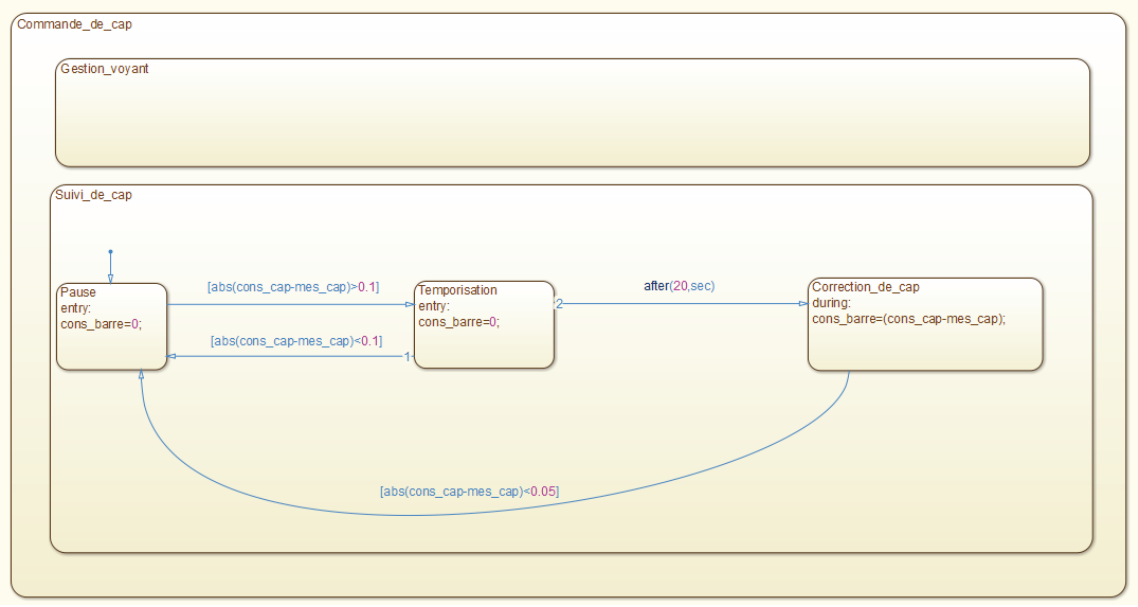


Figure 295 : création de super-états

L'état « **commande_de _cap** » possède 2 états enfants :

- « **Gestion_voyant** »
- « **Suivi_cap** »

Ces états enfants doivent être des états **parallèles** puisque l'allumage du voyant doit se faire pendant que l'état « **Suivi_de_cap »** est actif. Le bon fonctionnement du système de commande impose que les deux états enfants soient actifs simultanément lorque l'état « **commande_de_cap** » est activé.

L'état « **Suivi_de_cap** » possède 3 états enfants :

- « **Pause** »
- « **Temporisation** «
- « **Correction_de_cap »**

Ces états enfants doivent être des états **exclusifs** puisque ces états doivent être activés successivement lorsque l'état parent « **Suivi_de_cap »** est activé. Un seul et unique de ces états enfants sera actif.

Pour indiquer à **Stateflow** que les états enfants de l'état parent « **Commande_de_cap** » seront des états parallèles, **cliquer** avec le bouton droit de la souris à l'intérieur de l'état « **Commande_de_cap** », mais à l'extérieur des états enfants (Figure 296). Dans le menu contextuel choisir **Decomposition/parallel**.

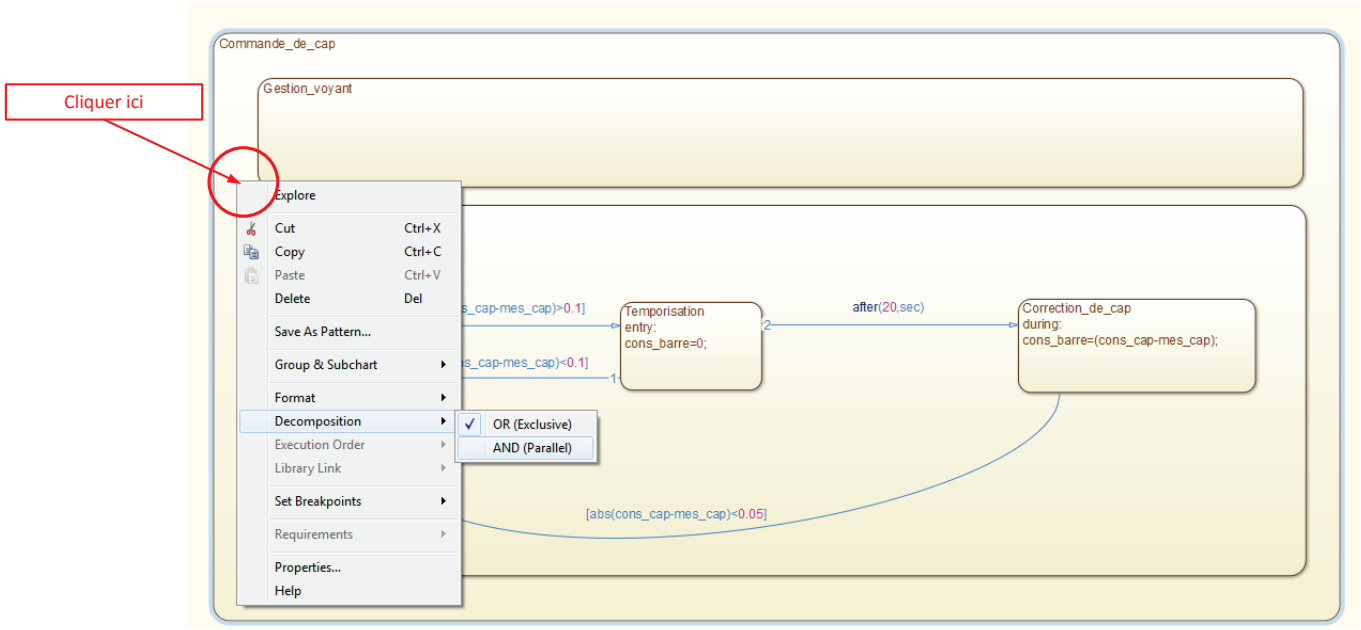


Figure 296 : créer des sous-états parallèles

Les états parallèles apparaissent maintenant en pointillés.

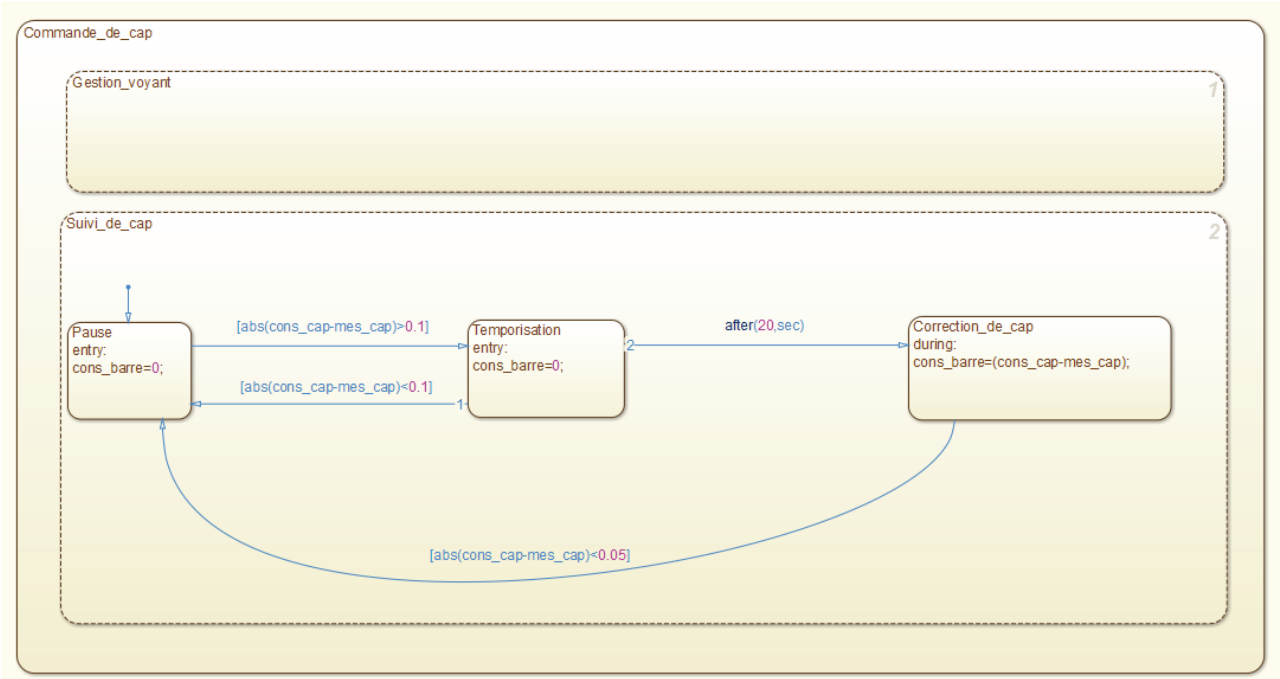


Figure 297 : représentation des états parallèles dans Stateflow

Il faut maintenant indiquer que la variable « voyant_rouge » sera à 1 lorsque l'état « **Correction de cap »** sera actif.
Pour cela **Stateflow** dispose de variables internes qui caractérisent l'activation des états.
La commande *in(nom_de_l_etat)* renvoie la valeur 1 quand l'état est actif et la valeur 0 quand l'état est inactif.

Le nom d'un état est constitué de toute l'arborescence hiérarchique des états parents.

Compléter l'état « **Gestion_voyant** » comme indiqué sur la Figure 298.

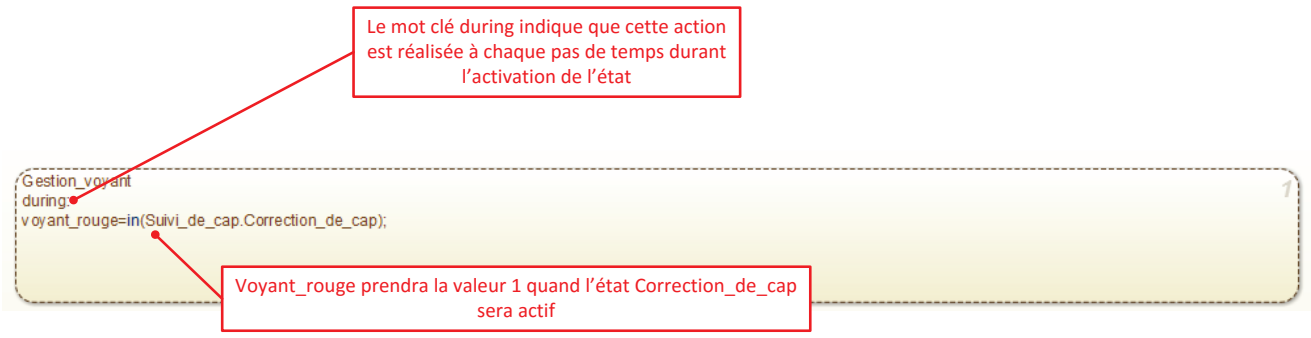


Figure 298 : utilisation de variables internes pour évaluer l'activation d'un état

En utilisant la commande **Chart/Add Inputs & Outputs/Data Outputs to Simulink**, créer la variable de sortie **voyant_rouge**.

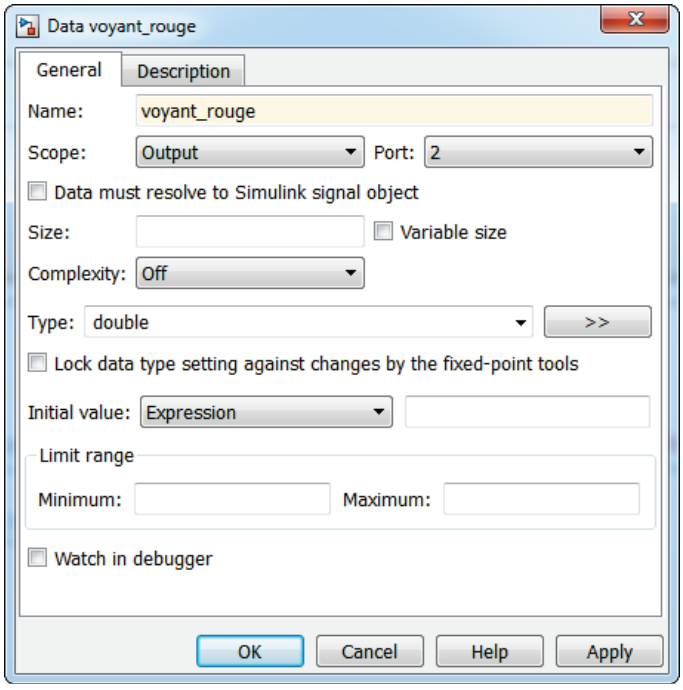


Figure 299 : création d'une variable de sortie vers Simulink

Avant de lancer la simulation, il faut ajouter une « transition par défaut » sur l'état « **Commande_de_cap** » afin d'activer le diagramme au démarrage.

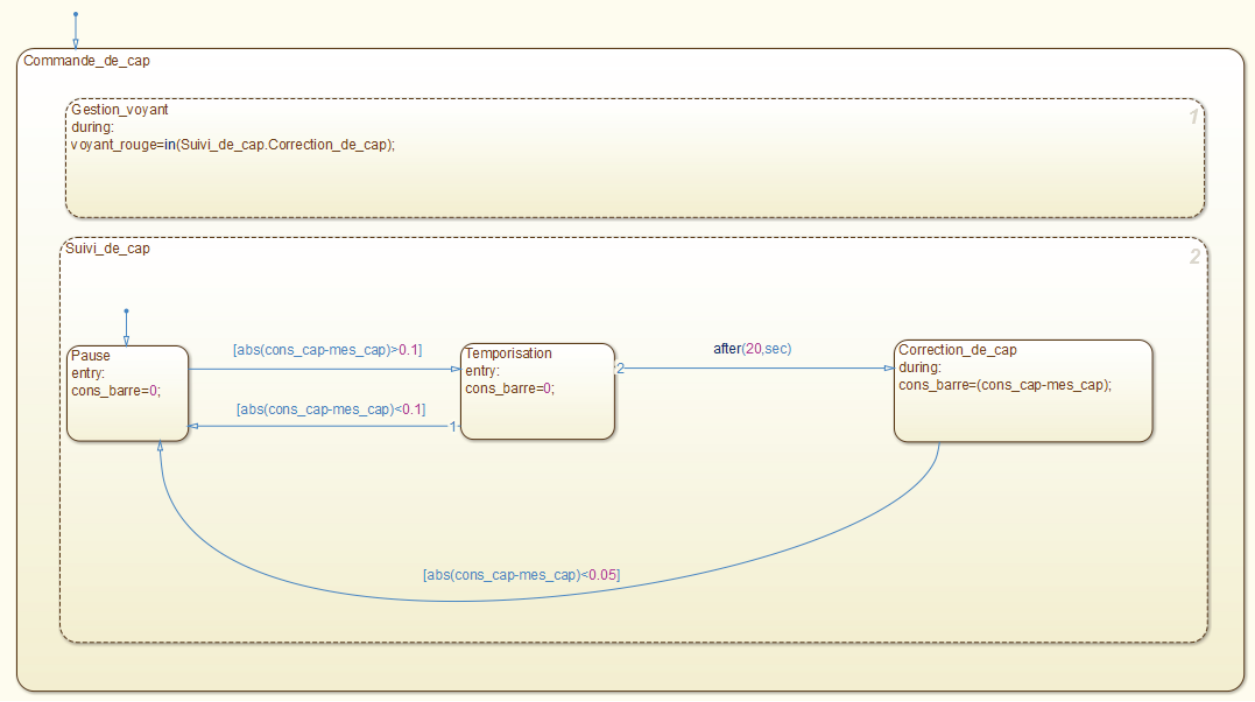


Figure 300 : chart complet de commande de cap avec super-états et états parallèles

Connecter la variable de sortie voyant_rouge à un scope pour la visualiser.

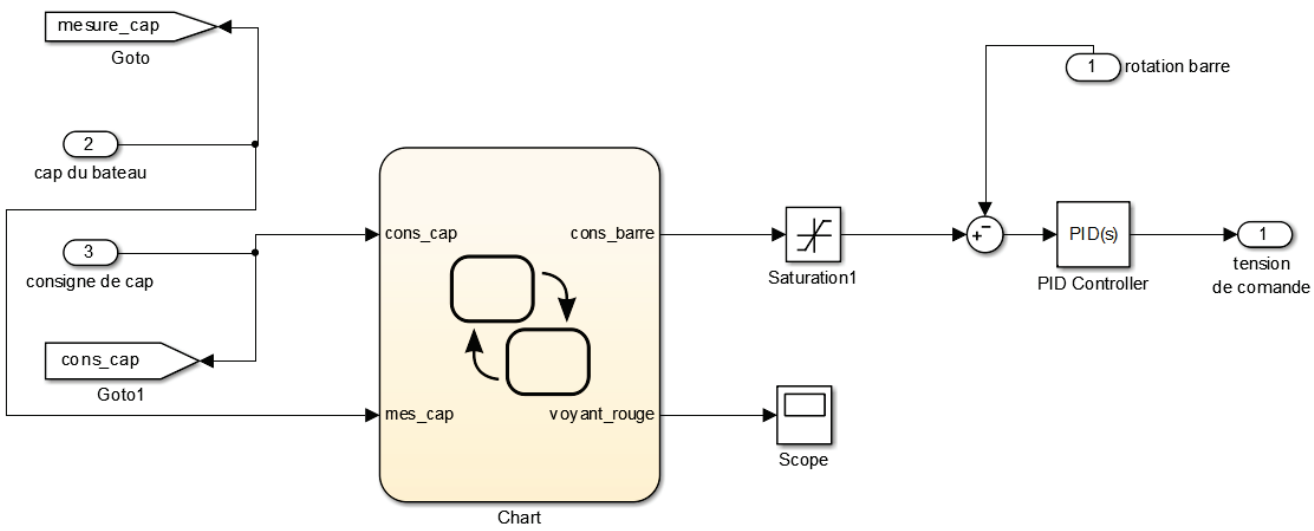


Figure 301 : connexion de la variable de sortie du chart à un scope

Si nécessaire le modèle complet est disponible dans le fichier ***pilote_hydraulique_Stateflow_1.slx.***

Lancer la simulation et observer l'évolution de l'état de la variable voyant rouge dans le scope.

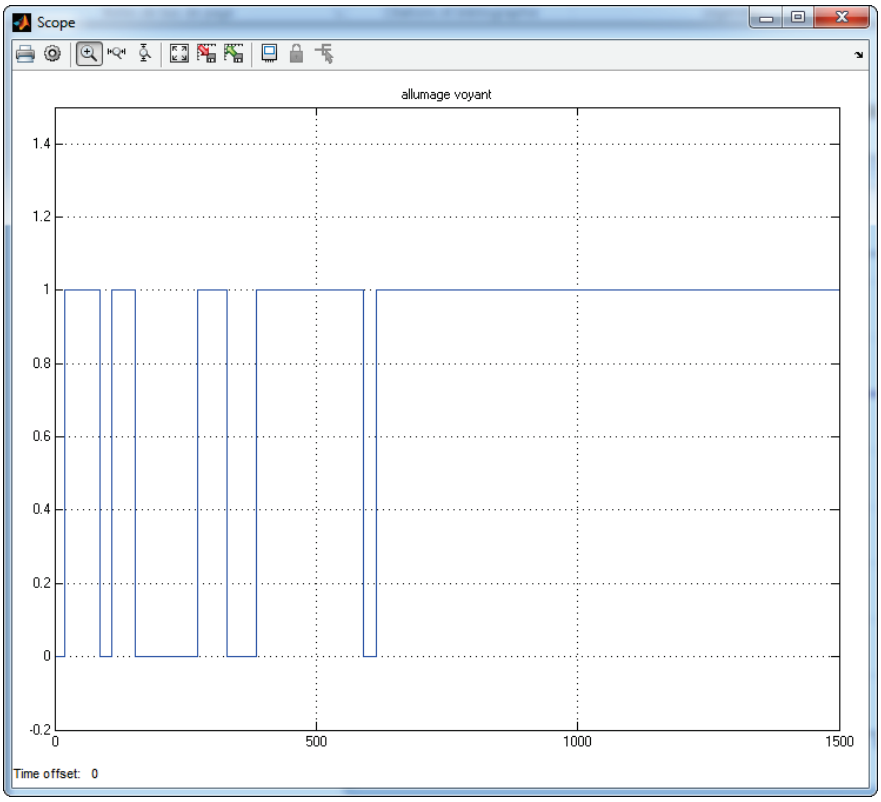


Figure 302 : visualisation de l'état d'activation du voyant

F. Récapitulatif et complément des commandes utiles de Stateflow

Commandes utiles

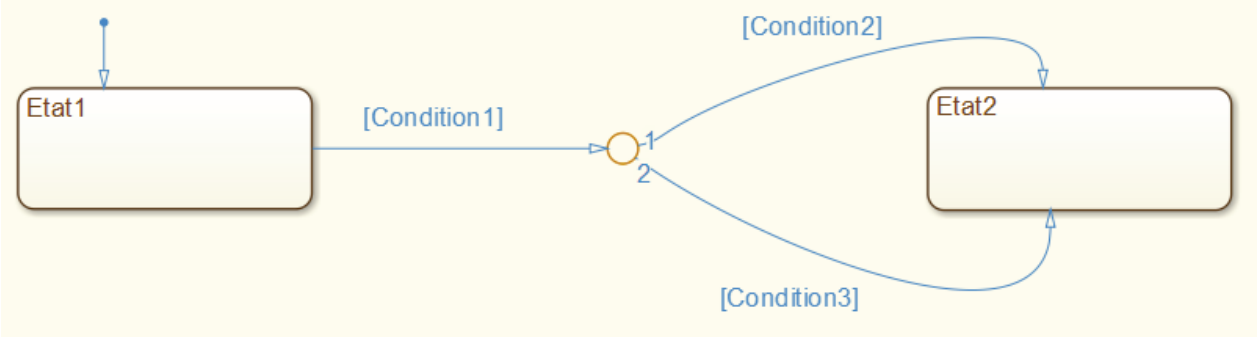
Fonctions	Commandes/syntaxe

Création d'un état :

Les mots-clés :

en ou **entry** : l'action s'effectue uniquement à l'activation de l'état
du ou **during** : l'action s'effectue en continu tant que l'état est activé
ex ou **exit** : l'action s'effectue uniquement à la désactivation de l'état

Création d'une transition avec jonction:



Pour créer une transition avec chemin multiples, il est possible d'utiliser une jonction  disponible dans le menu « Chart ».
Pour passer à l'Etat2, il faut que **[Condition1]** soit vrai et que soit **[Condition2]** ou **[Condition3]** soit vrai. Le chemin numéroté **1** est évalué avant le chemin numéroté **2**.

Création d'une transition incluant des actions:

La **Condition** permet de valider le passage de l'Etat1 à l'Etat2.

Pour passer à l'Etat2, il faut que **[Condition]** soit vrai et que soit **[Condition1]** ou **[Condition2]** soit vrai.

L'**Action de condition** est une action qui s'exécute uniquement si le test de la condition est vrai, et même si le chemin de la transition n'aboutit pas à l'activation de l'Etat2.
Si **[condition]** est vrai alors **{action de condition}** s'exécute même si **[Condition1]** et **[Condition2]** sont faux**.**

L'**Action de transition** est une action qui s'exécute uniquement si le chemin de la transition aboutit à l'activation de l'Etat2.
/Action de condition s'exécute lors du passage de l'Etat1 à l'Etat2.

Test sur les variables dans les transitions :	a et b	a&&b
	a ou b	a||b
	complément de a	!a
	a égal b	a==b

Suivi de l'activation des états :

in(Etat) : cette variable interne est vrai (valeur 1) tant que l'état est actif.

Temporisation de 15 s dans une transition	**after**(15,*s*)

Chapitre 6 : Prise en main de SimMechanics

I. Introduction à SimMechanics

SimMechanics est un outil de MATLAB permettant d'inclure dans un modèle multi-physique, la maquette CAO 3D du système en l'important depuis Solidworks (ou tout autre logiciel de CAO). Cela permet de visualiser les mouvements des solides durant la simulation, de tenir compte du comportement dynamique des solides en mouvements, de prendre en compte toutes les non-linéarités géométriques de manière implicite…
Les résultats issus du modèle multi-physique sont plus proches des résultats observés sur le système réel par la prise en compte de ces nouveaux paramètres sans avoir à écrire les équations caractérisant le comportement dynamique du système.
Il existe deux générations de l'outil SimMechanics, 1G et 2G. Ce document portera sur l'utilisation de SimMechanics seconde Génération (2G) dont l'utilisation a été grandement amélioré par rapport à SimMechanics 1G.

A. Analyse d'un modèle SimMechanics 2G

Dans SimMechanics les systèmes sont représentés sous la forme d'un graphe de liaison. Les solides sont donc représentés par des blocs reliés entre eux par des liaisons.

Ouvrir le fichier « ***pilote_SimMechanics_2G_0.slx*** »
Ce fichier a été importé directement depuis le logiciel Solidworks. Les procédures d'importation seront abordées ultérieurement.

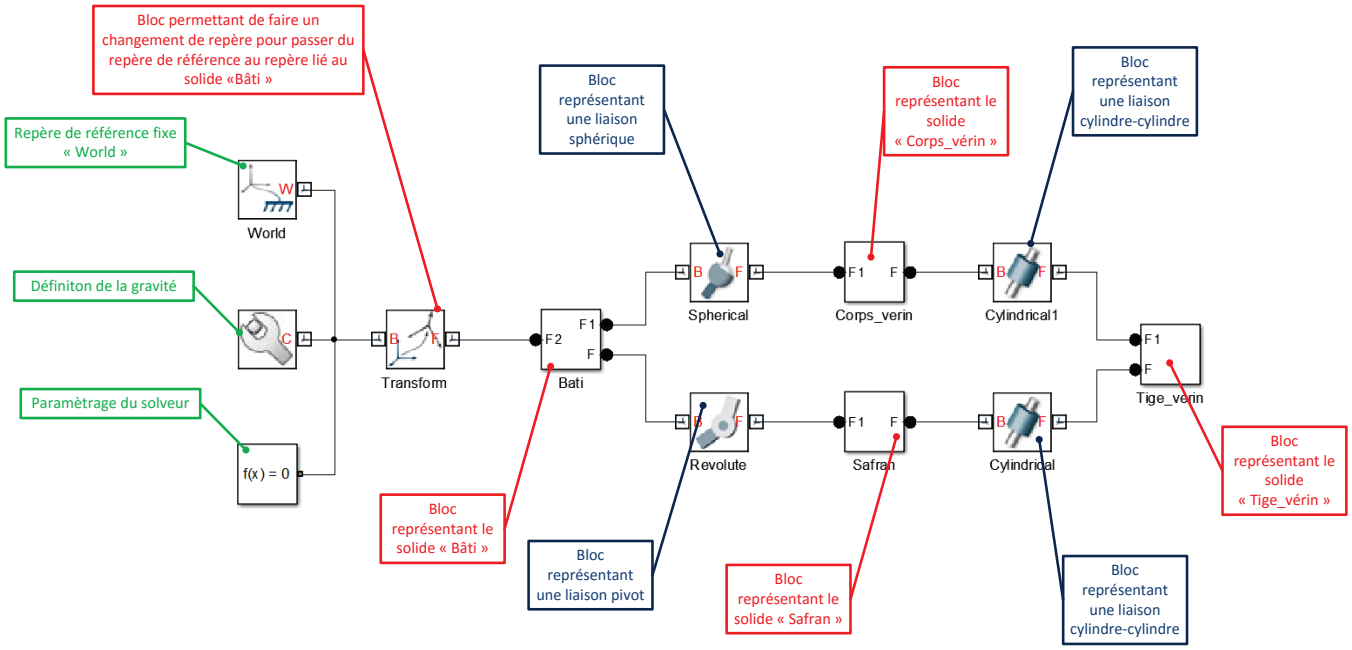


Figure 303 : modèle SimMechanics 2G de la partie mécanique du pilote hydraulique

Pour une meilleure lisibilité du modèle, il est conseillé de créer des masques sur les blocs « solide » afin de visualiser directement les formes.

Ouvrir le fichier « ***pilote_SimMechanics_2G_0_mask.slx*** »

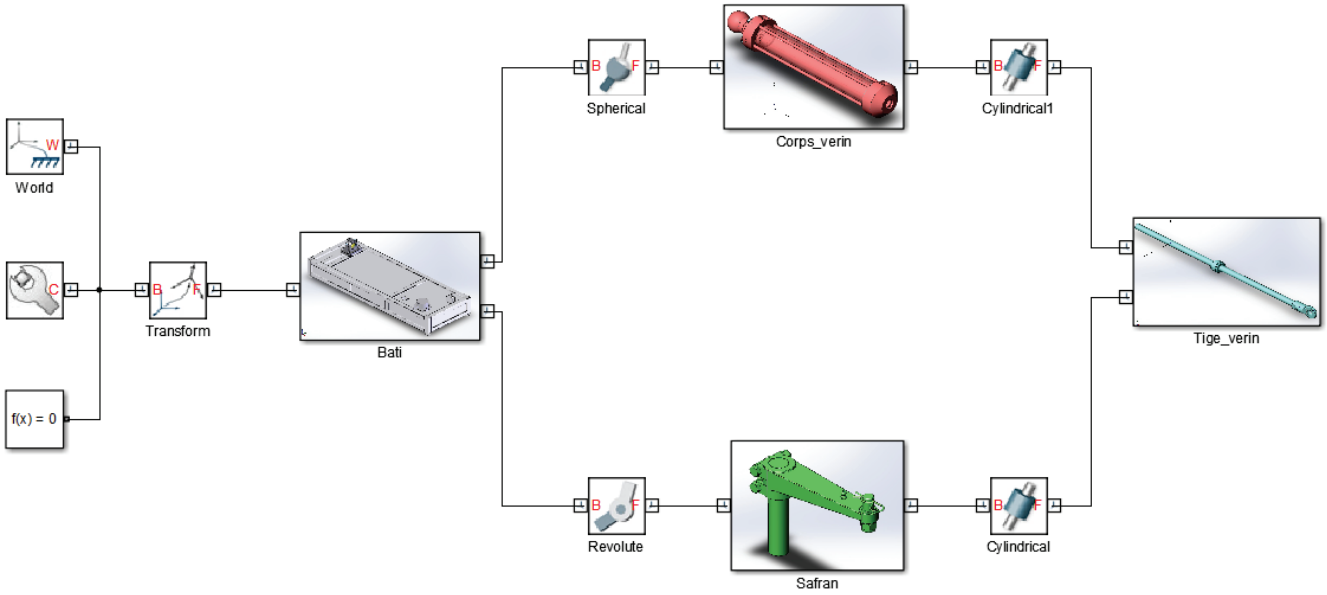


Figure 304 : modèle SImMechanics 2G du pilote avec masques sur les solides

Lancer la simulation du modèle.

La fenêtre **Mechanics Explorer** apparaît et la maquette 3D de la partie mécanique du pilote hydraulique est visualisable.

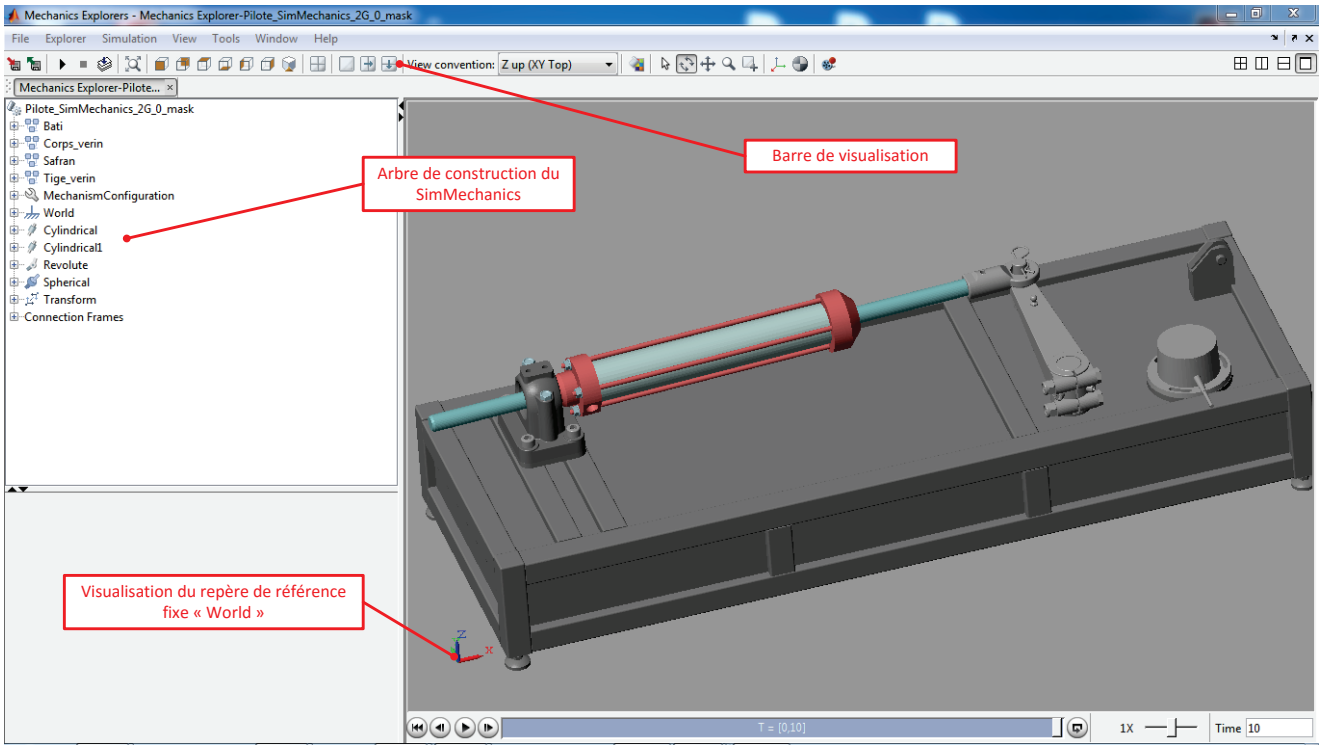


Figure 305 : fenêtre Mechanics Explorer de la partie mécanique du pilote hydraulique

La barre d'outils de SimMechanics permet différentes options d'affichage.

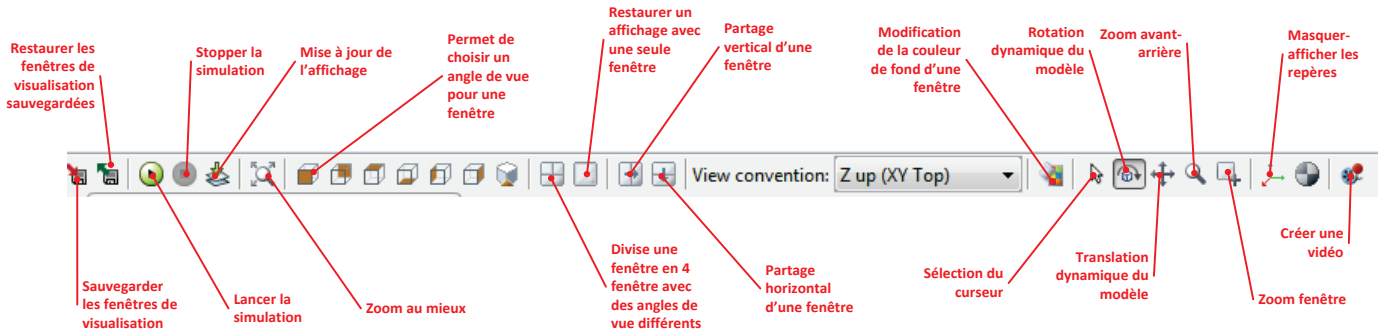


Figure 306 : options de la barre de visualisation de SimMechanics

A noter qu'aucun mouvement n'est observable dans la mesure où aucune liaison n'est pilotée.

B. Paramétrage de la gravité

Retourner dans la fenêtre du modèle et **ouvrir** le bloc Mechanism Configuration 

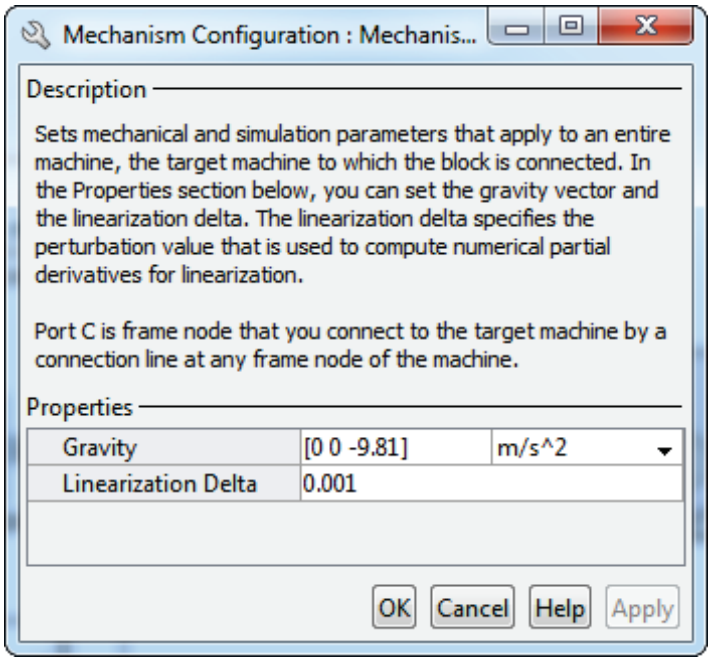


Figure 307 : paramétrage de la pesanteur dans SimMechanics

Par défaut la pesanteur a été placée sur l'axe z (valeur -9.81 m/s²), ce qui correspond à la réalité de fonctionnement du système. Le repère utilisé pour orienter la pesanteur est le repère fixe « World ».

Pour voir l'influence de la pesanteur sur la simulation, nous allons placer la pesanteur sur l'axe x.

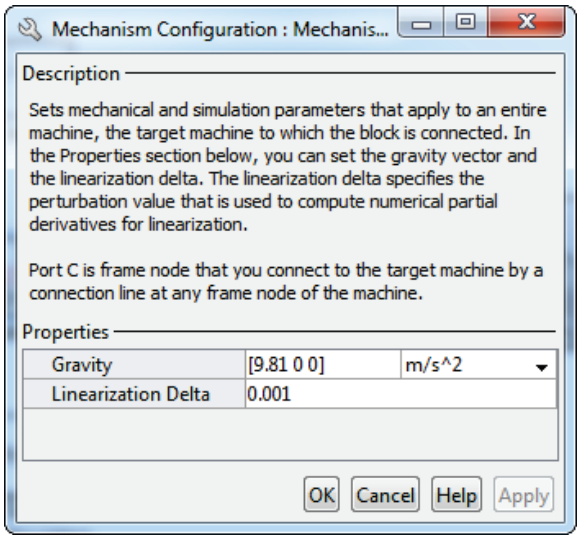


Figure 308 : modification de l'action de la pesanteur

Lancer à nouveau la simulation et observer le mouvement du système. On observe un mouvement de rentrée et sortie de la tige du vérin qui correspond à l'action imposée par la pesanteur. Aucun frottement n'étant modélisé, le mouvement est périodique.

Remettre la pesanteur sur l'axe z pour retrouver les conditions réelles de fonctionnement.

II. Intégration d'un modèle SimMechanics dans un modèle multi-physique

Nous allons apprendre dans cette partie à intégrer un modèle SimMechanics dans un modèle multi-physique. Pour cela nous utiliserons le modèle du pilote hydraulique de bateau.

Ouvrir le fichier ***pilote_hydraulique_SimMechanics_0.slx***

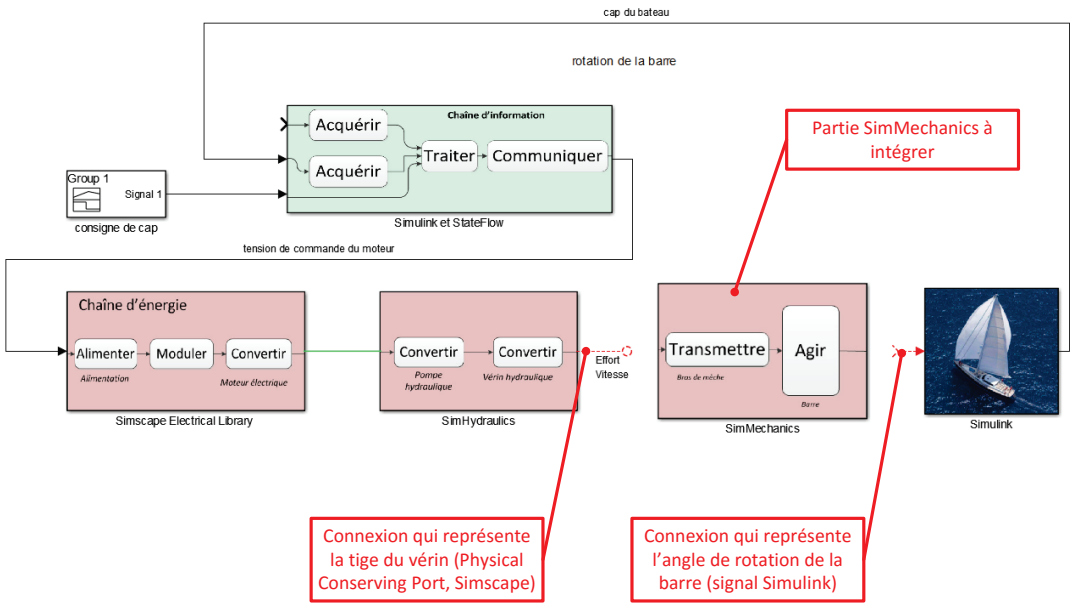


Figure 309 : modèle du pilote hydraulique avec modèle SimMechanics à intégrer

Le sous-système **SimMechanics** contient le modèle de la partie mécanique du pilote, mais qui n'est pas connecté avec le reste du modèle.

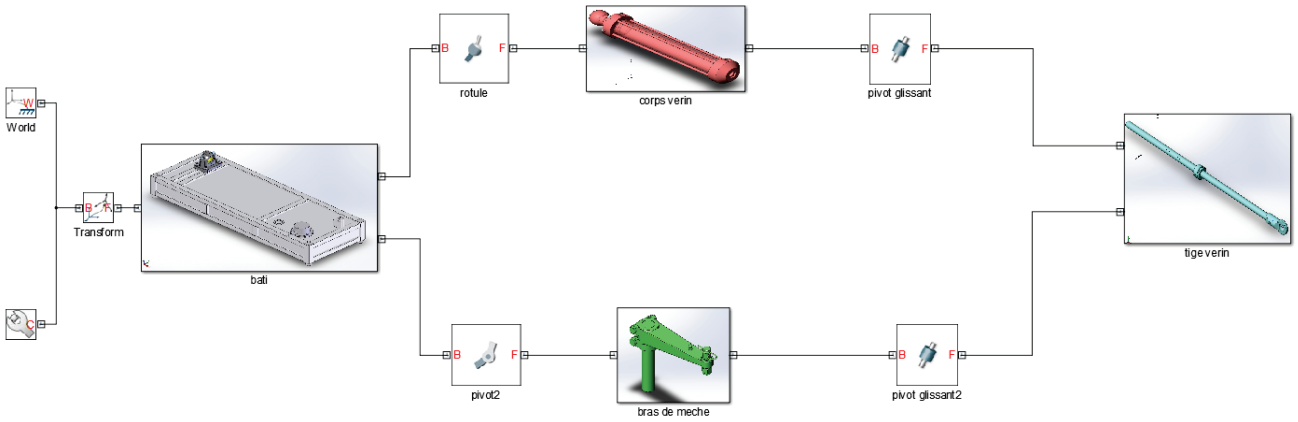


Figure 310 : visualisation du système SimMechanics sans connexion avec le reste du modèle

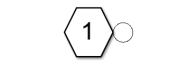
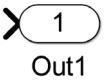
Sur la Figure 309, il est possible de visualiser les connexions que le sous-système **SimMechanics** doit assurer avec le reste du modèle.

- A l'entrée, le modèle reçoit une connexion physique qui représente la tige du vérin. Il s'agit d'un port de type **Physical Conserving Port** (Simscape)
- A la sortie le sous-système **SimMechanics** doit renvoyer l'angle de la barre afin d'agir sur le bateau et de donner l'angle de barre comme information d'entrée à la chaîne d'information.

A. Connexions du modèle

Afin de connecter le modèle, il faut créer à l'intérieur du sous-système **SimMechanics** deux ports :

- Un port Simscape pour se connecter à la tige du vérin
- Un port Simulink pour renvoyer l'ange de rotation de la barre

Désignation	Représentation	Bibliothèque
Port de connexions Simscape		Simscape/Utilities
Port de connexions Simulink		Simulink/Ports & Subsystems

Placer les deux ports et renommer les conformément à la Figure 311.

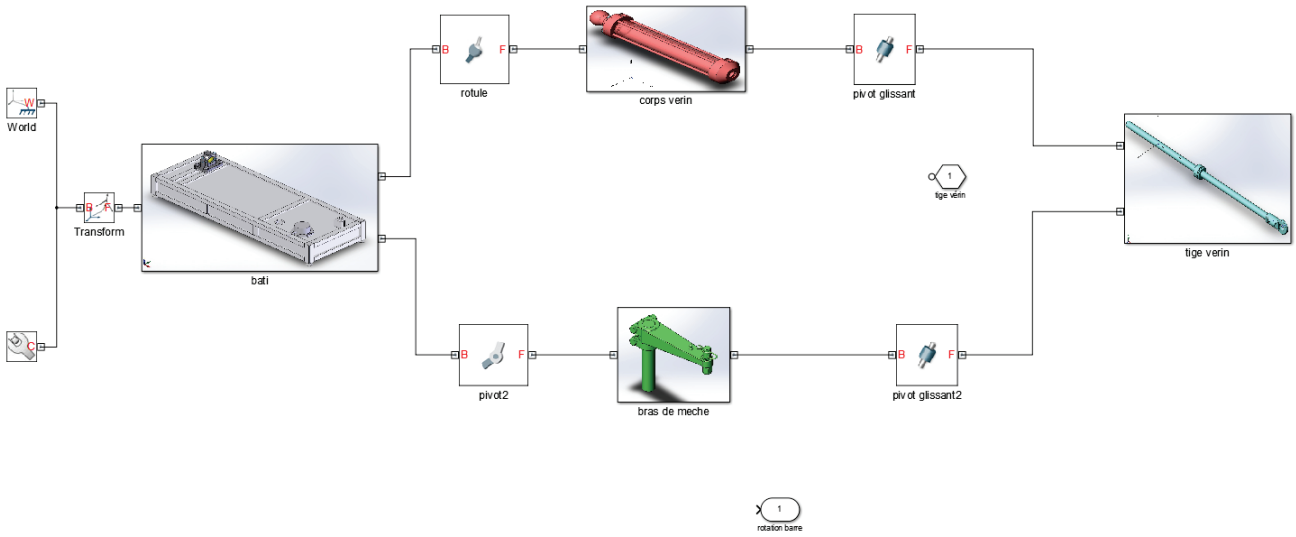


Figure 311 : placement des ports de connexion dans le modèle

Des ports apparaissent sur le sous-système SimMechanics. **Connecter** ces ports avec le modèle.

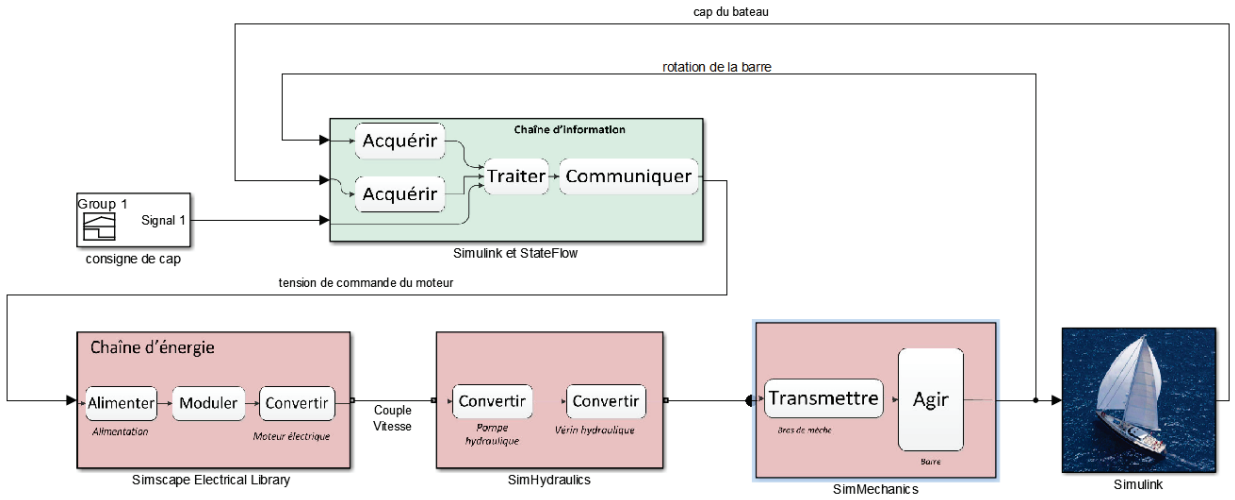


Figure 312 : connexions des ports du sous-système avec le modèle

B. Interfaçage entre Simscape et SimMechanics

Dans cet exemple la connexion « tige vérin » provenant de **Simscape** doit servir à actionner la liaison entre le corps du vérin et la tige. Dans **SimMechanics 2G**, les liaisons sont actionnées par des efforts de type « Force » ou de type « Torque » (moment). Il faut donc actionner la liaison pivot-glissant entre le corps et la tige du vérin avec une force qu'il faudra prélever sur la tige du vérin à l'aide d'un capteur de force. Cette force va agir sur la partie mécanique du système. Il résultera un mouvement de tout le mécanisme qui prendra en compte la dynamique de la partie mécanique ce qui imposera une vitesse de sortie à la tige du vérin. Afin de ne pas casser la boucle physique et de permettre à la tige du vérin modélisée dans **Simscape** de se déplacer à la même vitesse que la tige du vérin du modèle

SimMechanics, il est impératif de réinjecter cette vitesse dans le modèle **Simscape**. Cela garantira durant tout le fonctionnement une équivalence de comportement entre **Simscape** et **SimMechanics**.

1. Interfaçage entre Simscape et SimMechanics pour la translation

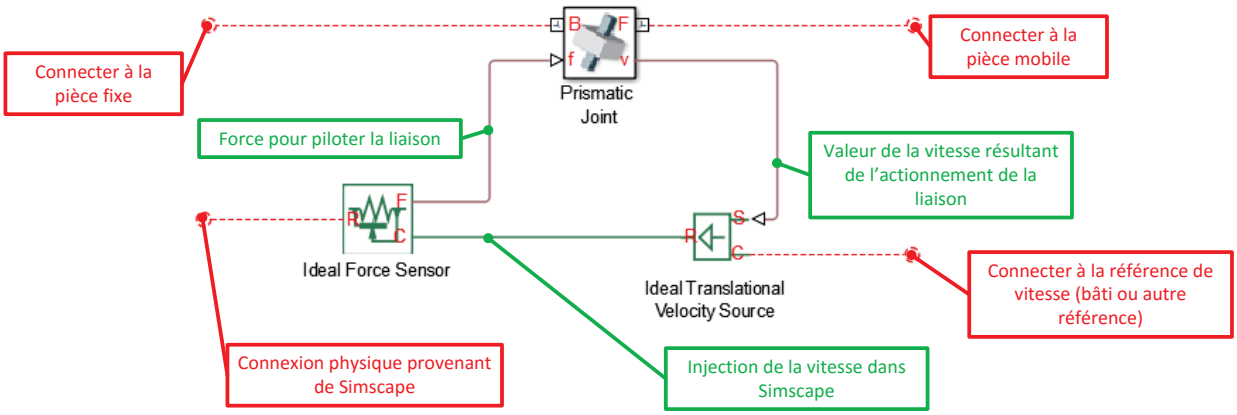


Figure 313 : interface de translation entre Simscape et SimMechanics

2. Interfaçage entre Simscape et SimMechanics pour la rotation

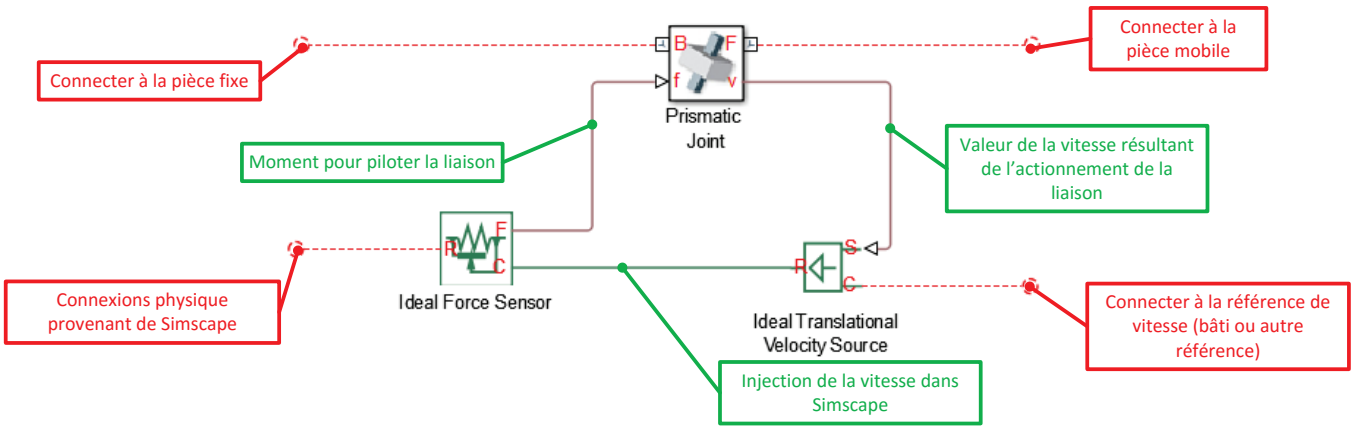


Figure 314 : interfaçage de rotation entre Simscape et SimMechanics

Ouvrir le fichier « ***Simscape_SimMechanics_Interfaces.slx* ».**

Ce fichier contient les interfaces entre **Simscape** et **SimMechanics**. Il est également possible de mettre ces interfaces sous la forme de sous-systèmes afin de les réutiliser facilement.

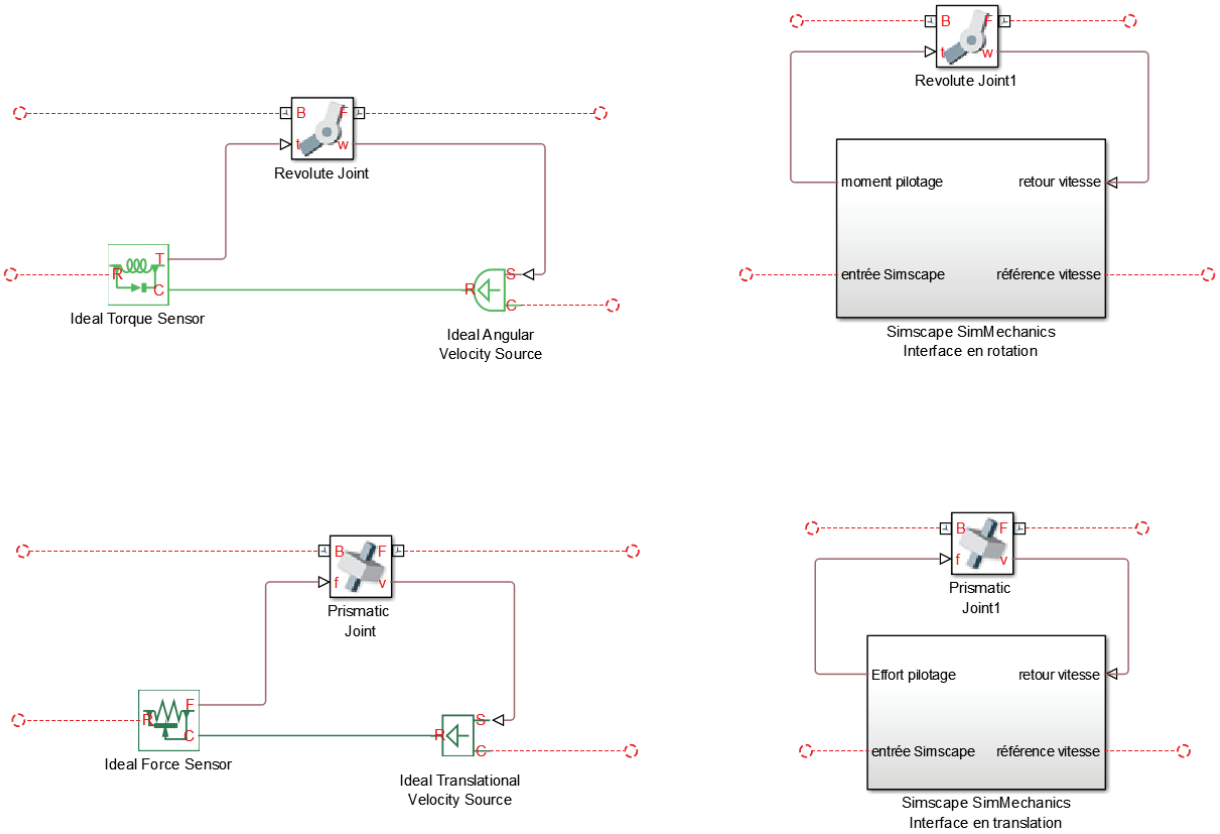


Figure 315 : les interfaces entre Simscape et SimMechanics

Copier le bloc **Simscape SimMechanics Interface** en translation dans le modèle.

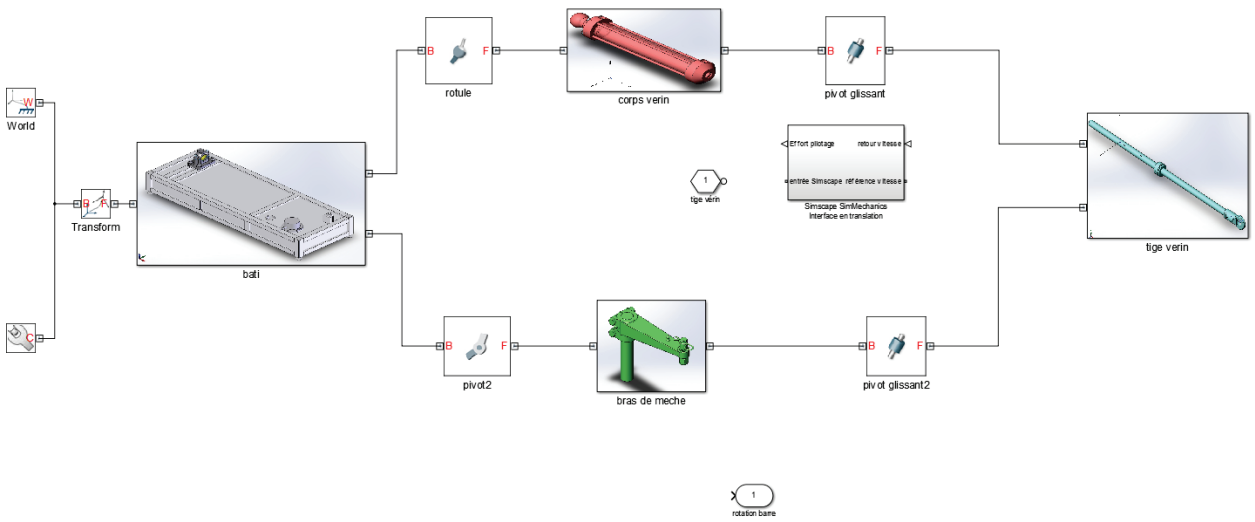


Figure 316 : ajout du bloc Simscape SimMechanics Interface

3. Ajout de ports sur une liaison

Il faut maintenant ajouter des ports de connexions sur le bloc liaison

- Un port afin de lui permettre d'être actionnée
- Un port afin de renvoyer la valeur de la vitesse

Double cliquer sur le bloc de la liaison pivot glissant pour faire apparaître la boite de dialogue du paramétrage d'une liaison et paramétrer la liaison conformément à la Figure 317 afin de créer un port pour actionner la liaison et un port pour relever la vitesse. Paramétrer également un coefficient de frottement visqueux (Damping Coefficient)de 200 N.s/m.

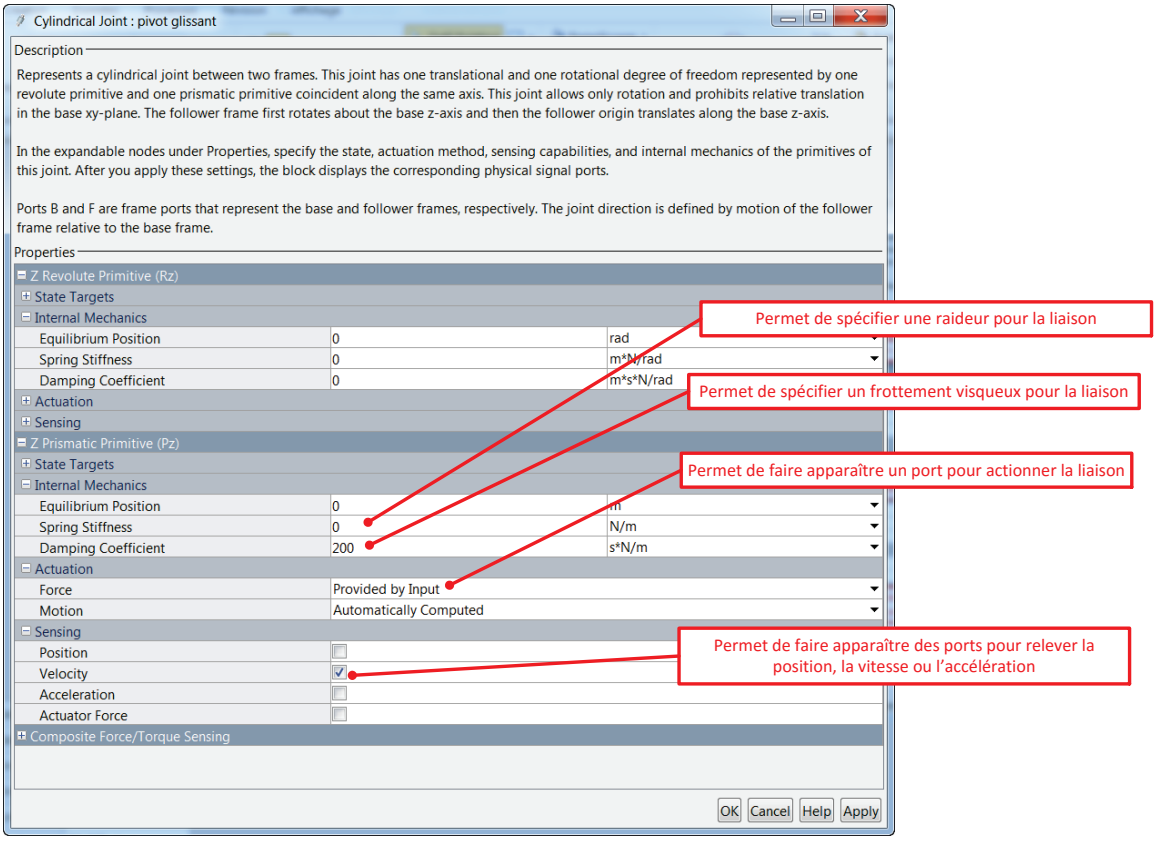


Figure 317 : paramétrage des ports d'une liaison

Des ports apparaissent sur le bloc de la liaison. **Connecter** l'interface conformément à la Figure 318.

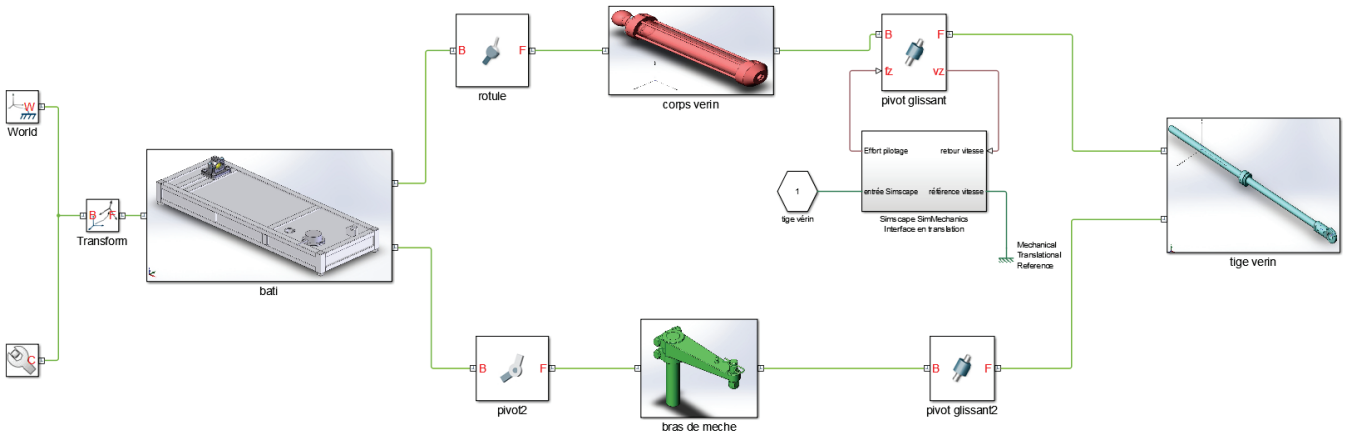


Figure 318 : connexion entre Simscape et SimMechanics

L'interface entre Simscape et SimMechanics étant réalisée, nous allons maintenant ajouter sur la liaison « pivot2 » entre le « Bâti » et le « Bras de mèche ».

- Un port pour imposer l'effort résistant qui agit sur la barre du bateau.
- Un port pour prélever l'angle de rotation de la barre du bateau.

Pour cela **double cliquer** sur la liaison « pivot2 » et ajouter un port pour imposer une action mécanique sur la liaison comme représenté sur la Figure 319.

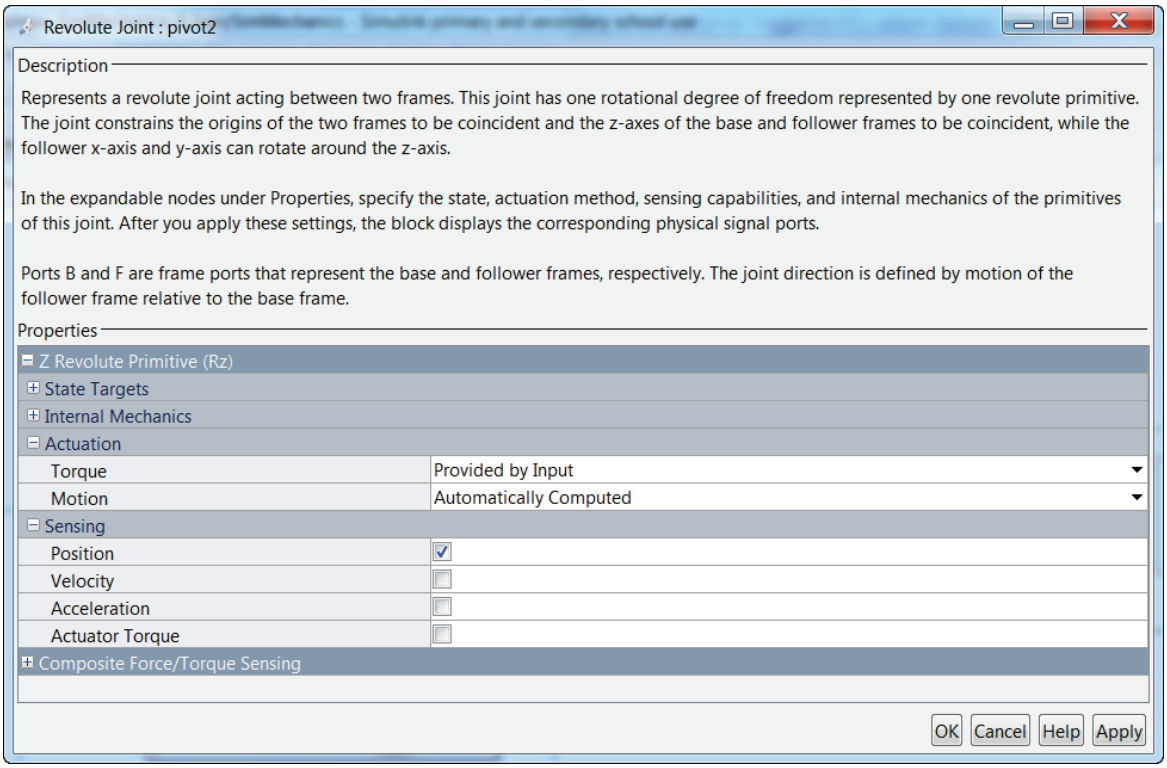


Figure 319 : ajout d'un port pour imposer un couple à la liaison

Les deux nouveaux ports apparaissent maintenant sur la liaison. Ajouter un bloc **PS-S** pour convertir le signal d'angle de rotation de la barre en signal **Simulink**. Choisir les degrés comme unité d'angle.

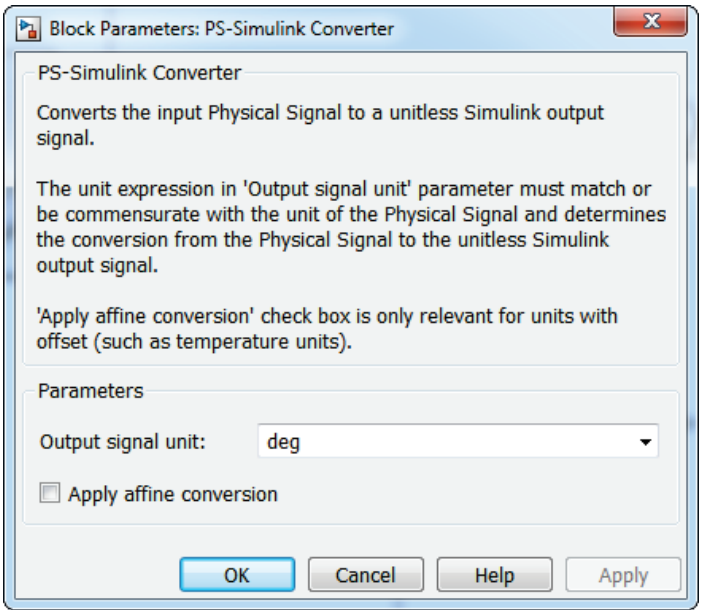


Figure 320 : conversion du signal physique en signal Simulink

Vous devez obtenir la configuration de la Figure 321.

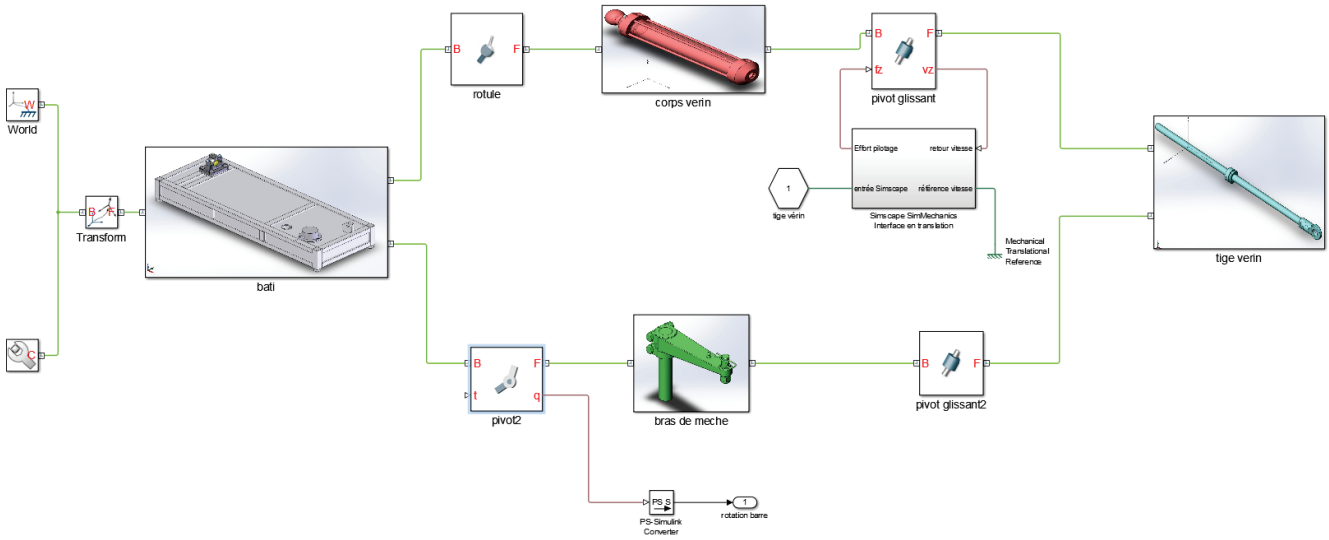


Figure 321 : ajout de port sur une liaison dans SimMechanics

En début de simulation l'angle de rotation de la barre possède une valeur arbitraire fixée lors de l'importation du modèle depuis Solidworks. Il est possible de connaître la valeur initiale (offset) de la barre.

Pour cela **double-cliquer** sur la liaison « pivot2 » et développer **State Target/Specify Position Target.**

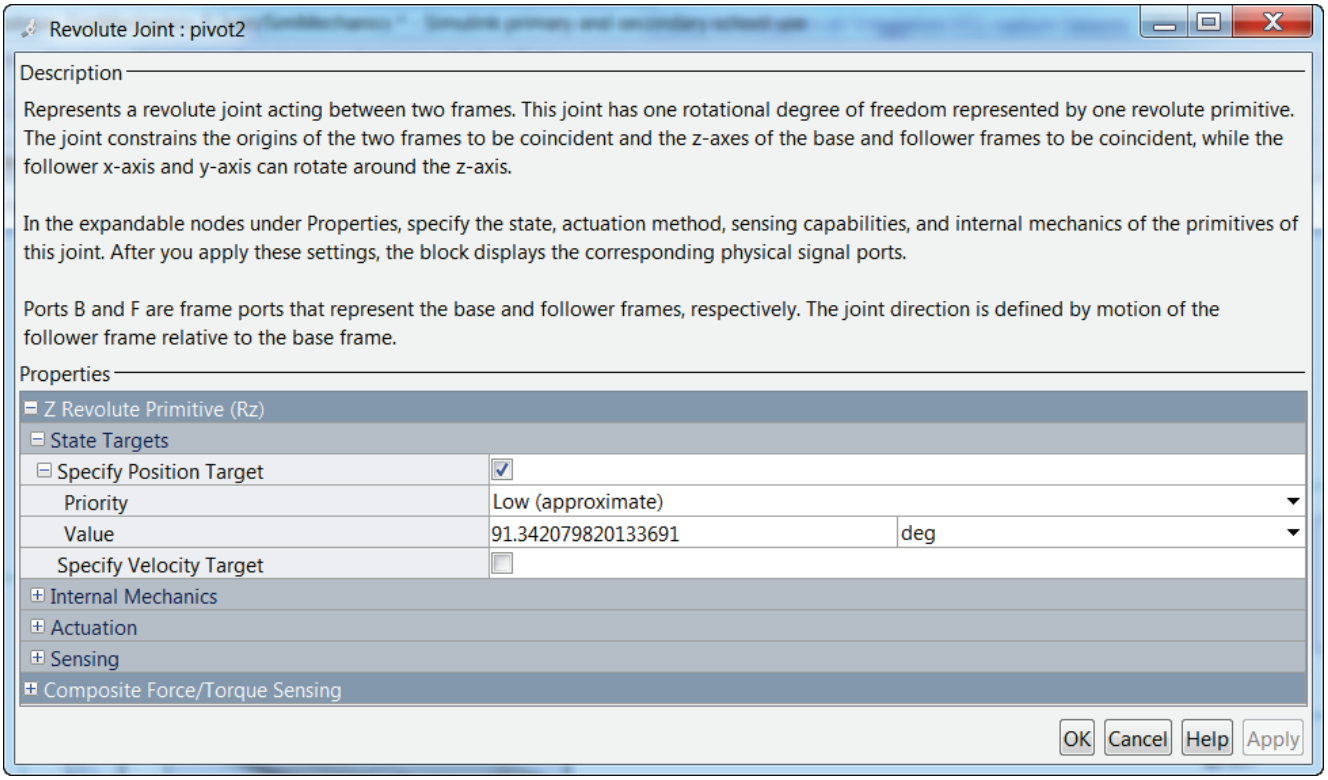


Figure 322 : visualisation de l'offset de l'angle de barre

En début de simulation l'angle de la barre doit être nul, il faut donc retrancher cette offset à la valeur de l'angle renvoyé par la liaison. Pour cela **compléter** le modèle conformément à la Figure 323 en ajoutant un bloc **constant** que vous compléterez en faisant un copier/coller de la valeur de l'offset de l'angle de barre et un soustracteur pour retrancher l'offset au signal.

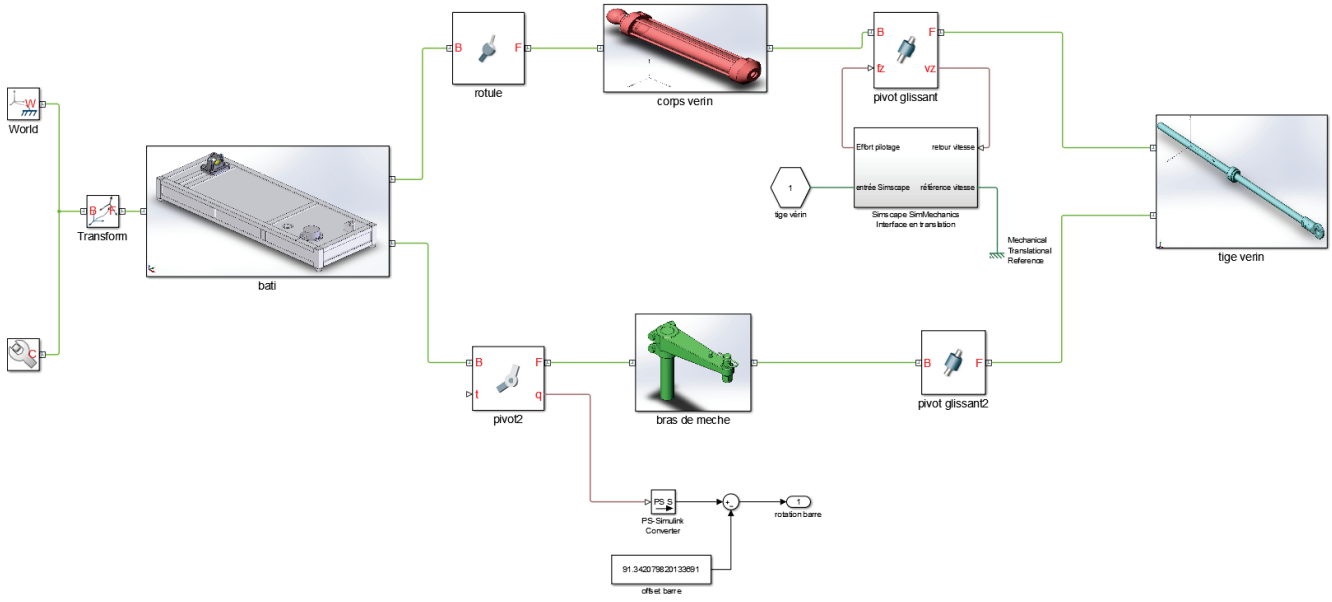
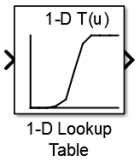


Figure 323 : compensation de l'offset de l'angle de la barre

4. Modélisation d'un effort extérieur variable

Le moment qui s'oppose à la rotation de la barre durant le mouvement varie en fonction de l'angle de rotation de la barre. Cette variation étant non linéaire, nous utiliserons un bloc **1D Lookup Table** pour le modéliser. Le **1D Lookup Table** représente un gain qui pourra varier en fonction de la valeur de l'entrée. Ce bloc est très pratique pour modéliser des lois entrée sortie non linéaires de types géométrique ou dynamique.

Désignation	Représentation	Bibliothèque
1D Lookup Table		Simulink/ Lookup Tables

Ouvrir le fichier ***pilote_SimMechanics_fin.slx*** et observer l'ajout de l'effort non linéaire sur la liaison. L'angle de la barre est prélevé et le modèle impose à la liaison « pivot2 » un effort qui varie en fonction de l'angle de la barre.

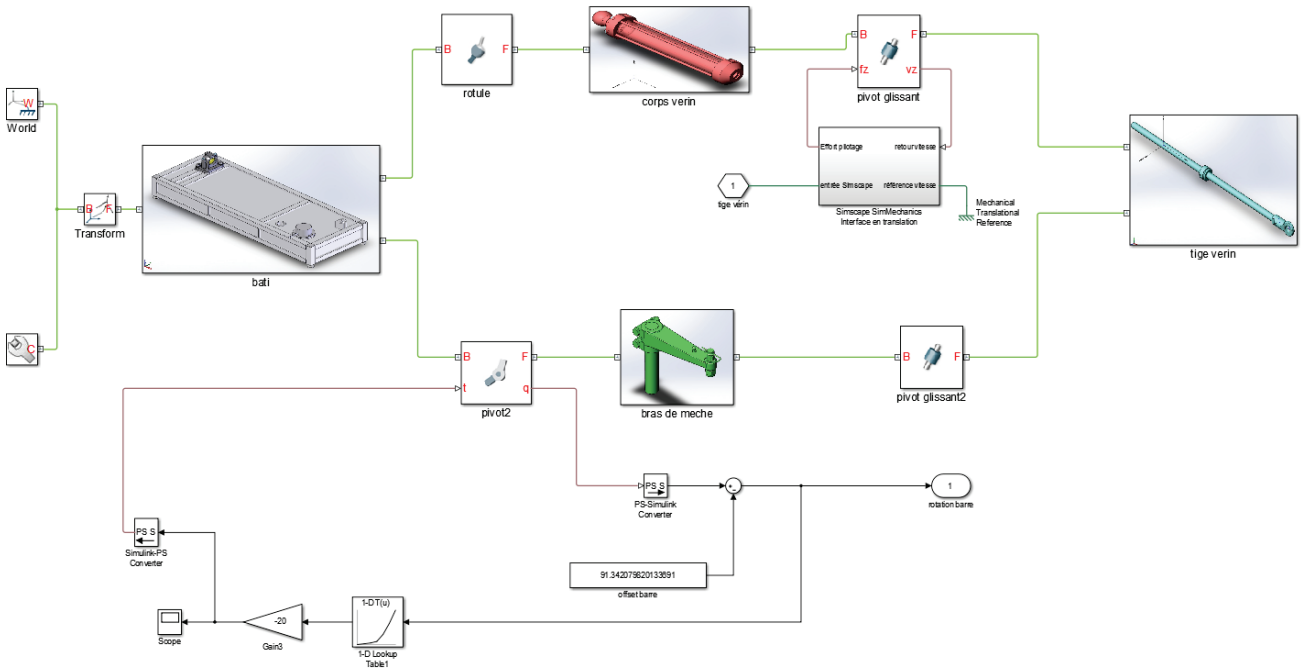


Figure 324 : ajout d'un effort sur la liaison, utilisation d'une Lookup Table

Double cliquer sur le bloc **1D Look-up Table** pour explorer le paramétrage de ce type de bloc.

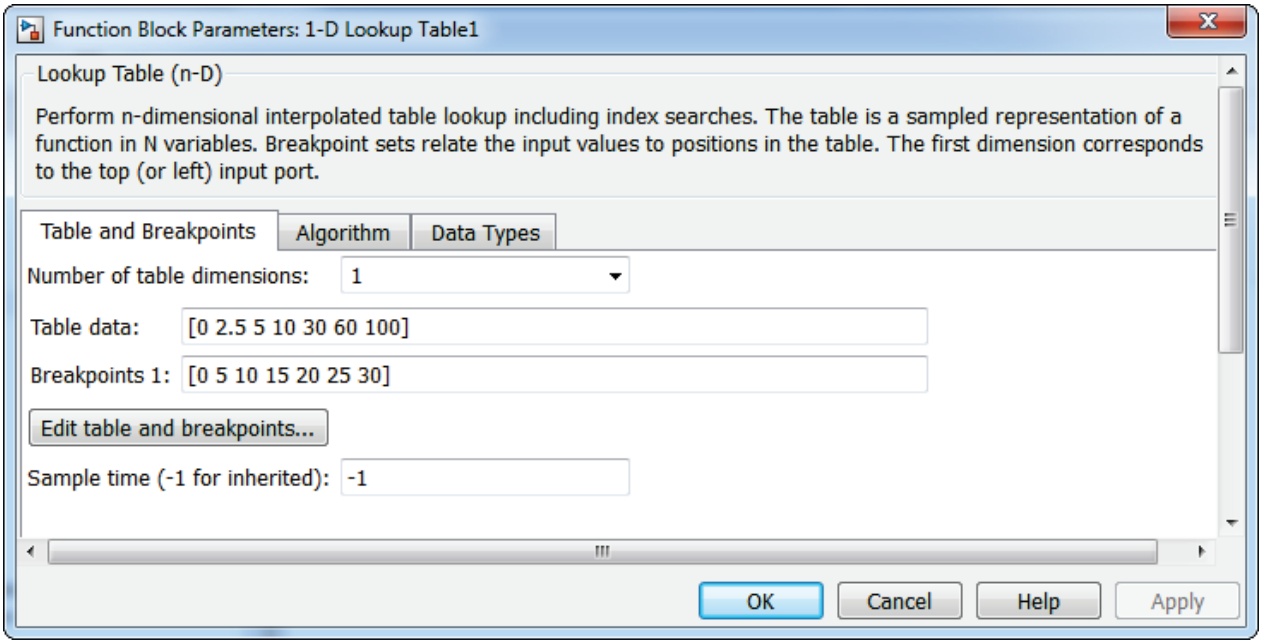


Figure 325 : paramétrage d'une Lookup Table

Pour paramétrer la table, il suffit d'entrée des points appartenant à la courbe qui caractérise la loi entrée sortie du bloc. Le logiciel effectue automatiquement une interpolation pour associer à chaque valeur de l'entrée, la valeur de la sortie correspondante.
Les coordonnées des points sont saisies dans les champs **Breakpoints 1** (abscisses) et **Table data** (ordonnées).

Il est possible de visualiser la loi ainsi saisie en cliquant sur **Edit table breakpoints.**

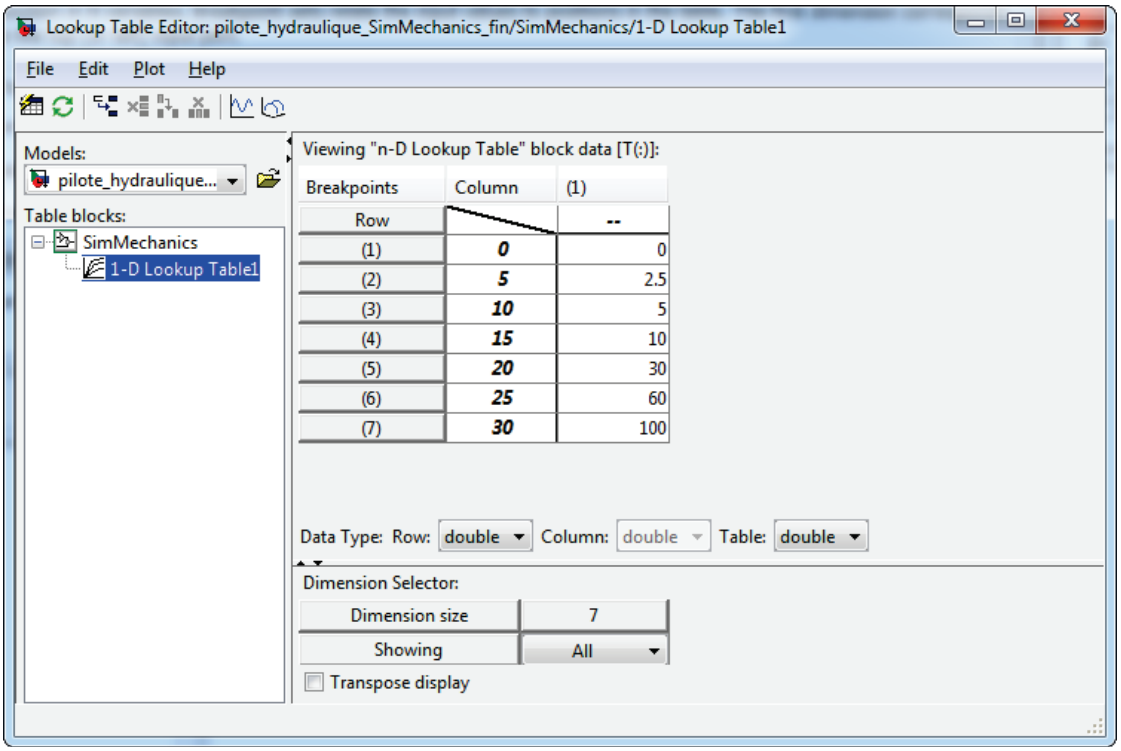


Figure 326 : visualisation d'une Lookup Table

Il est possible de visualiser la courbe représentant la loi entrée sortie en cliquant sur **Linear Plot**  .

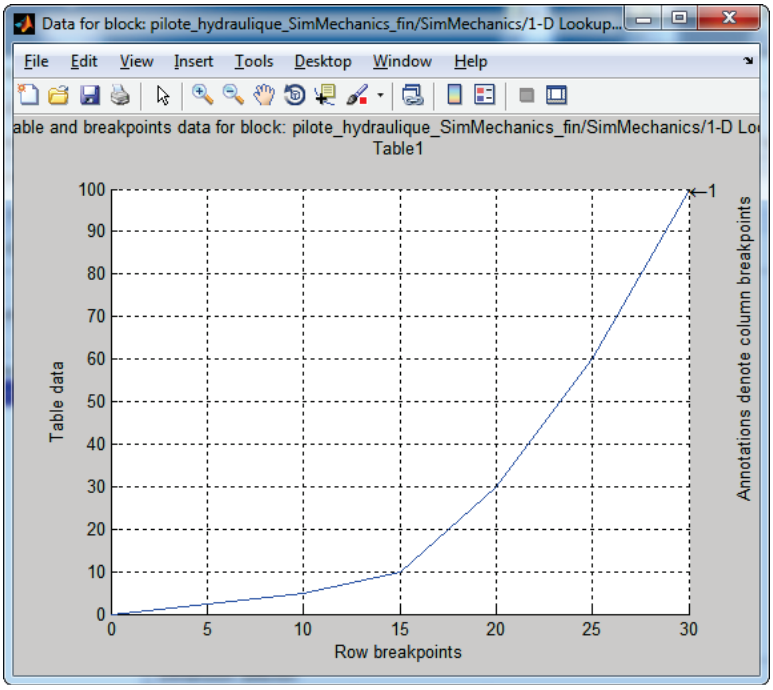


Figure 327 : visualisation de la courbe représentative de la loi entrée sortie d'une Lookup Table

Valider le paramétrage de la lookup Table.

C. Résultat de la simulation

Lancer la simulation et observer la réponse en cap du bateau qui prend en compte le modèle **SimMechanics** ajouté.

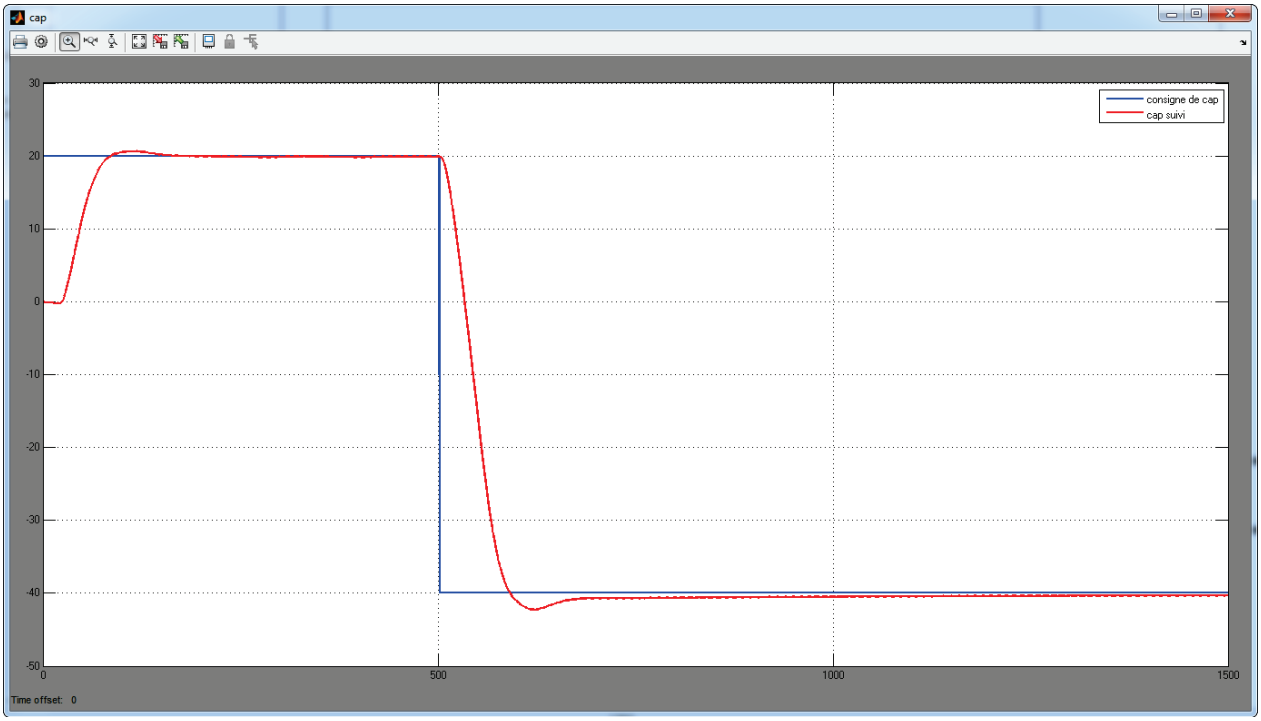


Figure 328 : résultat de la simulation du cap suivi par le bateau

III. Importation d'un modèle SolidWorks dans SimMechanics

A. Les principes

Pour exploiter dans l'environnement **MATLAB** – Simulink une maquette CAO 3D issu de Solidworks, il faut convertir cette maquette en un fichier portant l'extension **xml**. Pour cela il faut d'abord installer dans SolidWorks un addon « **SimMechanics Link** » permettant de réaliser la conversion d'un fichier assemblage de Solidworks en fichier xml. Le fichier xml est ensuite converti par **MATLAB** en fichier exploitable par SimMechanics.
Lors de la conversion chaque fichier « pièce » de Solidworks sera converti en fichier STL. Ces fichiers contiennent les formes des pièces et les caractéristiques cinétiques. La présence de ces fichiers dans le « path » de **MATLAB** est indispensable afin de visualiser les animations dans la fenêtre **Mechanics Explorer.**

B. Installation de « SimMechanics Link »

Pour effecteur le téléchargement de **SimMechanics Link**, vous devez posséder un compte Mathworks . Si vous ne possédez pas de compte Mathworks, vous devez donc le créer.

Solidworks et MATLAB doivent être installés sur votre ordinateur.

Connectez-vous avec votre login et votre mot de passe.
Rendez-vous à l'adresse suivante :

http://www.mathworks.fr/products/simmechanics/download_smlink.html

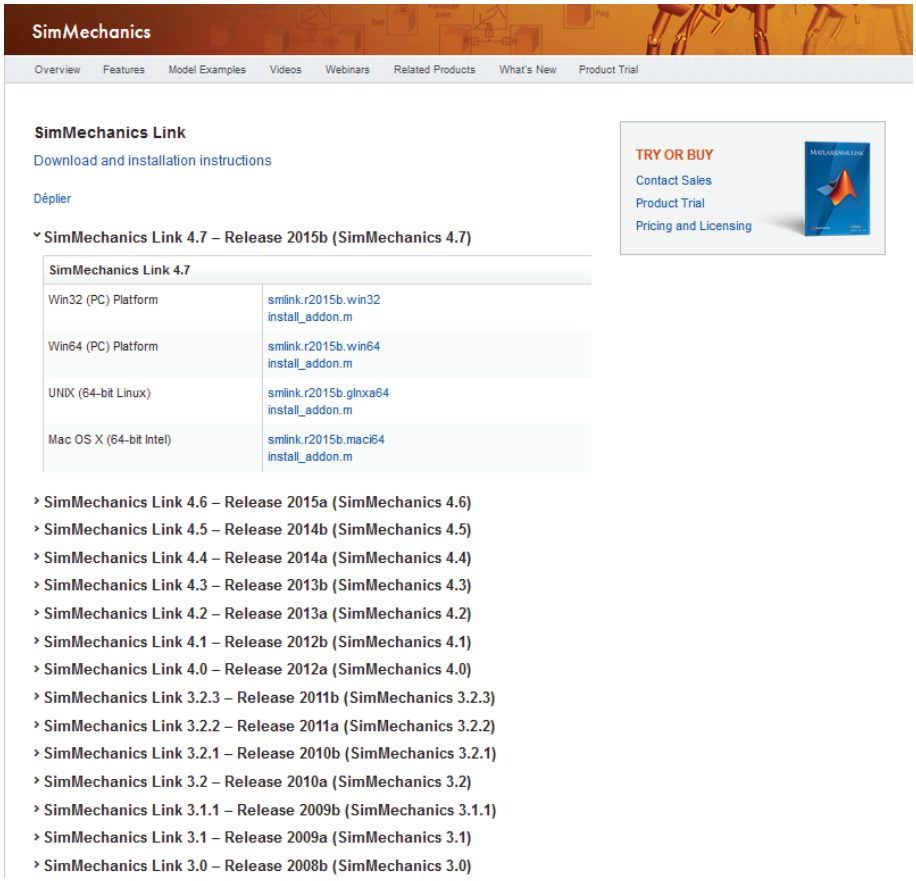


Figure 329 : choix de la version de SimMechanics Link

Il faut choisir la version de **SimMechanics Link** correspondant à votre version de **MATLAB** et télécharger les deux fichiers correspondant à votre système d'exploitation et les placer impérativement dans un dossier du « path » de **MATLAB.**

- Fichier script MATLAB : **install_addon.m**
- Fichier archive qui contient les fichiers à installer : **smlink.r…**

Dans la fenêtre de commande de MATLAB taper la commande qui lance l'installation de SimMechanics Link :

```
>> install_addon('smlink.r2015b.win64.zip')
.....
Installation of smlink complete.

To view documentation, type "doc smlink"
```

Attention, le nom du fichier archive contenu entre les parenthèses de la commande **install_addon** doit être exactement celui que vous avez téléchargé et dépend donc de votre version de MATLAB et de votre système d'exploitation.

Une fois l'installation terminée, taper la commande suivante afin de créer le lien entre MATLAB et Solidworks.

```
>> smlink_linksw
```

La procédure d'installation est terminée.

C. Conversion d'un fichier assemblage de Solidworks en fichier xml

Lancer Solidworks.

La fenêtre d'accueil du logiciel s'ouvre

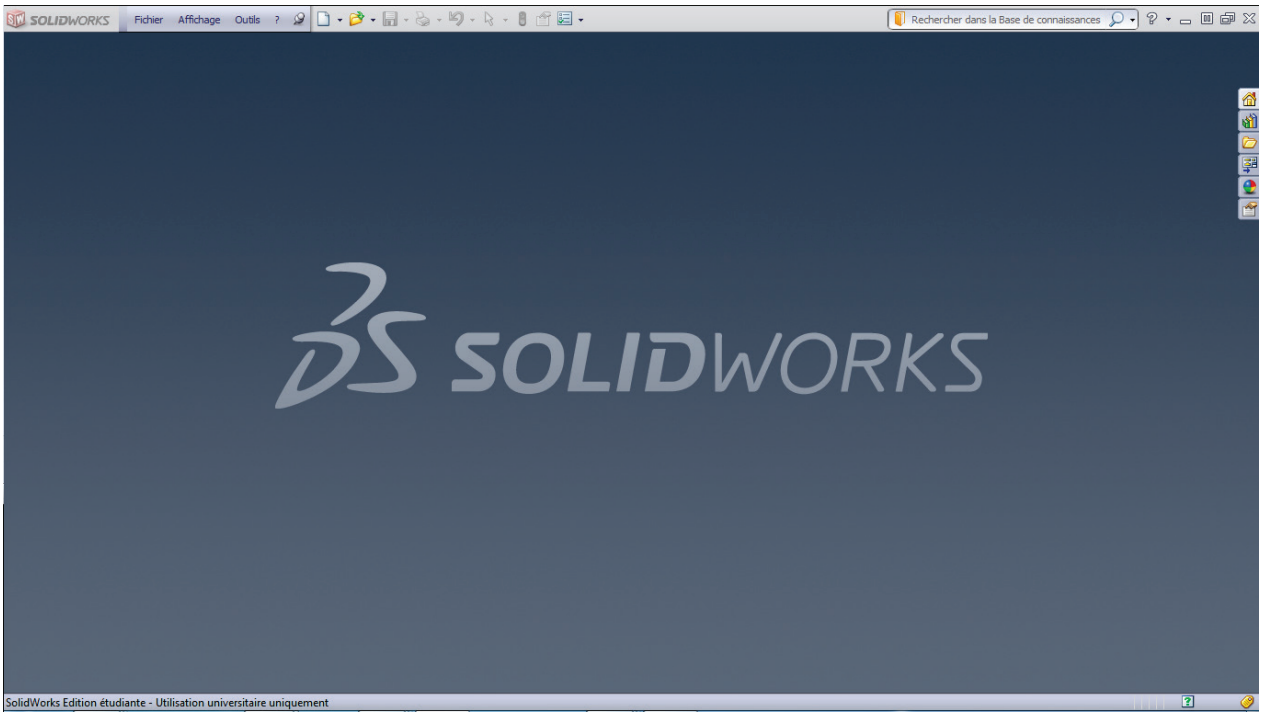


Figure 330 : fenêtre d'accueil de Solidworks

Lancer Solidworks.

La fenêtre d'accueil du logiciel s'ouvre, dérouler le menu **options**  puis sélectionner **compléments** pour ouvrir la fenêtre de définition des compléments du logiciel.

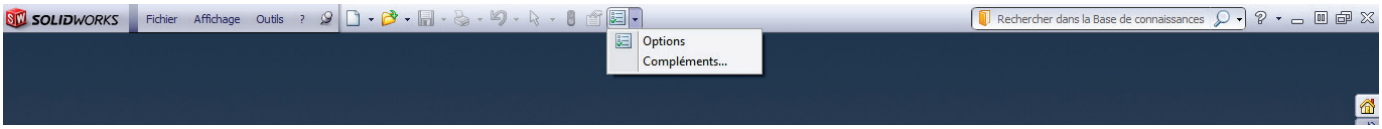


Figure 331 : ouverture de la fenêtre des compléments de Solidworks

Cocher les deux cases à gauche et à droite de **SimMechanics Link** afin d'activer **SimMechanics Link** à chaque démarrage de Solidworks (Figure 332).

Cliquez sur **OK**.

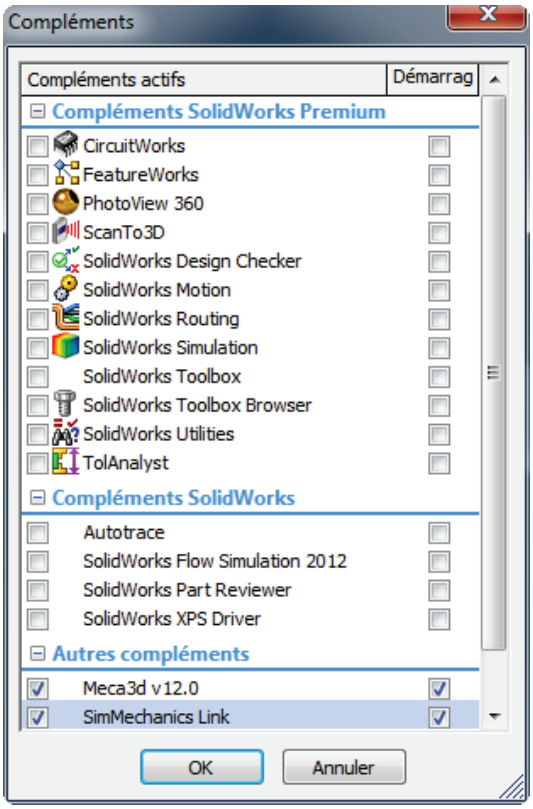


Figure 332 : sélection des compléments de Solidworks

Quitter, puis **relancer** Solidworks.

Avec Solidworks, **Ouvrir** le fichier ***Assemblage pilote hydraulique.sldasm*** dans le dossier « Modèle CAO 3D du pilote/Partie Mécanique »

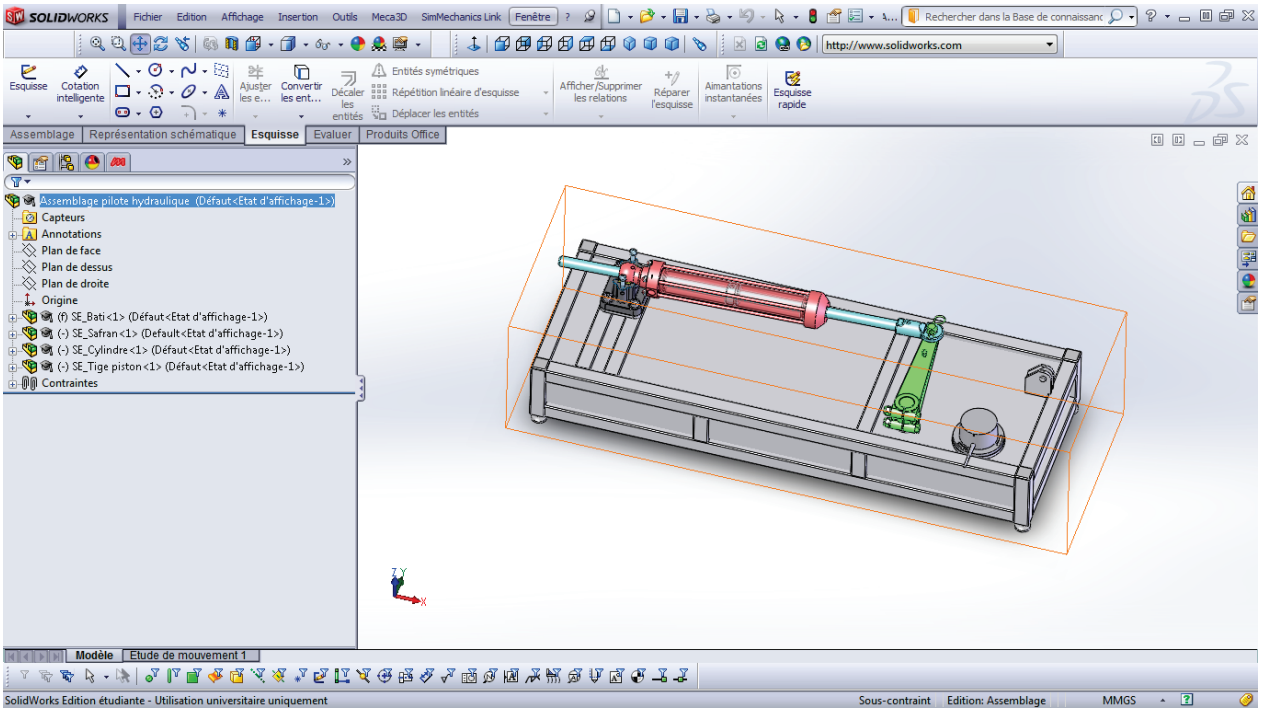


Figure 333 : ouverture du modèle Solidworks de la partie mécanique du pilote

Pour convertir l'assemblage au format **xml**, sélectionner **SimMechanics Link/export/SimMechanics Second Generation** (Figure 334)**.** Prendre soin de choisir comme dossier d'exportation un dossier qui appartient au « path » de MATLAB. Ici nous prendrons le dossier vide « conversion_pilote-xml » placé au même niveau que le dossier « Modèle CAO 3D du pilote ». Choisir comme nom de fichier ***Assemblage_pilote_hydraulique***.

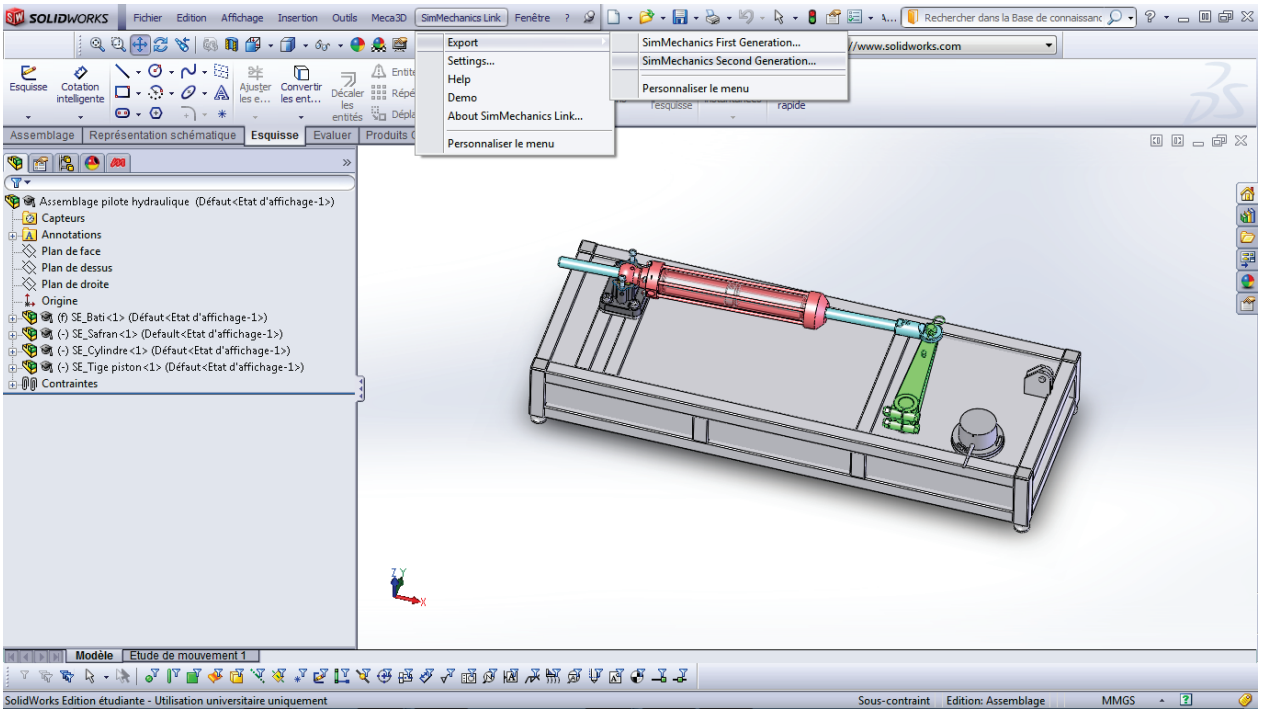


Figure 334 : conversion de l'assemblage en fichier xml

Après quelques secondes de réflexion, Solidworks va ouvrir et fermer tous les fichiers pièces qui constituent l'assemblage pour les convertir au format STL. La fenêtre de la Figure 335 indique la fin de la conversion.

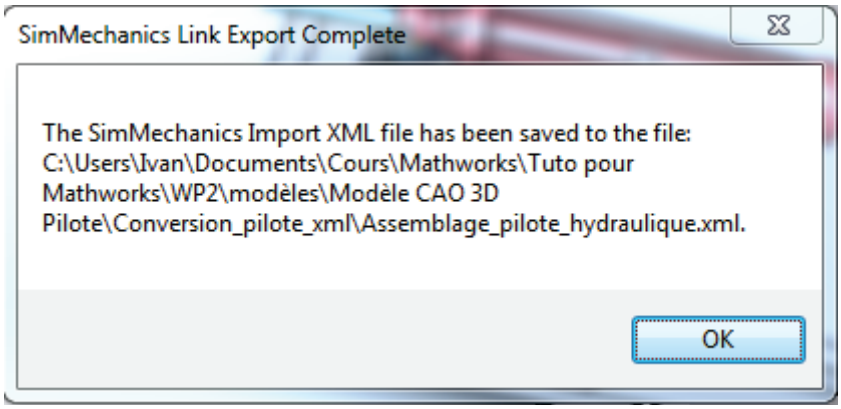


Figure 335 : fin de la conversion des fichiers

Ouvrir le dossier « conversion_pilote_xml » avec l'explorateur Windows pour constater qu'il contient toutes les pièces du mécanisme au format **STL** ainsi qu'un fichier nommé **Assemblage_pilote_hydraulique.xml.**

Retourner dans MATLAB et taper dans la fenêtre de commande :

```
>>smimport('Assemblage_Pilote_hydraulique.xml')
```

Cette commande va permettre de créer le fichier **SimMechanics** correspondant à l'assemblage et ouvre automatiquement un nouveau fichier contenant le modèle.

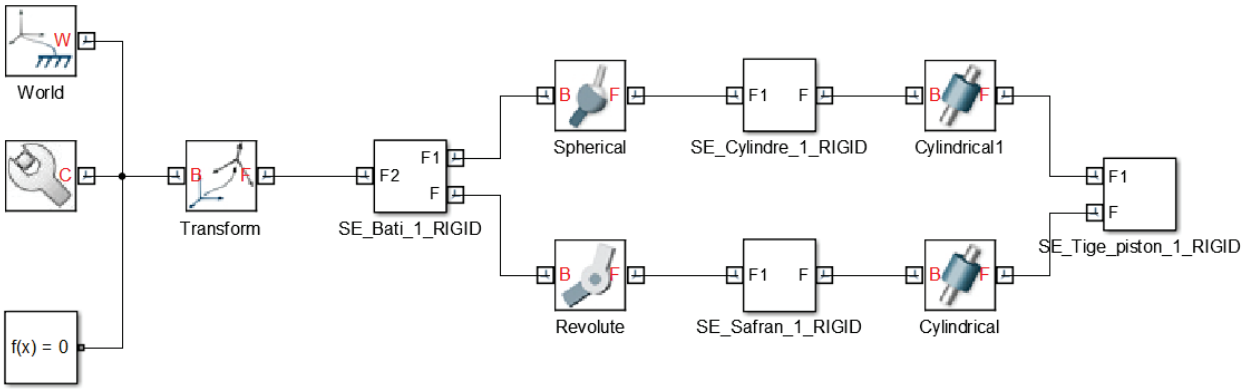


Figure 336 : modèle SimMechanics obtenu

Nous retrouvons le fichier sur lequel nous avons travaillé au début de ce chapitre.

Lancer la simulation, la fenêtre Mechanics Explorer s'ouvre et vous pouvez visualiser la maquette du pilote dans MATLAB. Il se peut que la pesanteur ne soit pas par défaut sur le bon axe.

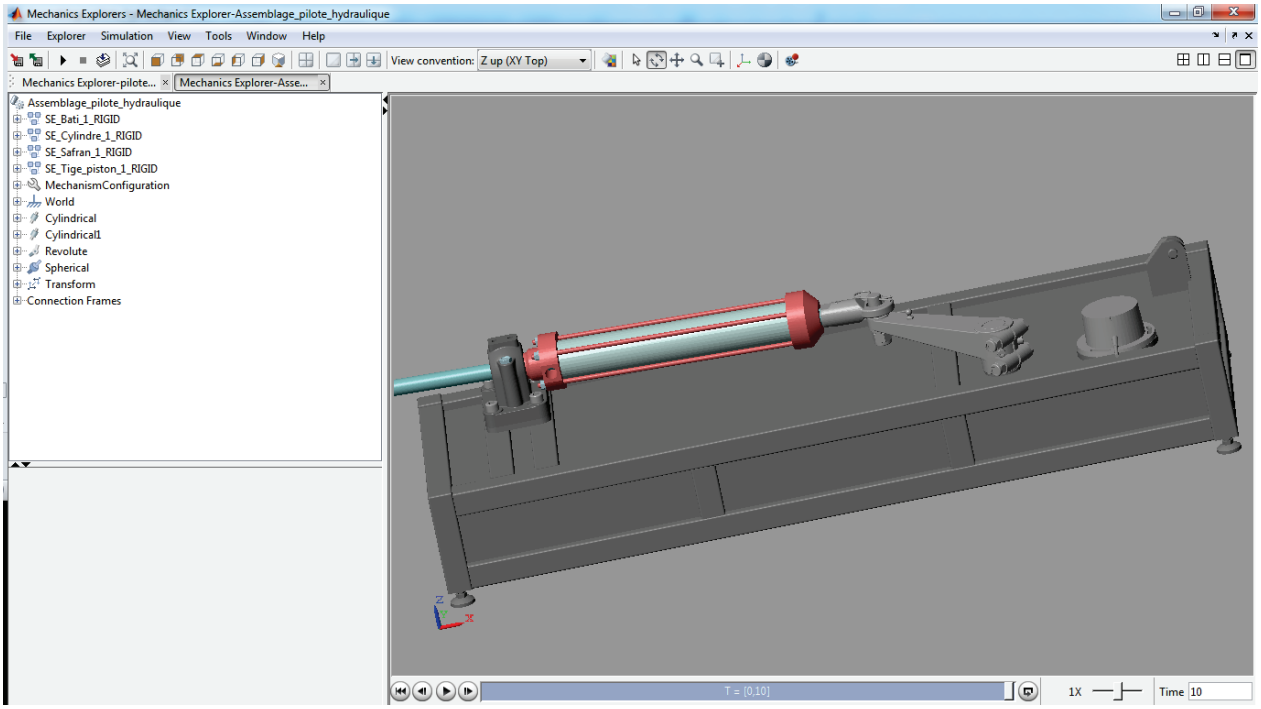


Figure 337 : fenêtre Mechanics Explorer

Chapitre 7 : L'identification d'un modèle

I. La modélisation black-box, l'identification

A. Présentation de la méthode

Cette approche, consiste à associer un modèle mathématique à un comportement mesuré expérimentalement. Le système est considéré ici comme une boite noire avec une grandeur d'entrée et une grandeur de sortie. Le principe consiste à imposer une grandeur d'entrée connue au système et à relever la grandeur de sortie. Il suffit ensuite de trouver un modèle mathématique pour la fonction de transfert du système qui donnera la même relation entre l'entrée et la sortie. Cette méthode donne uniquement un modèle du comportement global du système sans se préoccuper de l'influence séparée des différents paramètres sur les performances.

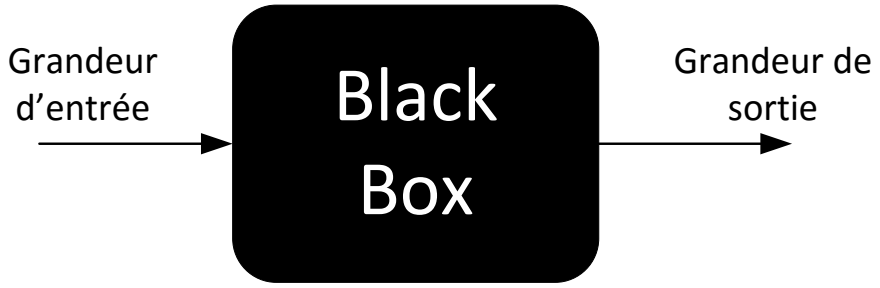


Figure 338 : la modélisation black box

Cette méthode permet d'obtenir un modèle validé expérimentalement sans avoir à écrire les équations qui régissent le comportement du système. Le modèle doit être manipulé avec précaution dans le cas ou des phénomènes non-linéaires entre en compte dans le comportement réel du système. En pratique on réalise toujours plusieurs essais correspondant à des sollicitations différentes (grandeur d'entrée constante ou sinusoïdale avec une variation de la fréquence).

Mise à part quelques cas simples, l'identification de la fonction de transfert demande une ressource de calcul importante. L'outil logiciel va nous permettre d'automatiser cette tâche en proposant des modèles mathématiques pour des processus d'identification qui peuvent être complexes.

MATLAB propose une toolbox appelé « System Identification » qui permet de mener à bien ce processus.

B. Mise en œuvre de la méthode en utilisant la toolbox Identification

1. Analyse des données utilisées pour l'identification

Ouvrir et exécuter le script **donnees_identification.m.**

Ce script permet d'afficher les grandeurs d'entrée et de sortie d'un essai en boucle ouverte sur un axe linéaire. Une tension est imposée aux bornes du moteur de l'axe linéaire (grandeur d'entrée) et la vitesse linéaire de l'axe est relevée (grandeur de sortie). Les conditions dans lesquelles l'essai a été réalisé sont illustrées Figure 339.

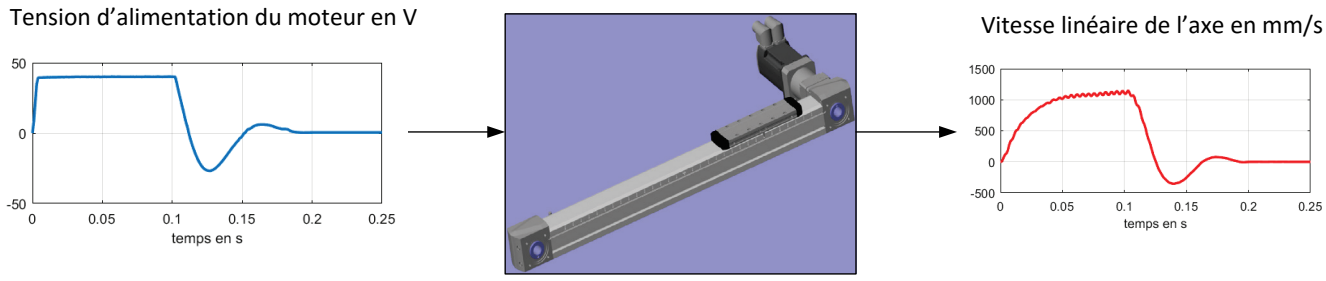


Figure 339 : les conditions de l'essai

La Figure 340 permet de visualiser les signaux d'entrée et de sortie.

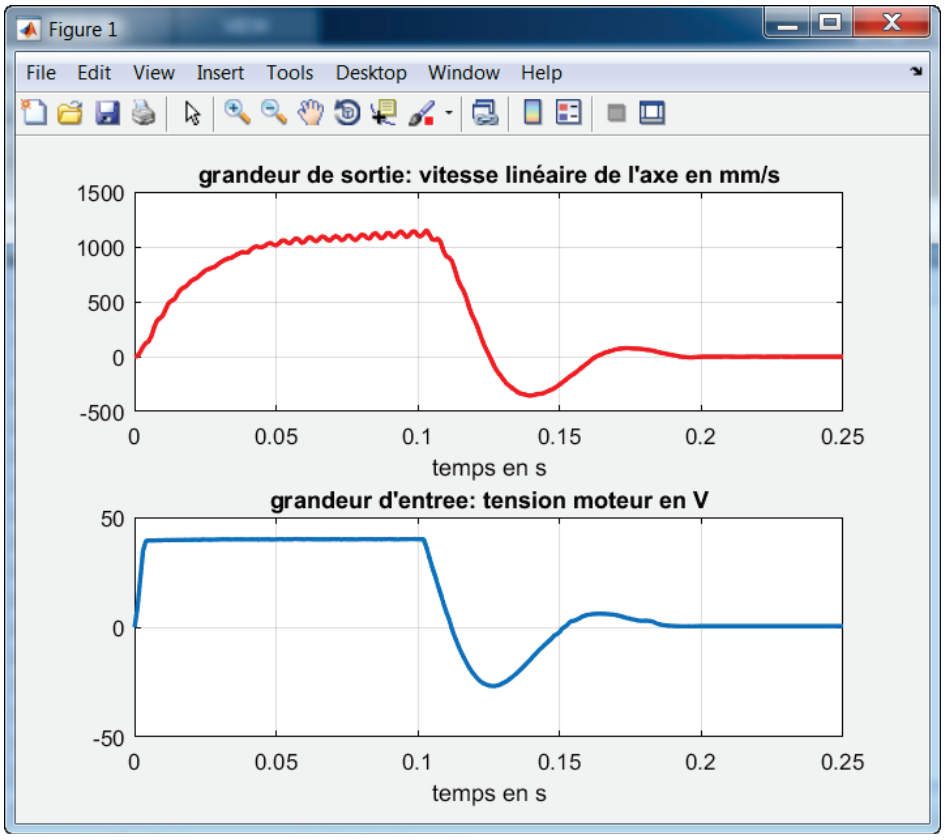


Figure 340 : visualisation des données utilisées pour l'identification

L'exécution du script a également crée dans le **Workspace** de MATLAB les variables suivante :

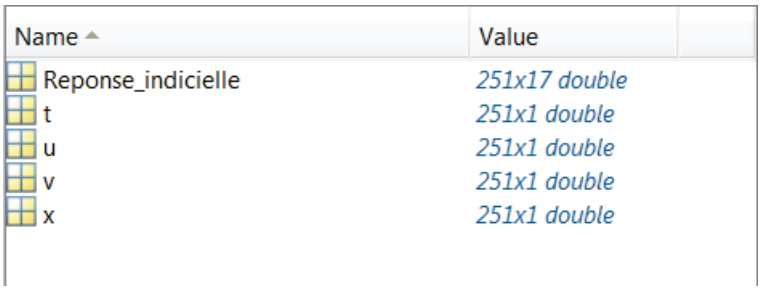


Figure 341 : visualisation dans le Workspace des données utilisées pour l'identification

- t : vecteur contenant la variable temps
- u : vecteur contenant la variable tension d'alimentation
- v : vecteur contenant la variable vitesse linéaire de l'axe
- x : vecteur contenant la variable position linéaire de l'axe

2. Ouverture et présentation de la toolbox « SystemIdentification »

Pour lancer la toolbox « System Identification », taper **ident** dans la fenêtre de commande.

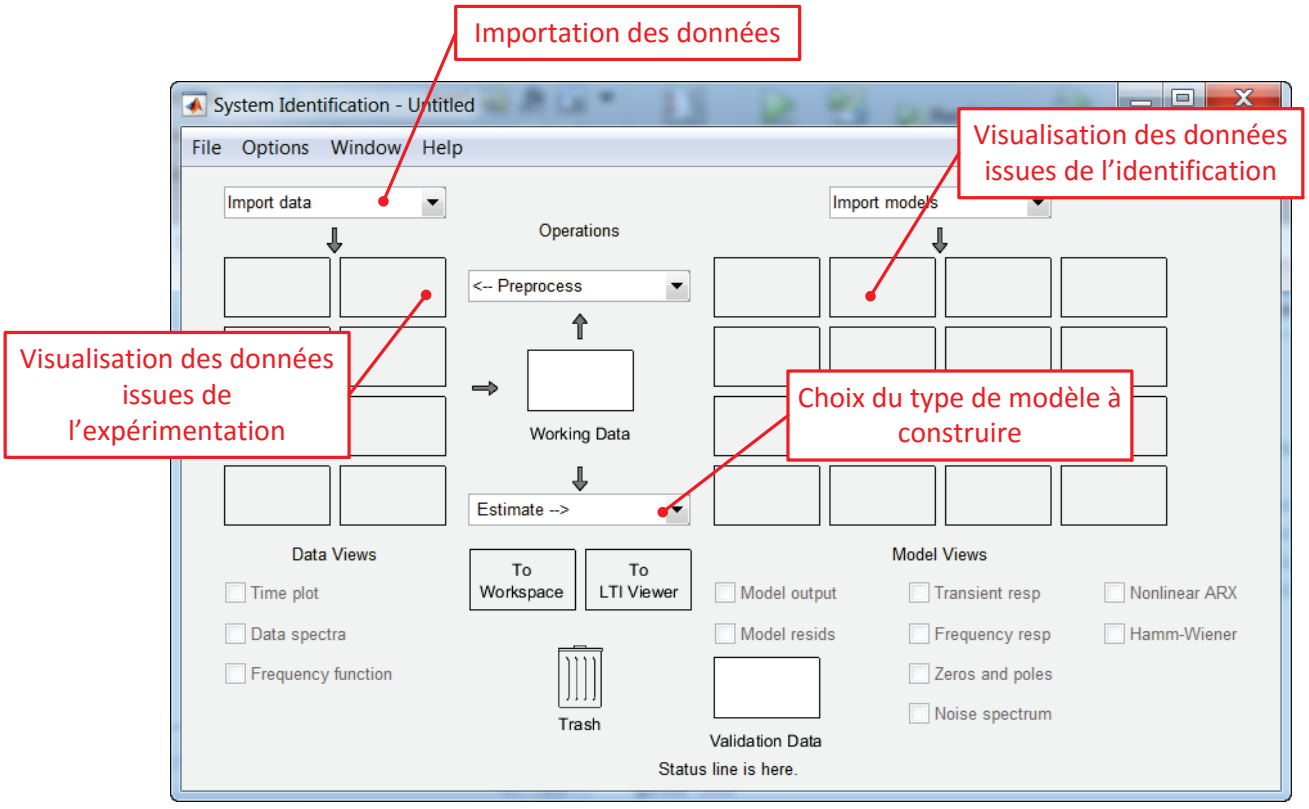


Figure 342 : description de la fenêtre de la toolbox identification

3. Importation des données

Il est possible d'importer tout type de données : temporelle, fréquentielle, Data object…
Dans notre exemple les données issues de l'expérimentation sont des données temporelles.

Sélectionner **Import data/Time domain data.**

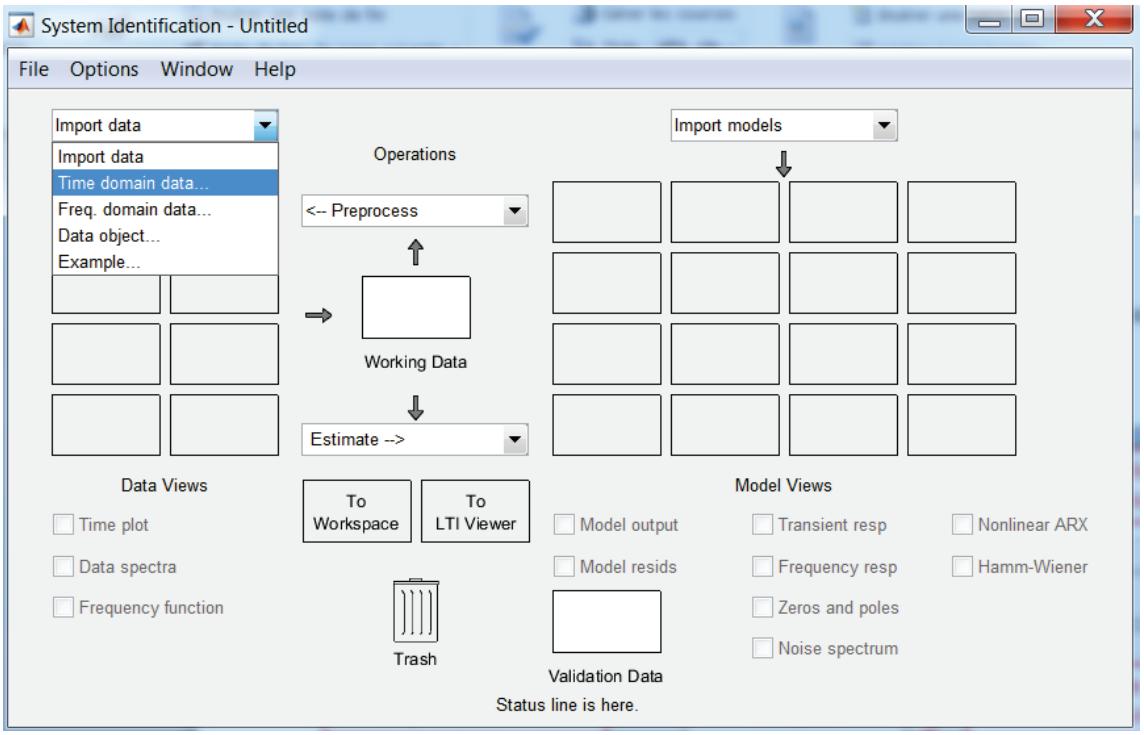


Figure 343 : importation de données temporelles

Cette commande permet d'ouvrir la fenêtre permettant de définir les données nécessaires pour l'identification. Compléter cette fenêtre avec les données de la Figure 344.

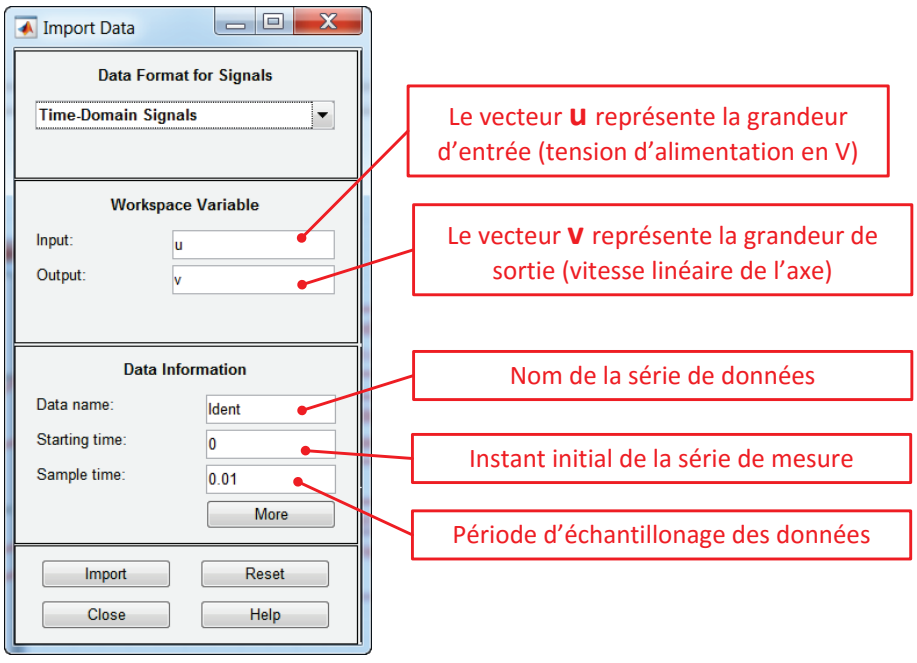


Figure 344 : sélection des données temporelles pour l'identification

Les données apparaissent alors dans la fenêtre de la « System Identification » toolbox comme le montre Figure 345. Cocher la case **Time plot** pour vérifier et visualiser la bonne importation des données (Figure 346).

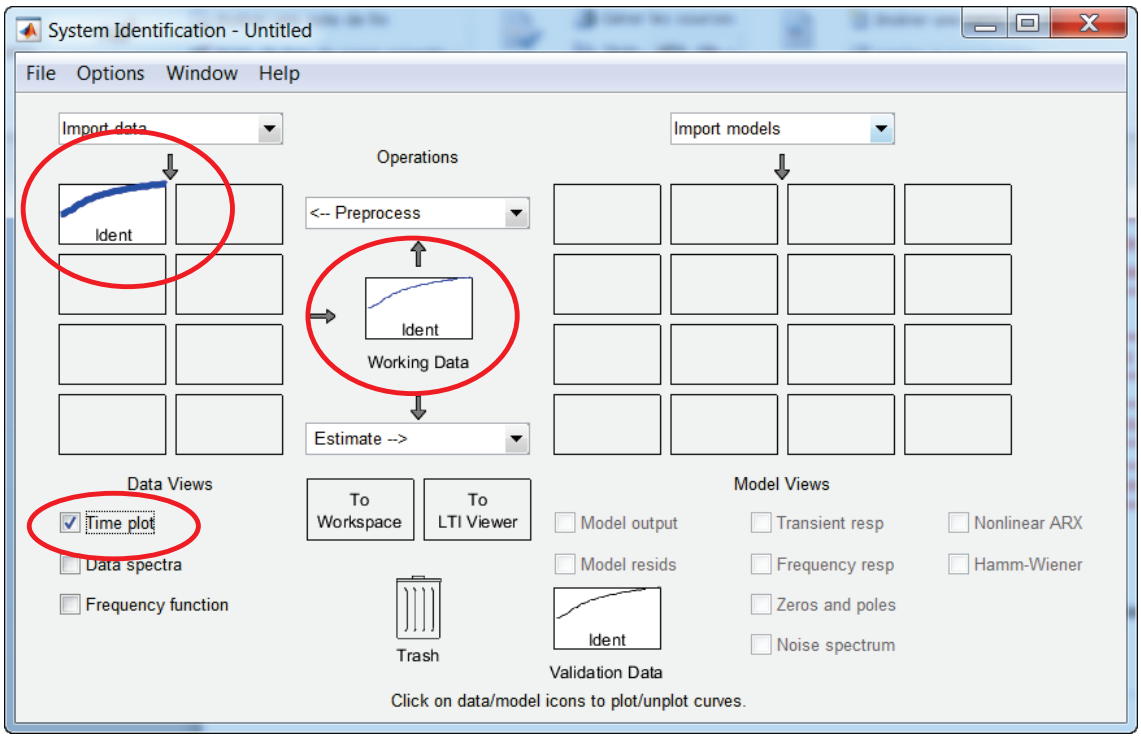


Figure 345 : visualisation des données importées dans la fenêtre de la « System Identification » toolbox

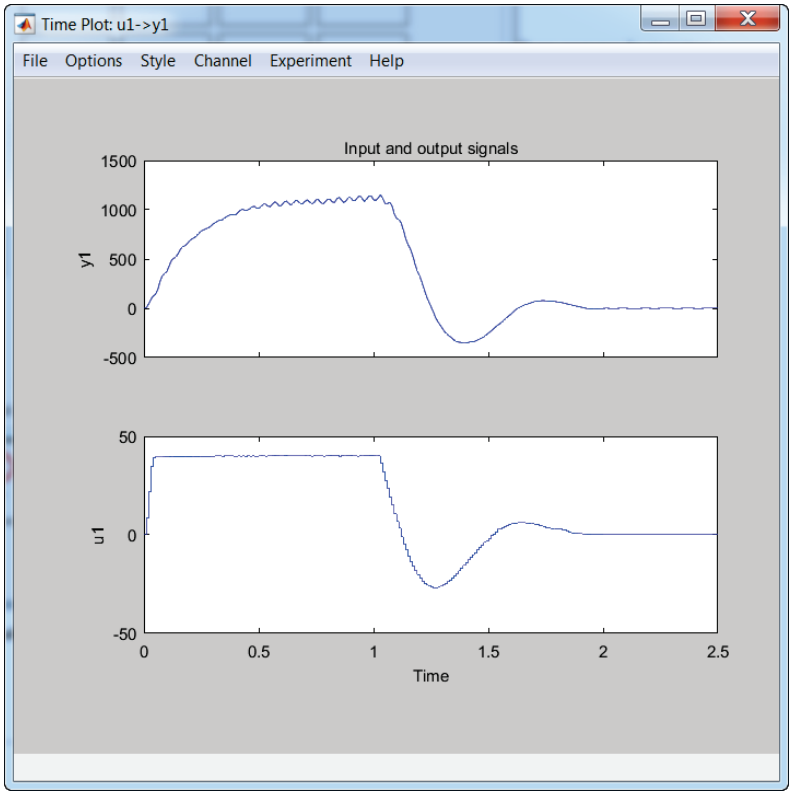


Figure 346 : visualisation des données importées dans la « System Identification »toolbox

Il faut maintenant choisir la forme du modèle que l'on souhaite obtenir (fonction de transfert, représentation d'état, modèle non-linéaires…). Il faudra choisir ici un modèle de type **fonction de transfert.**

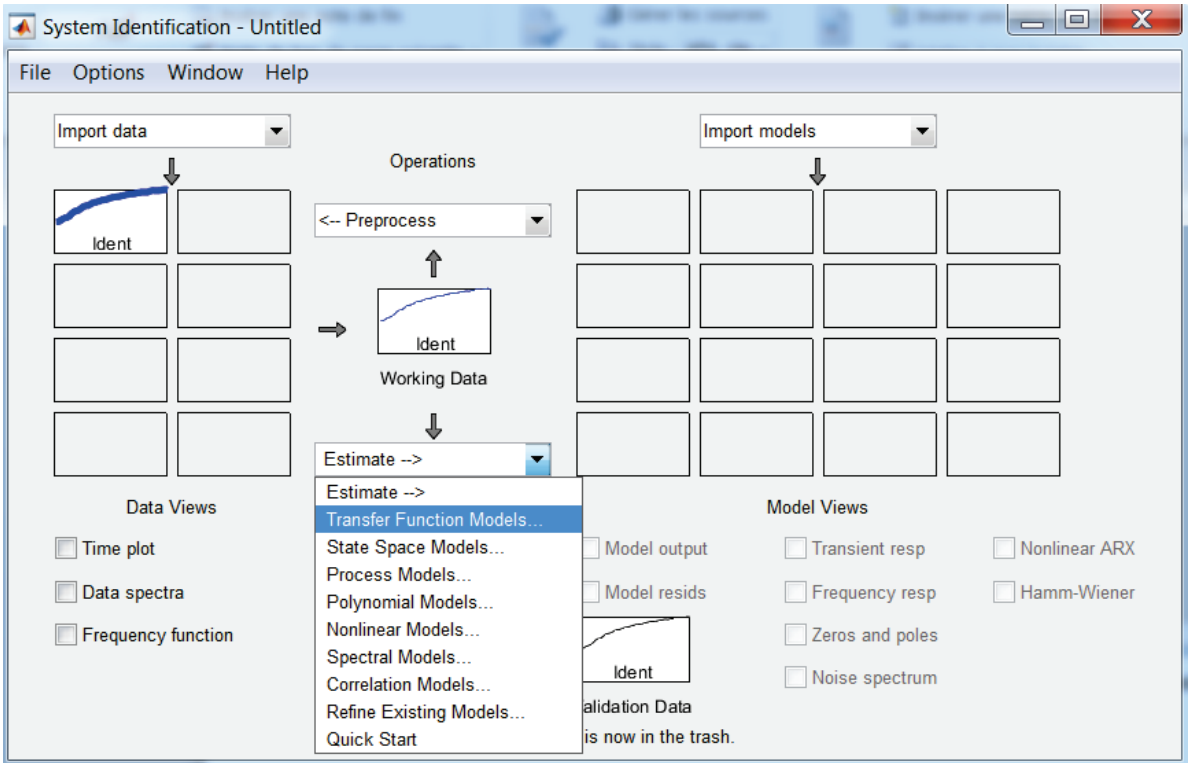


Figure 347 : choix de la forme du modèle dans la « System Identification » toolbox

La fenêtre qui s'ouvre permet de choisir la forme souhaitée pour la fonction de transfert. Ici nous prendrons une fonction de transfert du second ordre. Choisir **2 pôles et pas de zéro** puis cliquer sur **Estimate**.

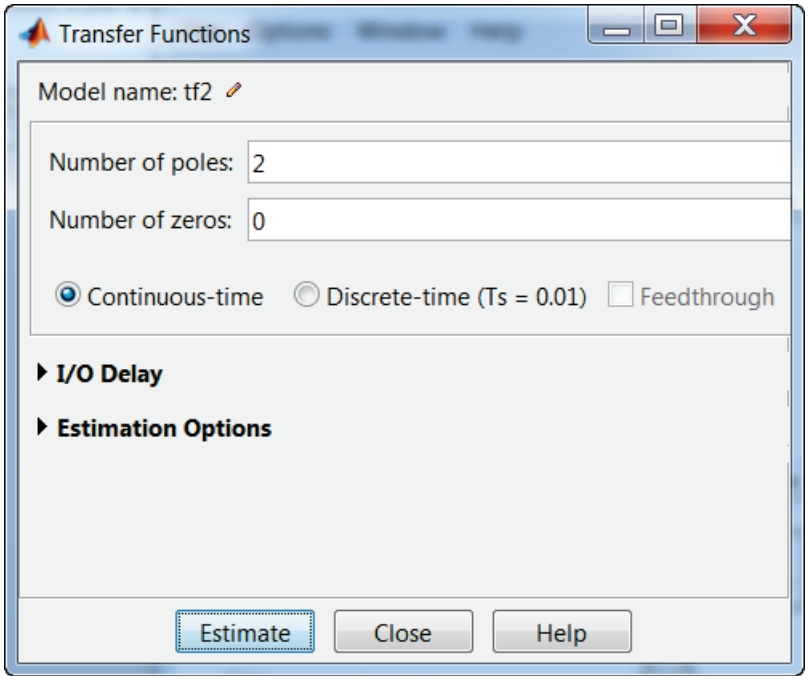


Figure 348 : choix de la forme de la fonction de transfert dans la « System Identification » toolbox

La fenêtre **Plant Identification Progress** donne les détails sur le processus numérique d'identification

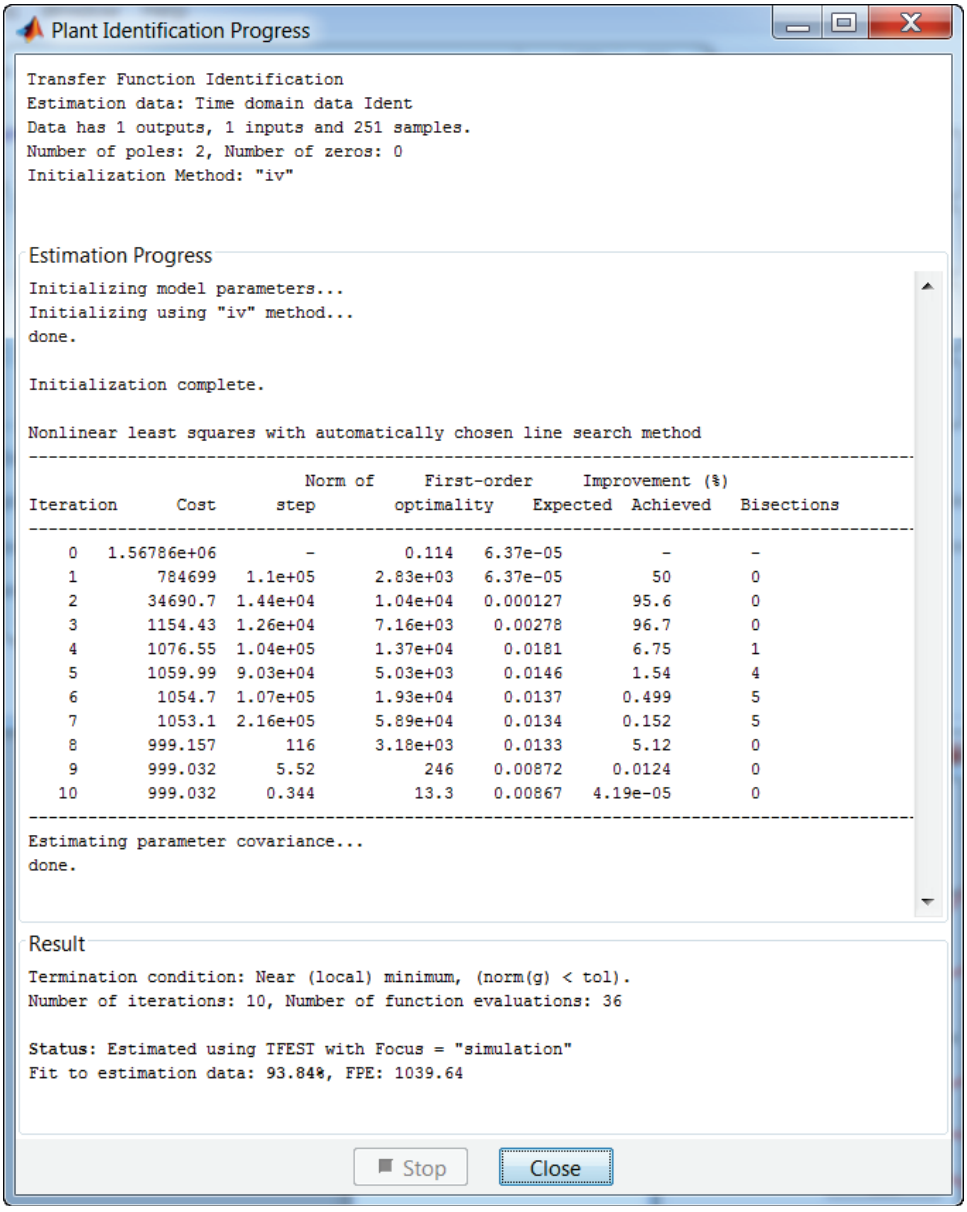


Figure 349 : la fenêtre Plant Identification Progress de la « System Identification » toolbox

Le résultat de l'identification est consultable en cliquant avec le bouton droit de la souris sur la case **tf1** représentant le résultat de l'identification (Figure 350 et Figure 351). Cocher également la case **Model output** pour visualiser la superposition des données expérimentales et du modèle issu de l'identification (Figure 352).

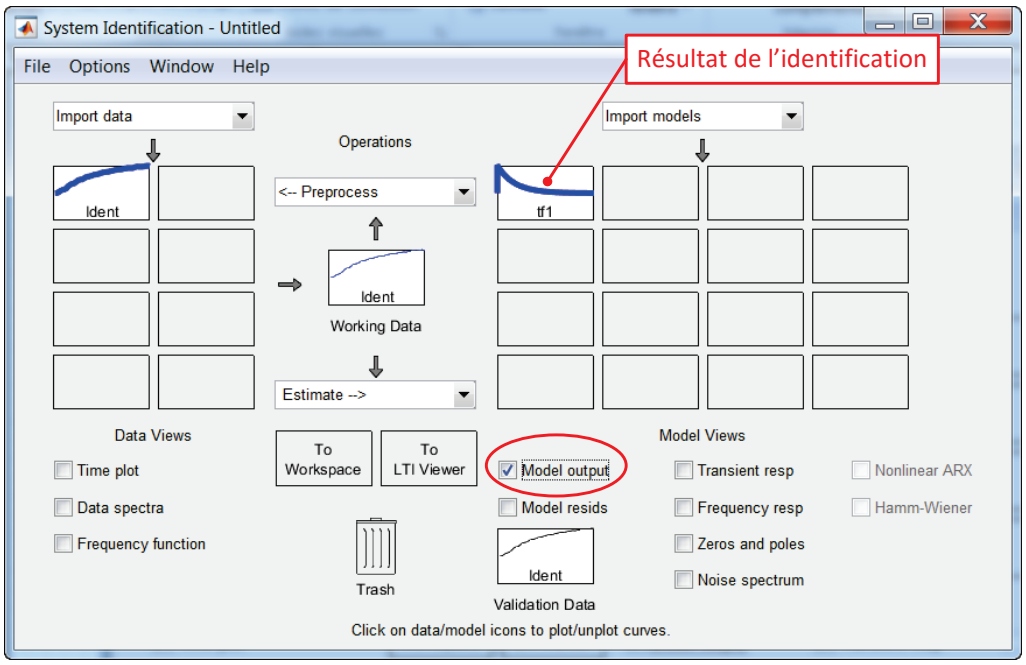


Figure 350 : visualisation de résultats de l'identification

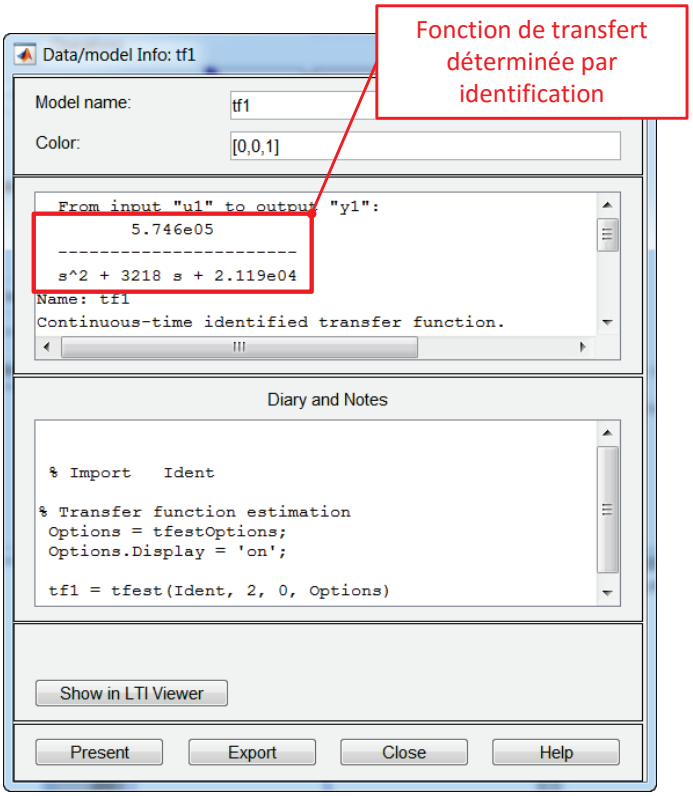


Figure 351 : fonction de transfert obtenue par identification

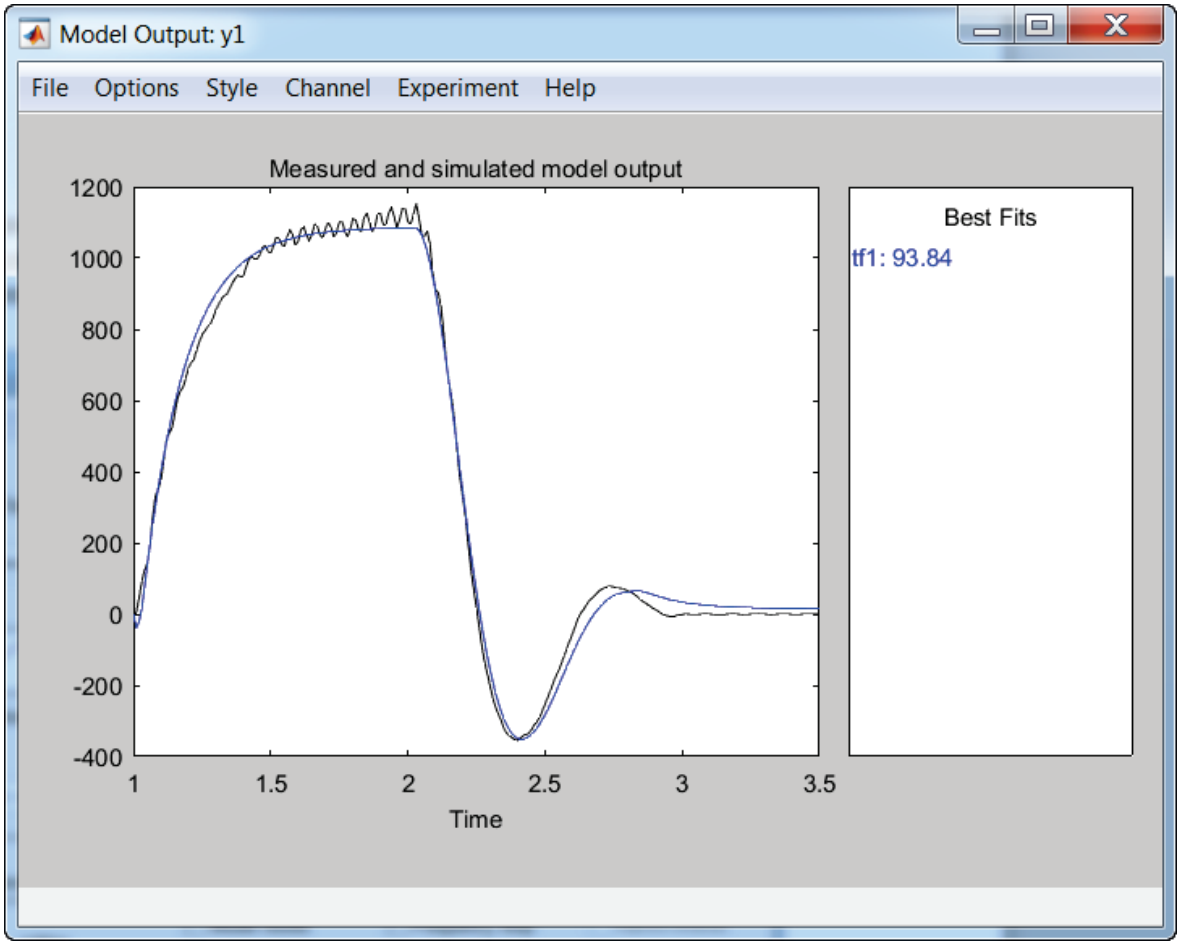


Figure 352 : superposition des données issues de l'expérimentation et du modèle issu de l'identification

C. Utilisation de la méthode en utilisant les lignes de commande

Pour identifier la réponse du système en utilisant les lignes de commande, il faut dans un premier temps créer un objet de type *iddata* contenant tous les éléments nécessaire à l'identification :

- Le vecteur contenant la réponse du système
- Le vecteur contenant l'entrée du système
- La période d'échantillonnage des données

Taper la commande suivante :

```
>> mydata=iddata(v,u,0.01)
mydata =

Time domain data set with 251 samples.
Sample time: 0.01 seconds

Outputs     Unit (if specified)
  y1

Inputs      Unit (if specified)
  u1
```

Il suffit maintenant de demander une identification par une fonction de transfert en spécifiant le nombre de pôle et le nombre de zéros de la fonction de transfert. On utilise pour cela la commande *tftest* en spécifiant 2 pôles et 0 zéro comme cela a été fait en utilisant la toolbox identification.

```
>>H = tfest(mydata,2,0)

H =

 From input "u1" to output "y1":
      5.746e05
 ----------------------
 s^2 + 3218 s + 2.119e04

Continuous-time identified transfer function.

Parameterization:
  Number of poles: 2   Number of zeros: 0
  Number of free coefficients: 3
  Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 93.84% (simulation focus)
FPE: 1040, MSE: 999
```

MATLAB renvoie alors la fonction de transfert du second ordre, identique à celle obtenue en utilisant la toolbox identifiaction.

Commandes utiles pour l'identification	
Fonctions	Commandes
Création d'un objet *iddata* pour l'identifiaction	mydata=iddata(y,u,Ts) y : vecteur contenant les données de la sortie u : vecteur contenant les données de l'entrée Ts : période d'échantillonnage des données
Identification par une fonction de transfert de la réponse d'un système	H=tftest(mydata,np,nz) mydata : objet créé à l'aide de la fonction iddata np : nombre de pôles souhaités pour la fonction de transfert nz : nombre de zéros souhaités pour la fonction de transfert

Il existe également de nombreuses commandes permettant de préparer les données pour l'identification (filtrage, réglages de l'offset, remplacement des données manquantes…). Pour plus d'informations sur ces commandes, vous pouvez consulter l'aide de la **System Identification Toolbox**.

Chapitre 8 : Le contrôle commande avec MATLAB - Simulink

I. Introduction

MATLAB – Simulink possède de très puissants outils de contrôle commande des systèmes asservis. Le logiciel peut réaliser automatiquement le réglage d'un correcteur PID dans un environnement ou l'utilisateur agit sur les paramètres de réglages sans qu'il lui soit nécessaire d'avoir recours à des bases théoriques. Le logiciel propose également des modes experts ou toutes les méthodes classiques du contrôle commande sont exploitées et peuvent être mises en œuvre. Le principal avantage est de pouvoir disposer de tous les tracés fréquentiels et temporels nécessaires au processus de synthèse d'un correcteur. Ces tracés évoluent de manière dynamique en fonction de la structure du correcteur imposée par l'utilisateur.

II. Réglages automatique d'un PID

Pour illustrer la démarche de contrôle commande avec MATLAB et Simulink, nous allons étudier l'asservissement en vitesse d'un moteur à courant continu.

A. Modélisation

Les équations électrique, mécanique et de couplage d'un moteur à courant continu sont données

$$\begin{cases} u(t) = e(t) + R\,i(t) + L\dfrac{d\,i(t)}{dt} \\ e(t) = K_e\,\omega_m(t) \\ J\dfrac{d\,\omega_m(t)}{dt} = C_m(t) - C_r(t) - f\,\omega_m(t) \\ C_m(t) = K_m\,i(t) \end{cases}$$

$u(t)$: tension de commande du moteur (V)

$\mathrm{e}(\mathrm{t})$: force contre électromotrice du moteur (V)

$i(t)$: intensité dans le moteur (A)

$C_m(t)$: couple exercé par le moteur $(\mathrm{N.m})$

$C_r(t)$: couple résistant ramené sur l'arbre moteur $(\mathrm{N.m})$

$\omega_m(t)$: vitesse de rotation du moteur $(\mathrm{rad/s})$

R : résistance de l'induit (Ω)

L : inductance de l'induit (H)

J : inertie équivalente ramenée sur l'arbre moteur $(\mathrm{kg.m^2})$

f : paramètre de frottemement visqueux $(\mathrm{N.m.s})$

$\mathrm{K_t}$: constante de couple $(\mathrm{N.m/A})$

$\mathrm{K_e}$: coefficient de force contre électromotrice $(\mathrm{V.s/rad})$

Figure 353 : modélisation du moteur à courant continu

Après avoir appliqué la transformée de Laplace à ces équations on obtient le schéma bloc du moteur à courant continu. On asservit ce moteur en vitesse selon le schéma bloc de la Figure 354.

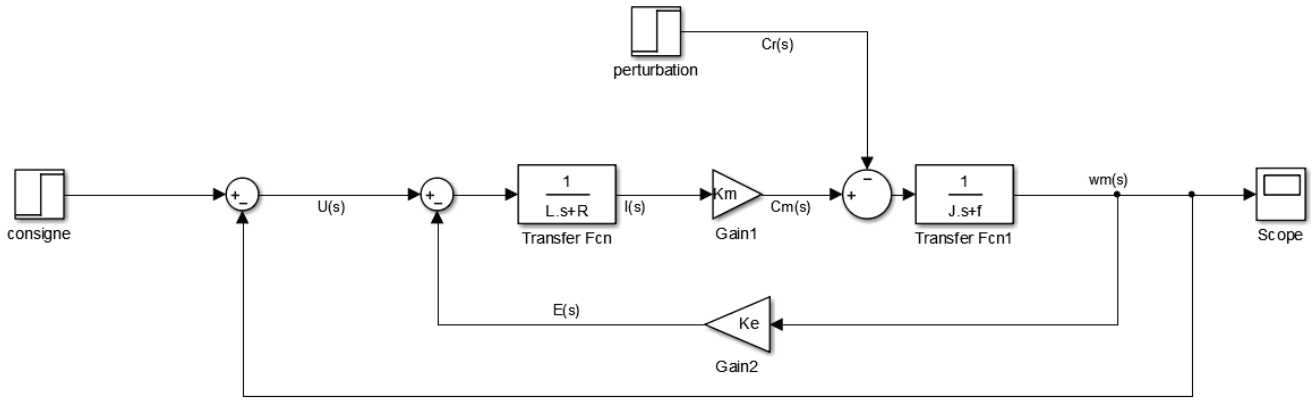


Figure 354 : asservissement en vitesse d'un moteur à courant continu

B. Ouverture du modèle

Ouvrir le fichier ***moteur_cc_ass_vit.slx***.
Ouvrir le script ***parametres_moteur_cc.m*** et éxécuter-le.

Le fichier contient la modélisation de l'asservissement du moteur à courant continu de la Figure 353.
Lancer la simulation et visualiser la réponse dans le scope.

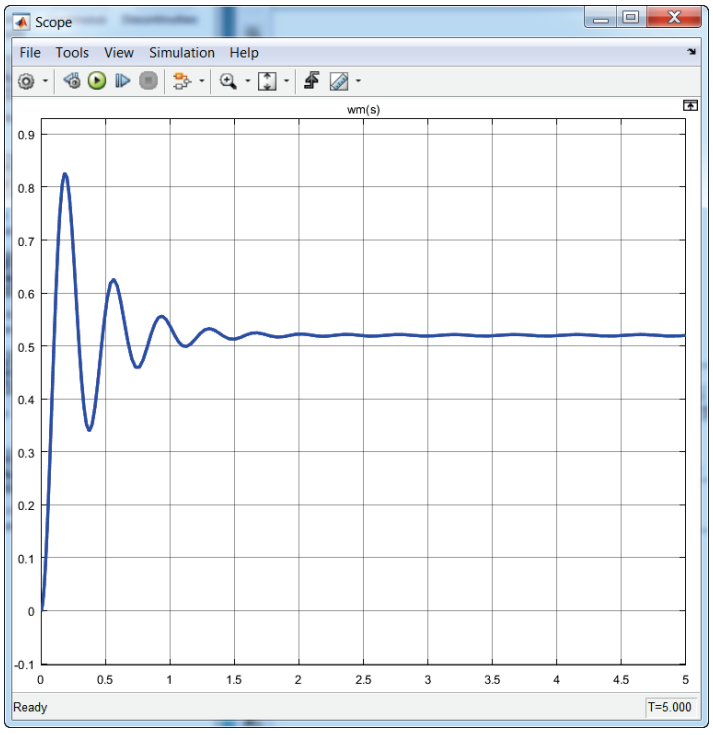


Figure 355 : réponse temporelle du système non corrigé à un échelon unitaire

Pour effecteur le réglage automatique d'un PID, il faut **insérer** le bloc **PID controller** conformément à la Figure 356.

Fonction du composant	Représentation	Bibliothèque
Correcteur PID	PID(s) PID Controller	Simulink/Continues

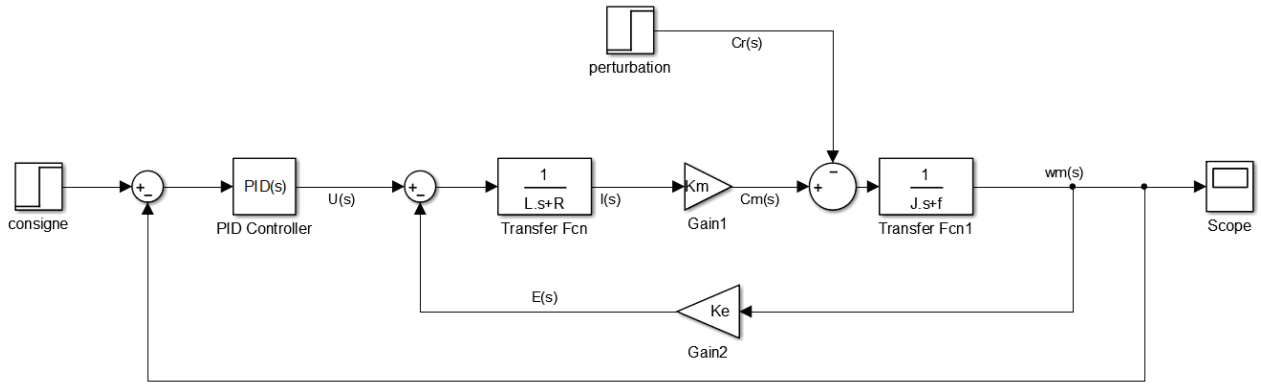


Figure 356 : ajout d'un bloc PID controller dans l'asservissement

Double cliquer sur le bloc **PID Controller** pour ouvrir la fenêtre de paramétrage de la Figure 357.

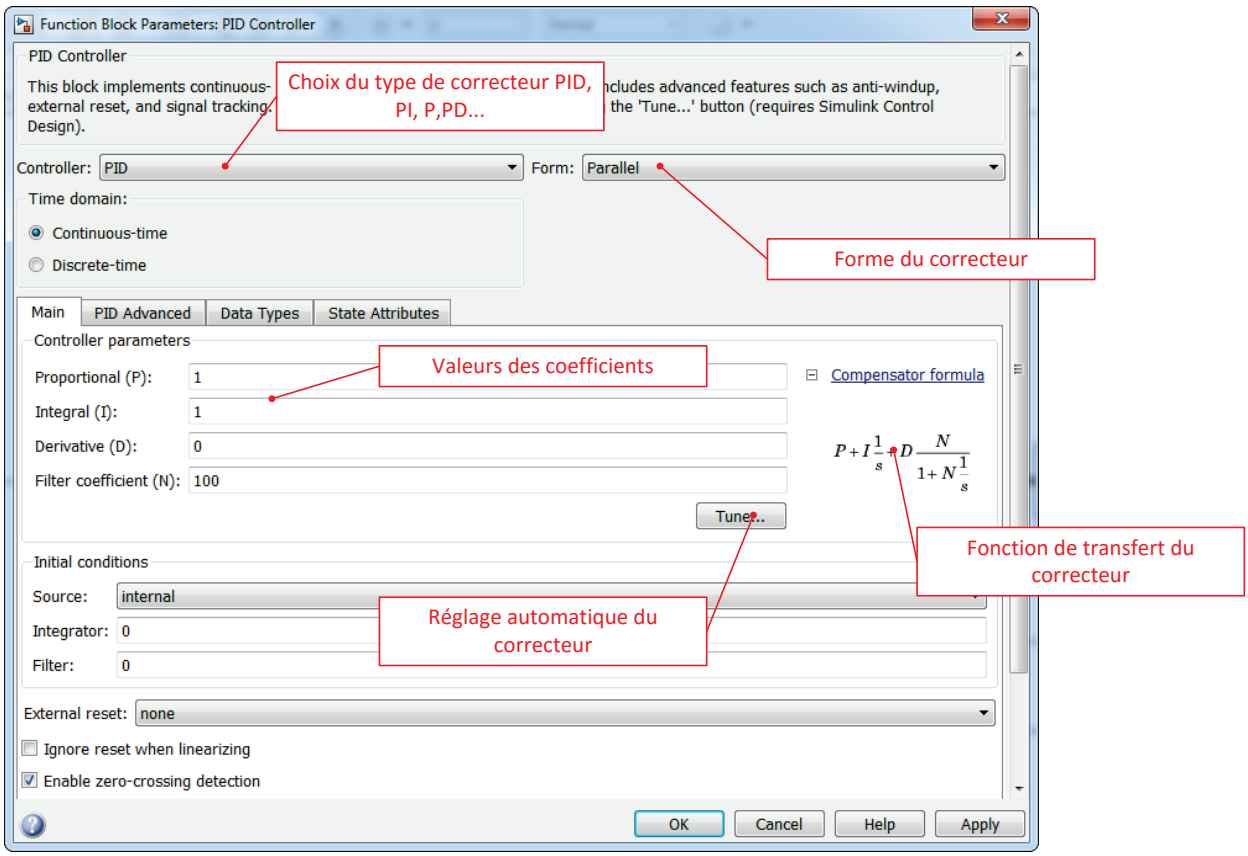


Figure 357 : fenêtre de paramétrage du PID Controller

1. Analyse de la réponse temporelle

Cliquer sur **Tune** pour ouvrir la fenêtre de réglage des performances du système.
MATLAB propose un réglage réalisant un compromis entre tous les critères de performance du système comme le montre la Figure 358. Par défaut la réponse indicielle de la fonction de transfert en boucle fermée (reference tracking) s'affiche pour le système corrigé et pour le système non corrigé.

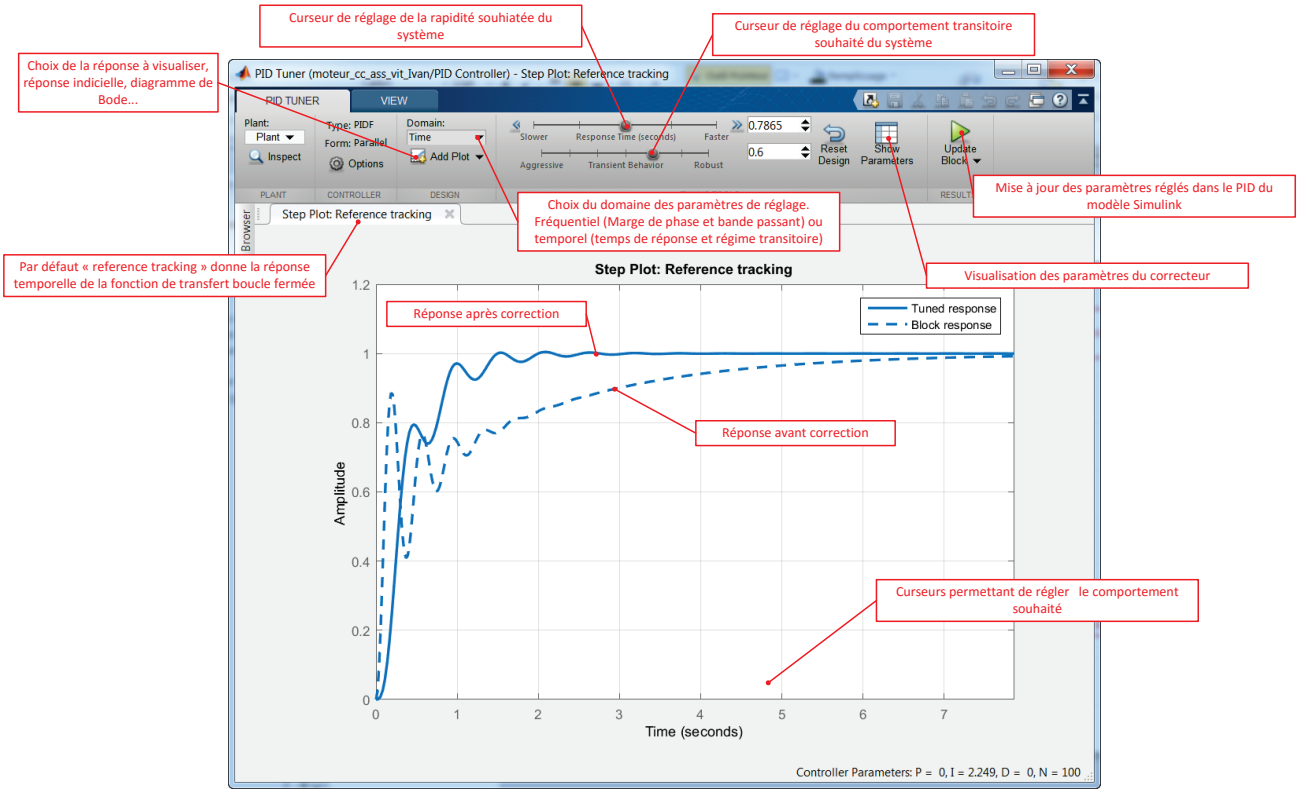


Figure 358 : fenêtre de réglage des performances du système

Il est cependant possible d'agir manuellement sur les curseurs pour modifier les performances du système et le rendre conforme à cahier un cahier des charges précis. Afin de disposer de toutes les informations nécessaires pour faire les réglages l'outil propose différentes fonctionnalités.

Cliquer sur **Show Parameters** en haut à droite de la fenêtre pour visualiser les critères de performances et les valeurs des gains du correcteur avant et après la correction comme le montre la Figure 359.

Controller Parameters

	Tuned	Block
P	0	1
I	2.2488	1
D	0	0
N	100	100

Performance and Robustness

	Tuned	Block
Rise time	0.719 seconds	2.95 seconds
Settling time	1.85 seconds	6.05 seconds
Overshoot	0.466 %	0 %
Peak	1	0.999
Gain margin	7.32 dB @ 11.8 rad/s	Inf dB @ Inf rad/s
Phase margin	83.8 deg @ 2.54 rad/s	33.7 deg @ 16.1 rad/s
Closed-loop stability	Stable	Stable

Figure 359 : les critères de performances du système et les valeurs de gain du correcteur

Pour obtenir le **diagramme de Bode la fonction de transfert en boucle ouverte**, sélectionner le menu **Add Plots/Bode/Open Loop.**

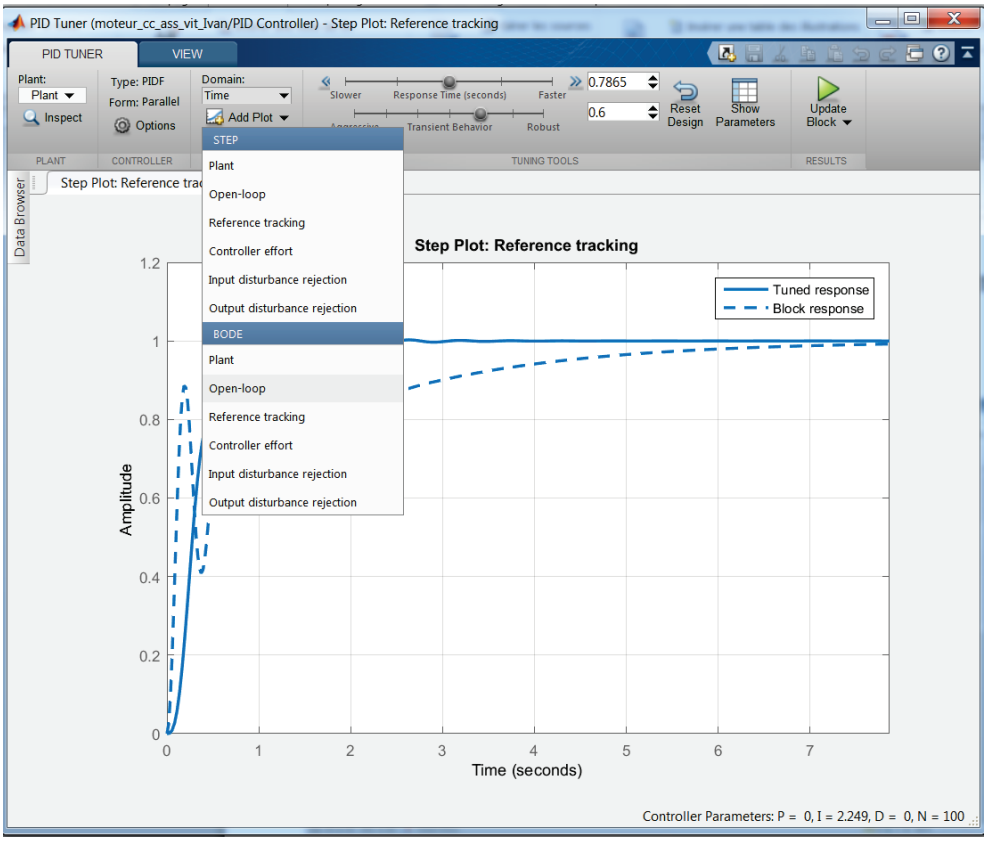


Figure 360 : ajout du diagramme de Bode de la fonction de transfert en boucle ouverte

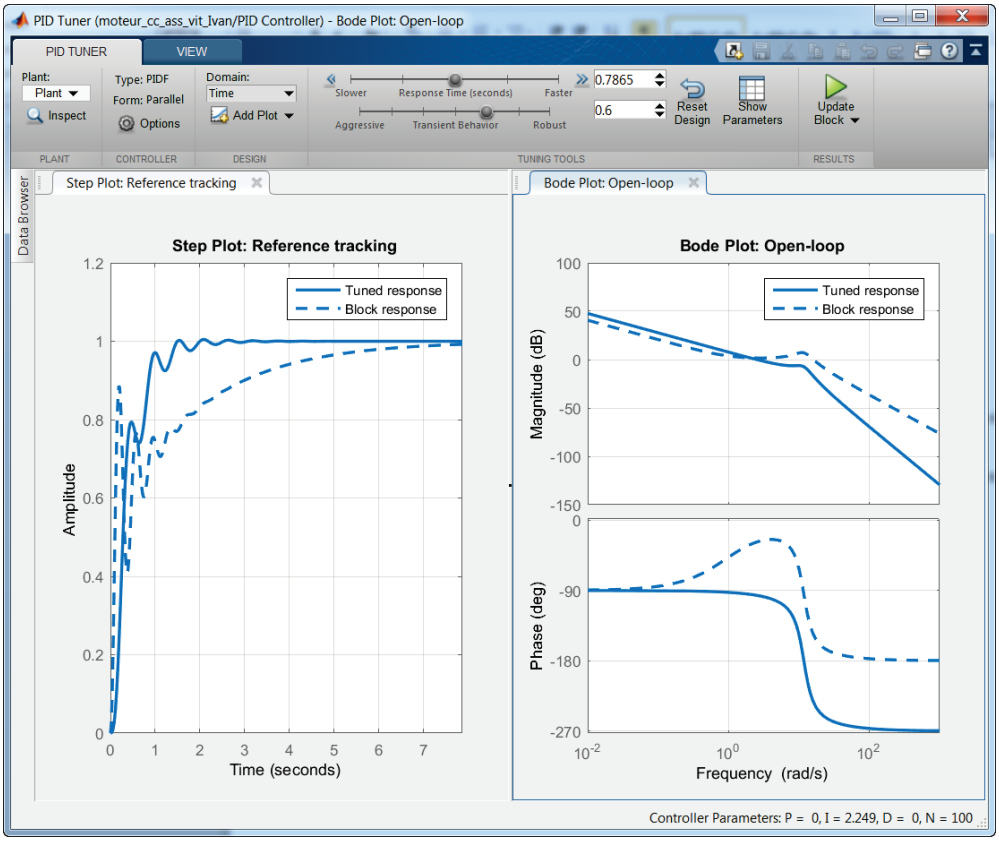


Figure 361 : visualisation du diagramme de Bode de la fonction de transfert en boucle ouverte pour le réglage du PID

Les diagrammes de Bode du système corrigé et non- corrigé apparaissent sur une seconde fenêtre.

Pour régler les options des critères de performances, **cliquer droit** dans la fenêtre graphique de la réponse temporelle puis sélectionner **Properties**, puis onglet **Options**

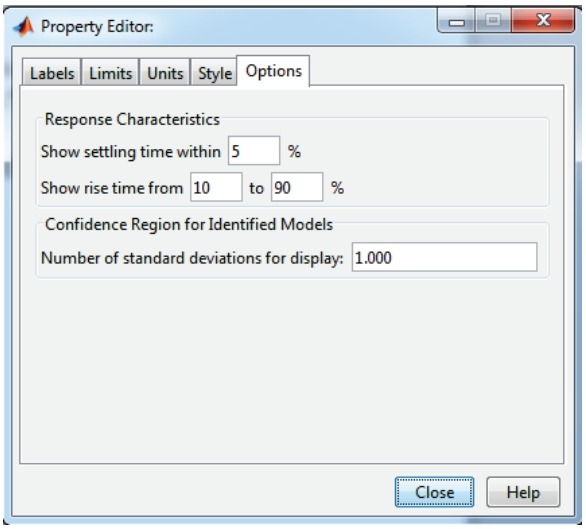


Figure 362 : réglage des options des critères de performances

Indiquer que le temps de réponse est évalué sur le critère du temps de réponse à 5% en indiquant **5**% dans le champ **Show settling time within**. Le critère **rise time** concerne le temps de montée qui, ici, est évalué entre 10% et 90% de la valeur finale atteinte. Cliquer sur **Close**.

Il est également possible de visualiser graphiquement tous les critères de performances.

Cliquer droit dans la fenêtre graphique de la réponse temporelle puis sélectionner **Characteristics/Peak Response** (dépassement), puis **Characteristics/Settling Time** (temps de réponse à 5%), puis **Characteristics/Rise Time** (temps de montée) puis **Characteristics/Steady State** (précision).

Cliquer droit dans la fenêtre graphique du diagramme de Bode, puis sélectionner **Characteristics/All Stability Margins** (marges de gain et marge de phase).

Il est maintenant possible de visualiser les critères de performances dans la fenêtre graphique.

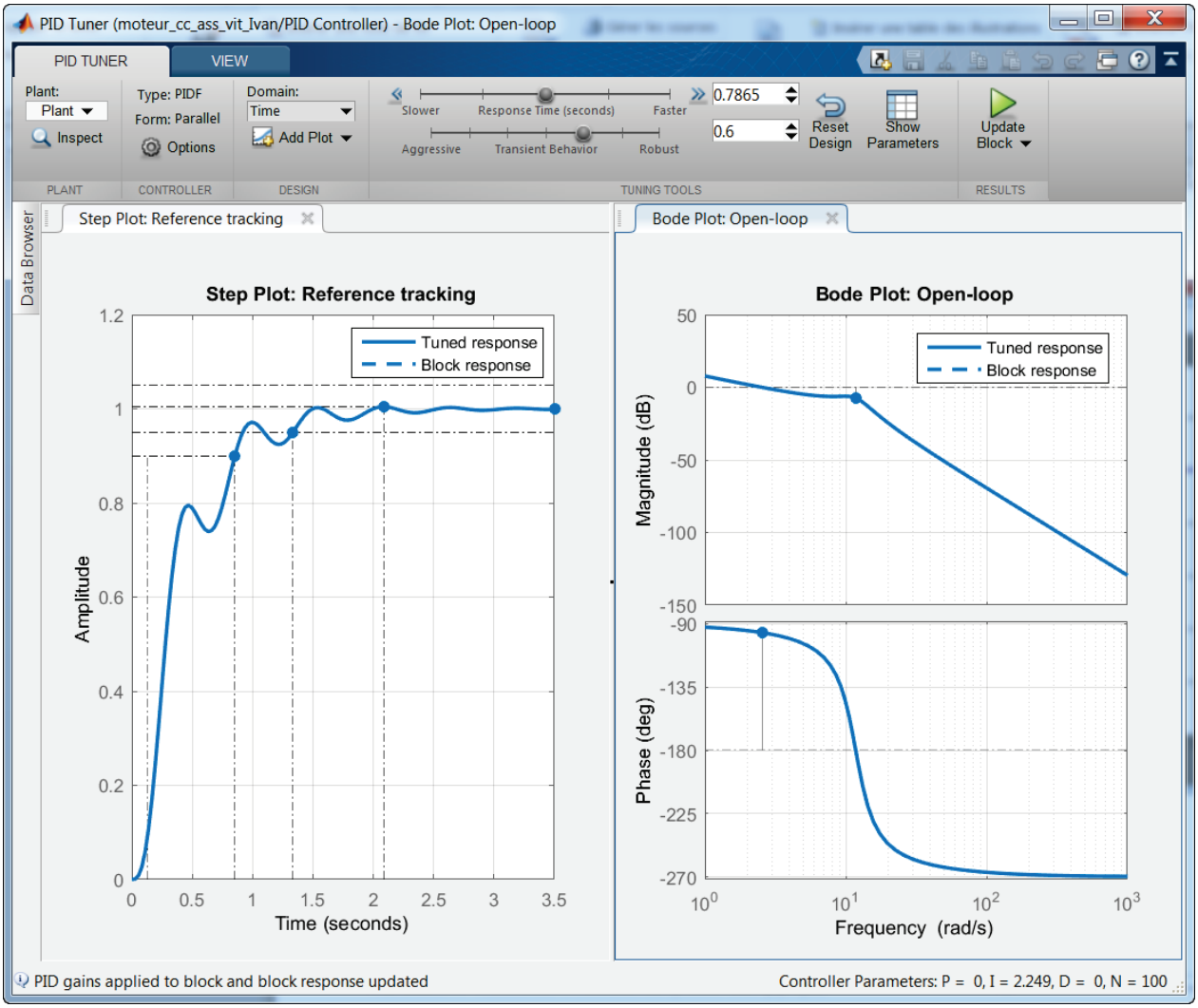


Figure 363 : visualisation des critères de performances

A ce stade, il est possible d'agir sur les curseurs de réglage et de voir évoluer de manière interactive les courbes et les paramètres.

Une fois le réglage effectué, **Cocher** la case **Update Block** pour mettre à jour automatiquement le bloc **PID Controller** de **Simulink**.

2. Importation dans Simulink

Une fois que la réponse du système et bien réglée, cliquer sur OK pour retourner dans le bloc de paramétrage du PID. Nous pouvons constater que les gains du correcteur ont été actualisés avec les valeurs correspondant au réglage qui vient d'être effectué.

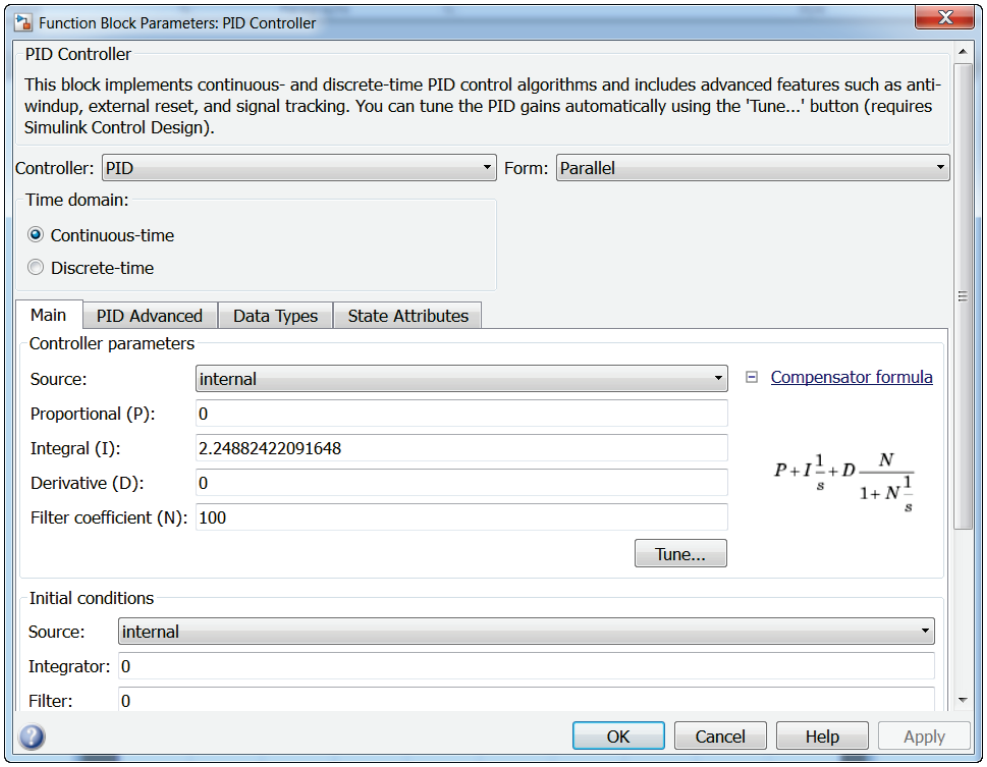


Figure 364 : actualisation automatique des paramètres du PID

Cliquer sur **OK**.
Lancer la simulation et visualiser la réponse corrigée dans le scope.

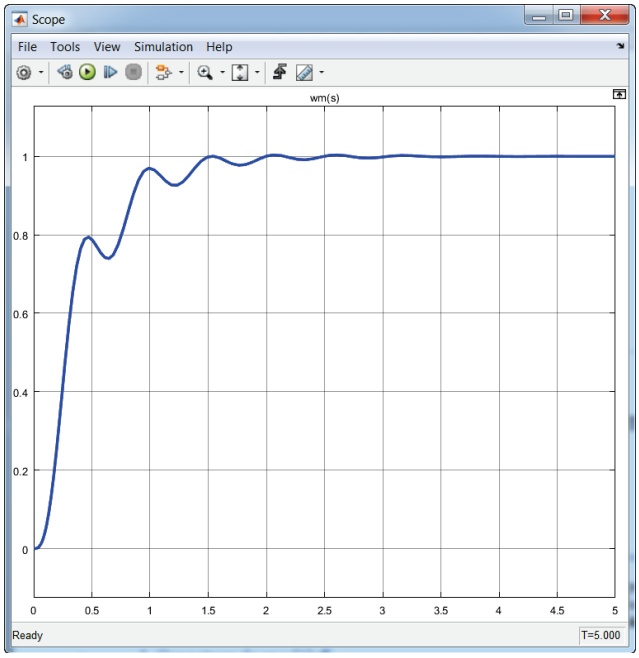


Figure 365 : réponse du système après réglage du PID

III. Réglage manuel d'un PID avec l'outil « compensator design »

Il est également possible de régler un PID tout en visualisant l'influence des réglages sur les réponses temporelles et fréquentielles de la fonction de transfert en boucle ouverte et de la fonction de transfert en boucle fermée. Pour cela, il faut utiliser l'outil « compensator design » de **Simulink**.

A. Ouverture du modèle

Ouvrir le fichier ***moteur_cc_ass_vit_PID_comp_des.slx.***

Ce fichier contient le modèle du moteur à courant continu asservi en vitesse avec un PID qui n'a pas encore été réglé (Gain proportionnel à 1 et les autres gains sont nuls).

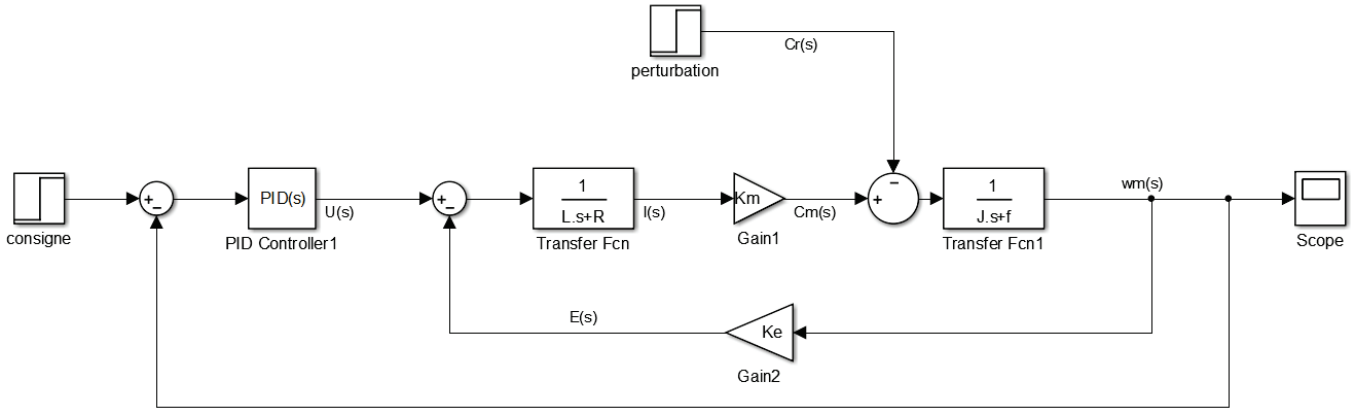


Figure 366 : modèle du moteur à courant continu asservi en vitesse avec perturbation

La consigne imposée est de 100 rad/s et une perturbation de couple est appliquée au système à l'instant t=3 s.

Les critères de performances sont les suivants :

- écart statique nul
- temps de montée (de 0 à 90%) de 0.8 s
- temps de réponse à 5% de 1 s
- dépassement maxi de 10%
- Marge de gain : 20 dB
- Marge de phase 45°
- rejet des perturbations en régime permanent

Lancer la simulation et observer la réponse du système dans le scope.

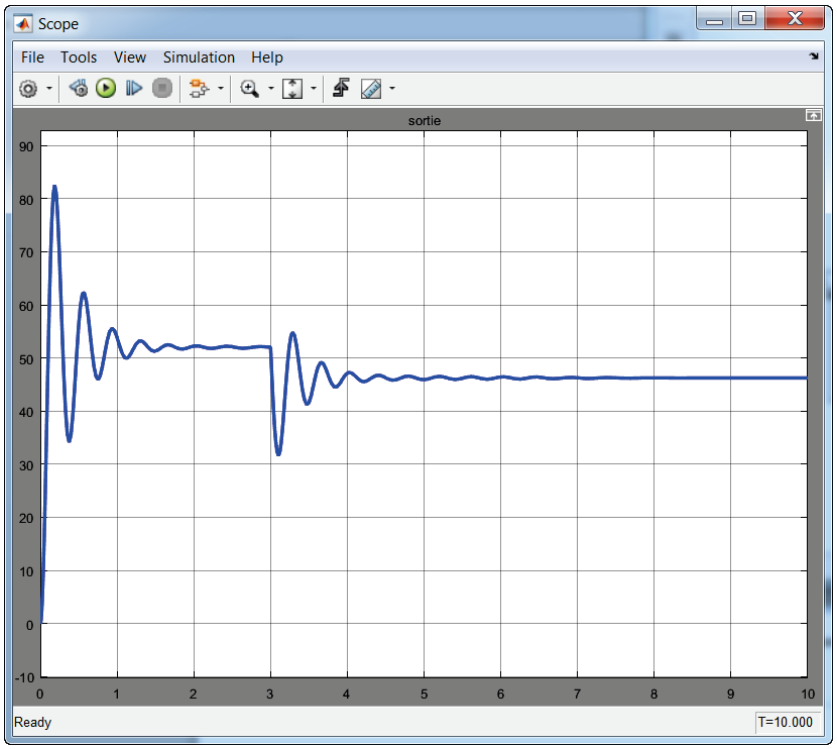


Figure 367 : réponse temporelle du système non corrigé

Nous pouvons constater que la réponse du système n'est pas satisfaisante :

Le système n'est pas précis, mal amorti et ne rejette pas correctement la perturbation.

B. Réglage du PID

Afin de pouvoir utiliser l'outil « compensator design », il est nécessaire de placer des points de linéarisation sur notre modèle. Il faut indiquer l'entrée et la sortie de la **fonction de transfert boucle fermée** pour que le « compensator design » puisse être utilisé. **MATLAB** détectera automatiquement la **fonction de transfert en boucle ouverte** et pourra tracer les lieux de transfert en boucle ouverte utiles pour effectuer les réglages.

1. Placement des points de linéarisation

Placer les points de linéarisation suivants (Le placement des points de linéarisation est décrit en détail page 234):

- un point de linéarisation de type **Input Perturbation** sur le signal qui représente la consigne de vitesse
- un point de linéarisation de type **Input Perturbation** sur le signal qui représente la perturbation
- un point de linéarisation de type **Output Measurement** sur le signal qui représente la vitesse de sortie du moteur

Vous devez obtenir le modèle de la Figure 368.

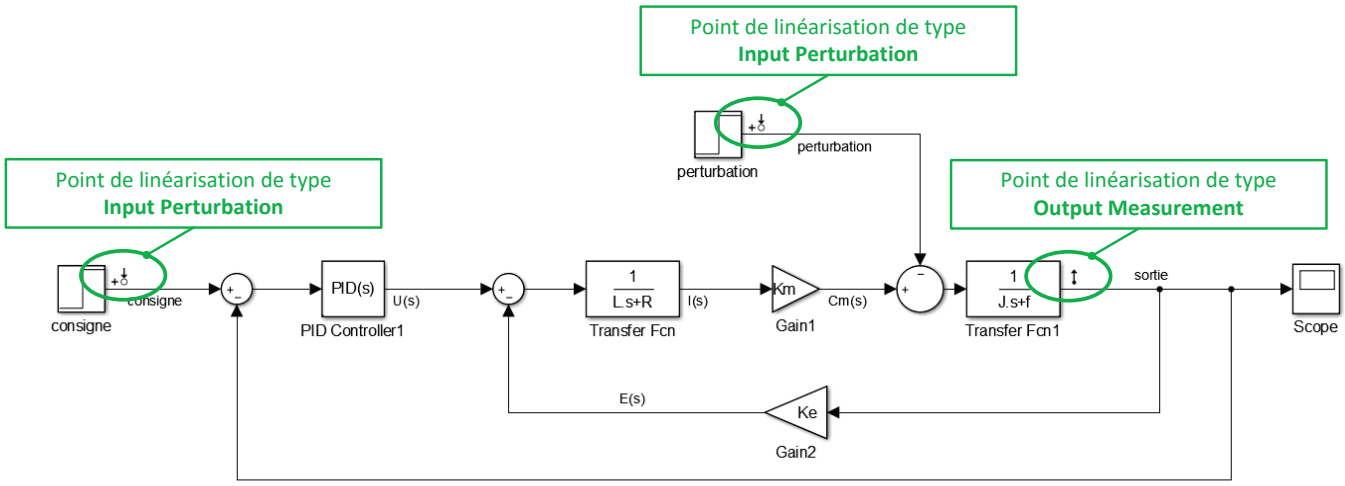


Figure 368 : placement des points de linéarisation sur le modèle

A partir de la barre de commande, sélectionner **Analysis/Control Design/Control System Designer**

Cette commande entraîne l'ouverture de la fenêtre **Control and Estimation Tools Manager** qui propose de choisir les blocs que l'on veut régler, de visualiser les points d'entrée et de sortie de la boucle fermée.

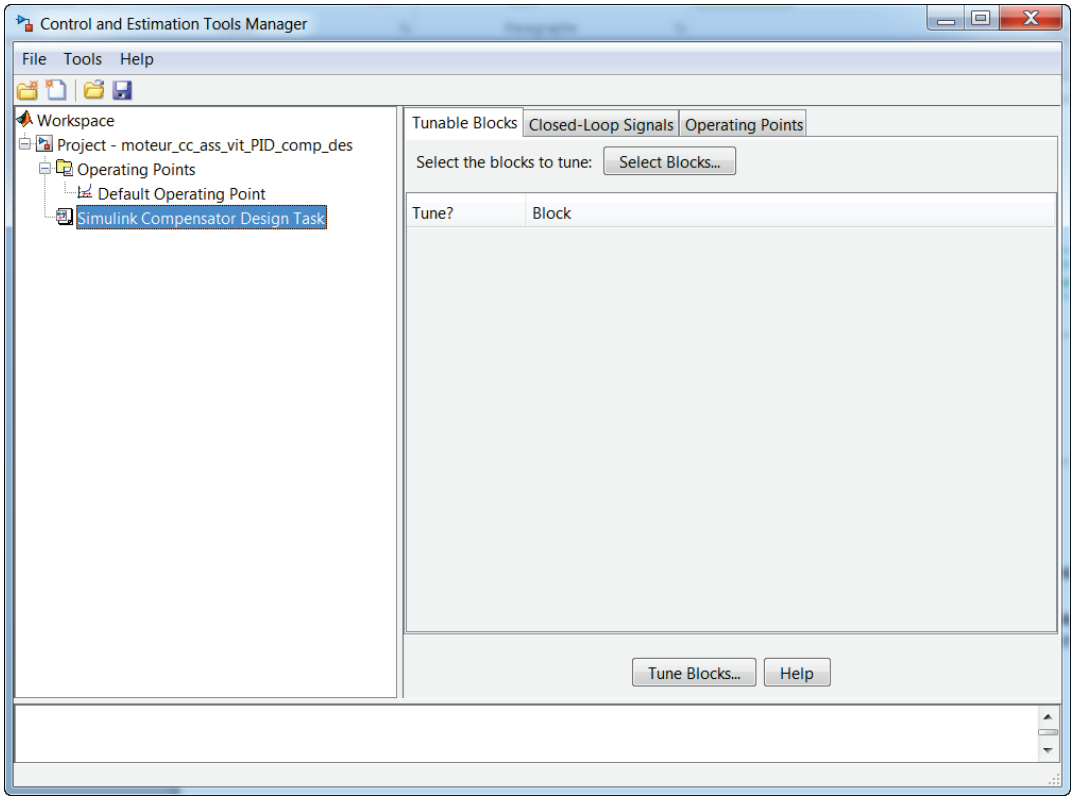


Figure 369 : fenêtre Control and Estimation Tools Manager

Sélectionner l'onglet **Closed-Loop signal** afin de visualiser les signaux d'entrée et de sortie pris en compte.

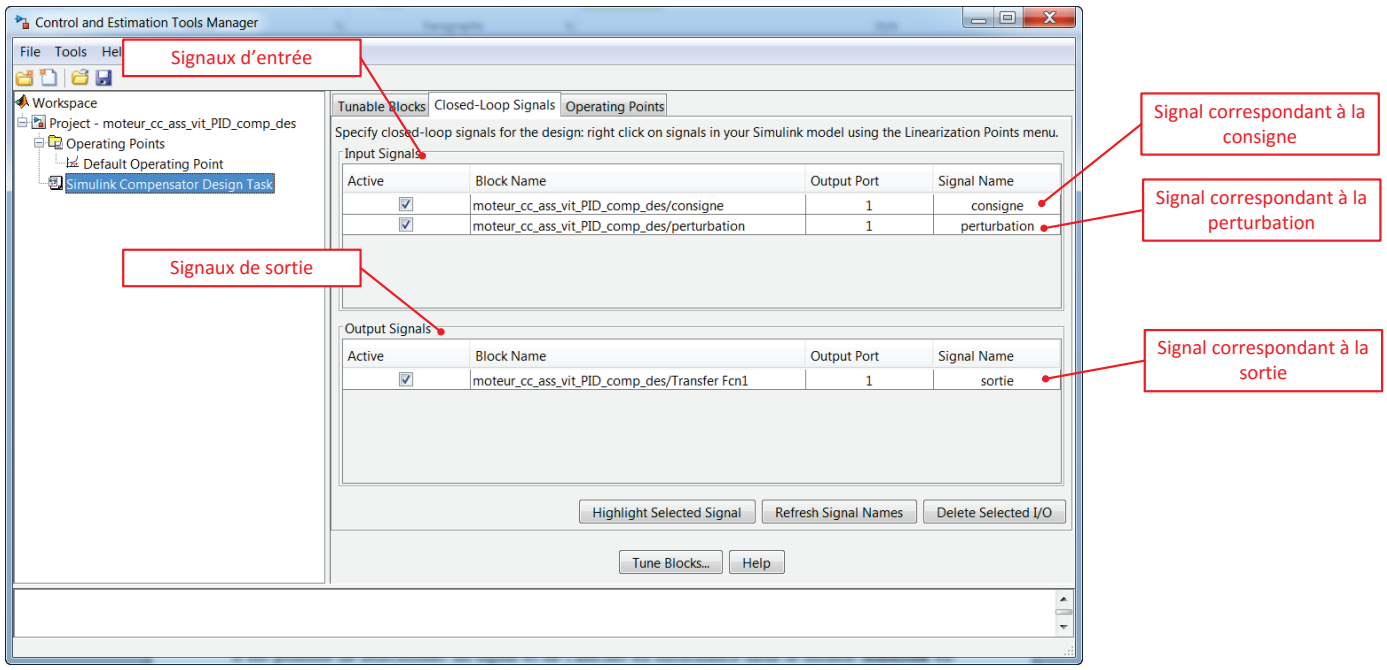


Figure 370 : visualisation des signaux d'entrée et de sortie pris en compte pour la boucle fermée

Il sera ensuite possible de construire toutes les fonctions transferts possibles à partir de ces signaux.
Il est possible de sélectionner un signal et de l'afficher en surbrillance dans le modèle **Simulink** en cliquant sur **Highlight Selected Signal**, ce qui permet de contrôler les points d'entrée et de sortie de la boucle fermée.

2. Choix du bloc à régler

Sélectionner l'onglet **Tunable Blocks** pour choisir le bloc à régler puis cliquer sur **Select Blocks.**

La fenêtre **Select Blocks to Tune** s'ouvre et propose tous les blocs du modèle qui peuvent être réglés (les gains, les fonctions de transfert, les PID…).

Sélectionner **PID Controllers** et cliquer sur **OK** afin de pouvoir régler le PID.

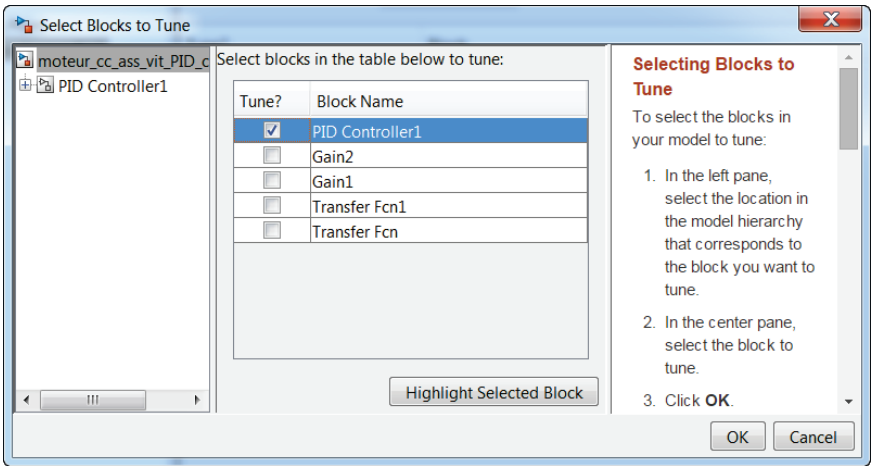


Figure 371 : choix du bloc à régler

Cliquer ensuite sur **Tune Block** pour commencer les réglages.

Une fenêtre d'informations s'ouvre en vous indiquant que vous allez ouvrir l'outil **Siso Design Tool.** Cette fenêtre vous indique que vous allez travailler dans cet outil avec deux types de tracés :

Design plots : ces tracés de type lieu des pôles, diagramme de Bode ou diagramme de Black-Nichols de la fonction de transfert en boucle ouverte sont interactifs dans l'outil. Ils seront utilisés pour régler graphiquement les performances du système en utilisant les méthodes de réglage basées, soit sur l'analyse de la fonction de transfert en boucle ouverte, soit sur l'analyse de la fonction de transfert en boucle fermée. Sur ces tracés, il sera possible de modifier les gains du correcteur en déplaçant manuellement les lieux de transfert ou en ajoutant et supprimant des pôles et des zéros dans la fonction de transfert du correcteur…Les tracés que l'on souhaite voir apparaître sont définis au début de la phase d'utilisation de l'outil mais pourront être modifiés à tout moment durant l'étude.

Analysis plots : ces tracés de type réponse indicielle, diagramme de Bode de la fonction de transfert en boucle fermée…montrent les conséquences des réglages effectués sur les performances du système.

Pour résumer, il faut agir et modifier les **Design Plots** et pour observer les conséquences sur les **Analysis Plots.** Il est important avant de choisir les différents tracés de bien analyser la démarche que l'on souhaite mener et d'afficher uniquement les tracés utiles.

Cliquer sur **Next**.

3. Choix des tracés à visualiser pour la boucle ouverte

La fenêtre Design Configuration Wizard s'ouvre et vous permet de choisir les tracés de type **Design Plots** sur lesquels il sera possible d'agir pour effectuer les réglages. (SISOTOOL Design Views)

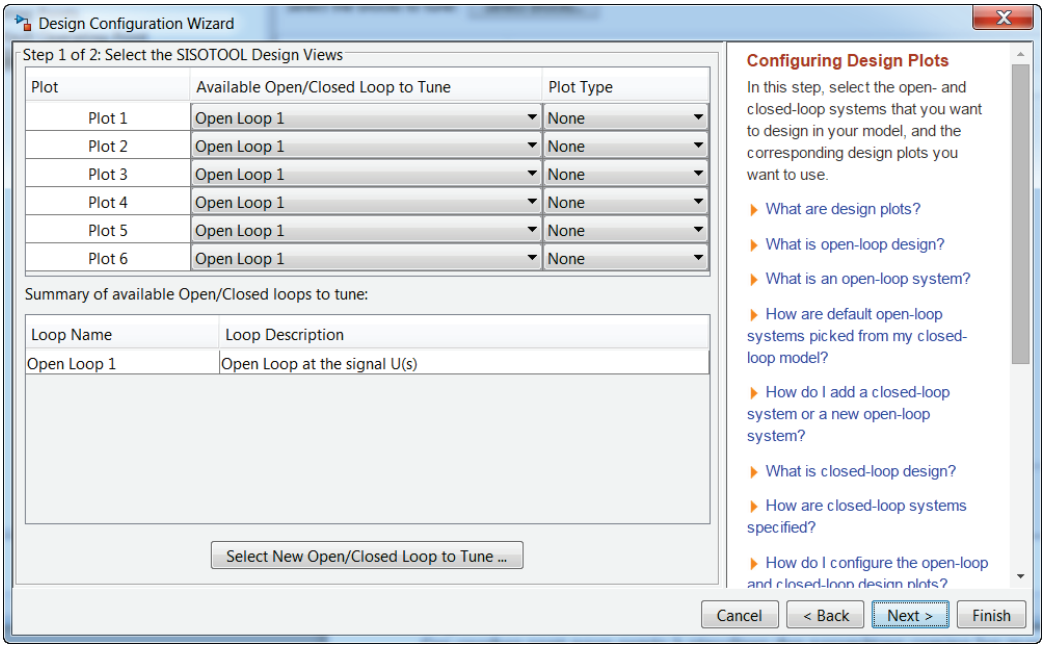


Figure 372 : la fenêtre Design Configuration WIzard

Nous pouvons choisir en utilisant les menus déroulants : (Figure 372)

- Root locus (lieu des pôles de la **fonction de transfert en boucle fermée**)
- Diagramme de Bode de la **fonction de transfert en boucle ouverte**
- Diagramme de Nichols de la **fonction de transfert en boucle ouverte**

A noter que dans cette fenêtre tous les tracés sont relatifs à la fonction de transfert en boucle ouverte. Même le lieu des pôles de la fonction de transfert en boucle fermée est déterminé à partir de la variation du gain de la fonction de transfert en boucle ouverte considérée (lieu d'Evans)
Ces courbes vont nous servir à visualiser des paramètres comme les marges de gain et de phase, la position des pôles de la fonction de transfert en boucle fermée qui donnera des informations sur la stabilité ou sur l'amortissement…
Nous aurons la possibilité d'agir sur ces tracés pour modifier les paramètres du correcteur et procéder aux réglages.
Configurer les tracés de type **Design Plots** comme indiqué sur la Figure 373.

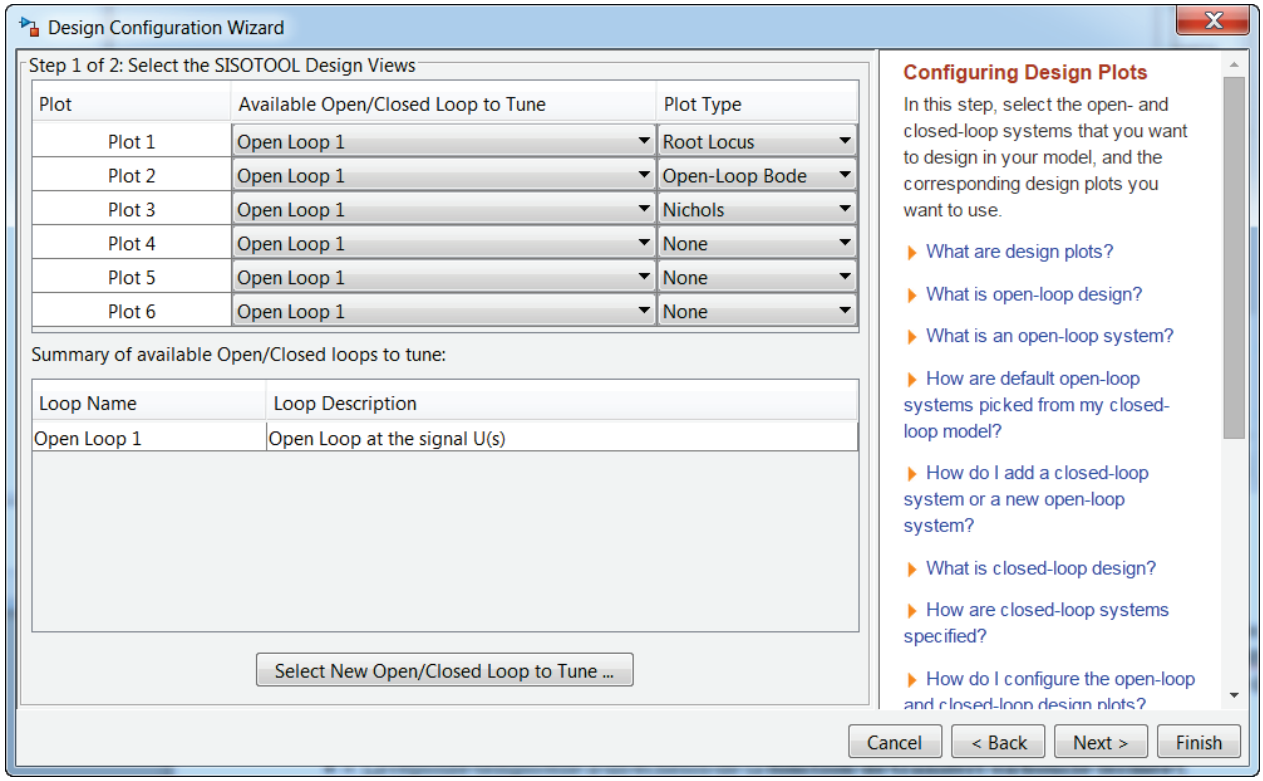


Figure 373 : choix des tracés à visualiser pour le réglage du PID

4. Choix des tracés pour visualiser les performances de la boucle fermée

La fenêtre suivante permet de visualiser les tracés de type **Analysis Plots** montrant les performances de la boucle fermée. Ces tracés seront les conséquences des réglages effectués sur les tracés de type **SISOTOOL Design Views** qui concernent la fonction de transfert en boucle ouverte. On ne pourra pas agir sur ces tracés et ils ne seront disponibles qu'à la visualisation.
Nous pouvons choisir en utilisant les menus déroulants : (Figure 374) différents types de diagrammes qui concernent tous le comportement de la fonction de transfert en boucle fermée :

- Réponse indicielle
- Réponse impulsionnelle
- Diagramme de Bode
- Diagramme de Black Nichols…

Dans notre cas nous nous contenterons des **Analysis Plots** suivants :

- La réponse temporelle de la **fonction de transfert en boucle fermée** à un échelon de **consigne**
- La réponse temporelle de la **fonction de transfert en boucle fermée** à un échelon de **perturbation**

Ce paramétrage correspond à la Figure 373.

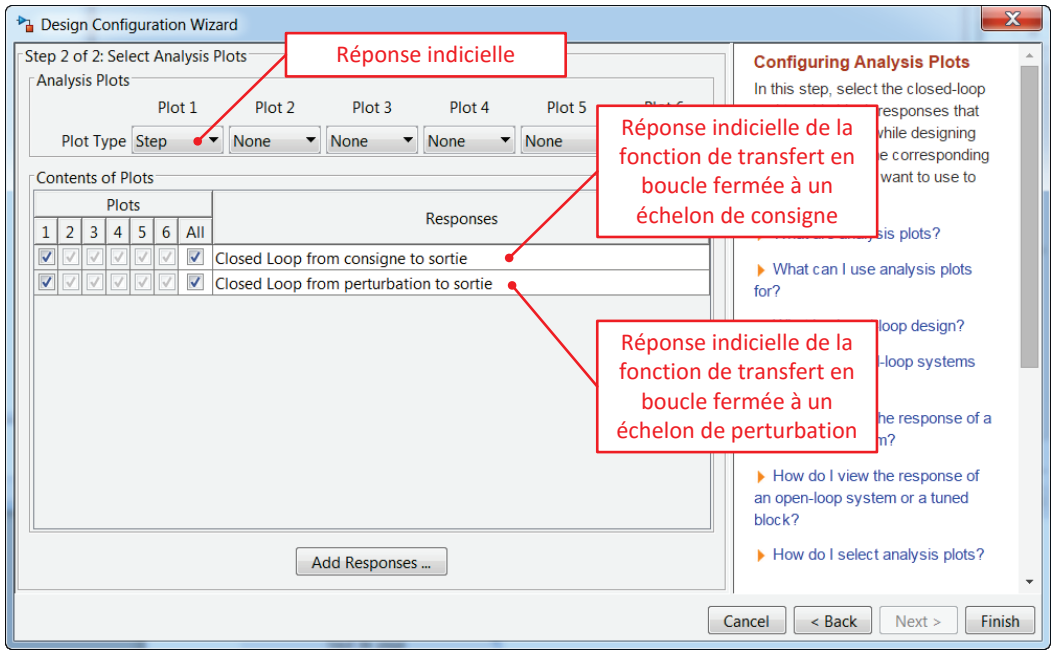


Figure 374 : choix des réponses de la boucle fermée

Cliquer sur **Finish**

5. Analyse des fenêtres graphiques de l'outil « Control System Designer »

Deux nouvelles fenêtres apparaissent alors à l'écran :

Linear Analysis for SISO Design Task : cette fenêtre permet de visualiser les réponses indicielles de la fonction de transfert en boucle fermée vis-à-vis de la consigne et vis-à-vis de la perturbation.

Pour obtenir le quadrillage du graphique, **Cliquer** avec le bouton droit de la souris dans la fenêtre graphique et sélectionner **Grid**.

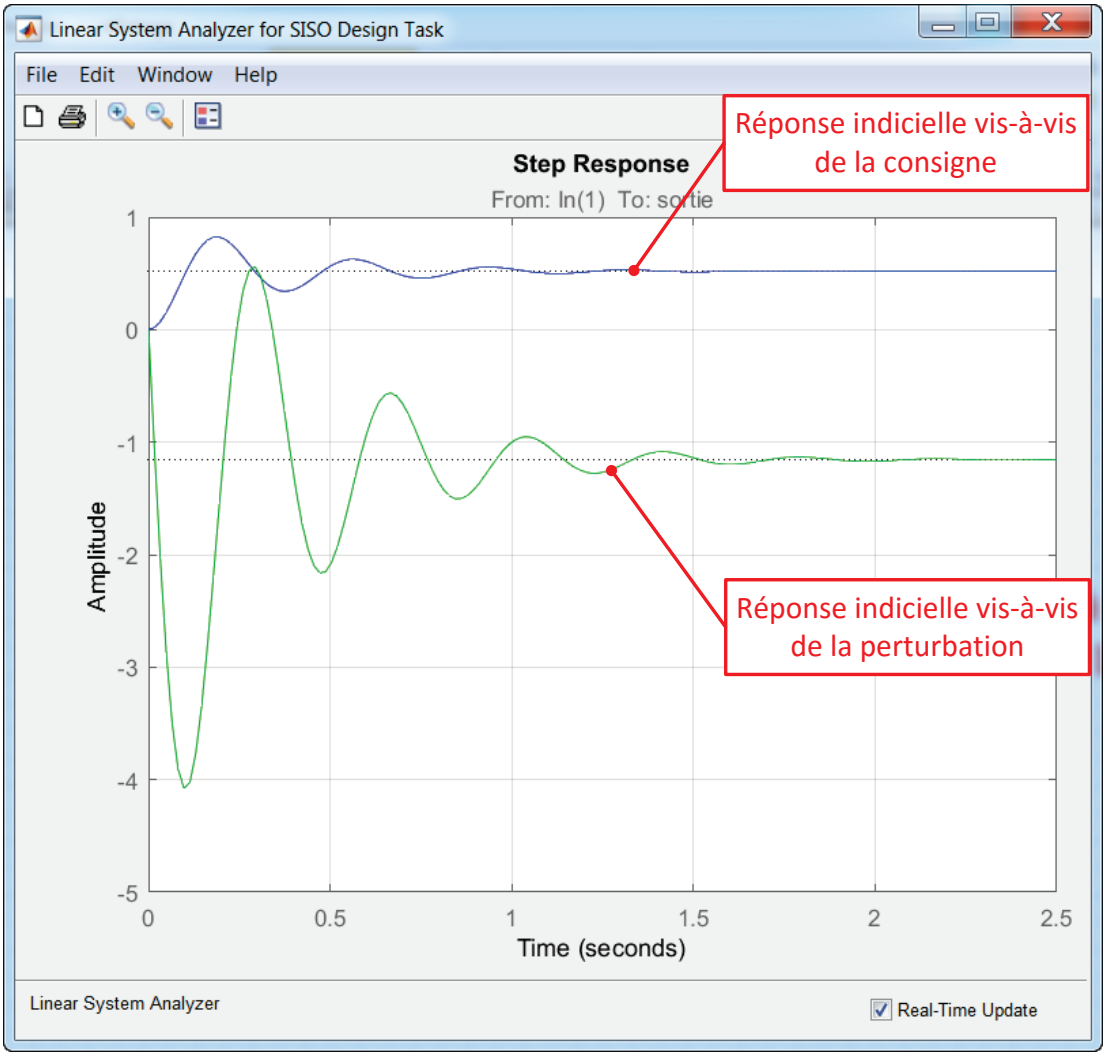


Figure 375 : LTI Viewer for SISO Design Task

SISO Design for SISO Design Task : cette fenêtre permet de visualiser les tracés sur lesquels nous allons agir pour effecteur le réglage du PID.

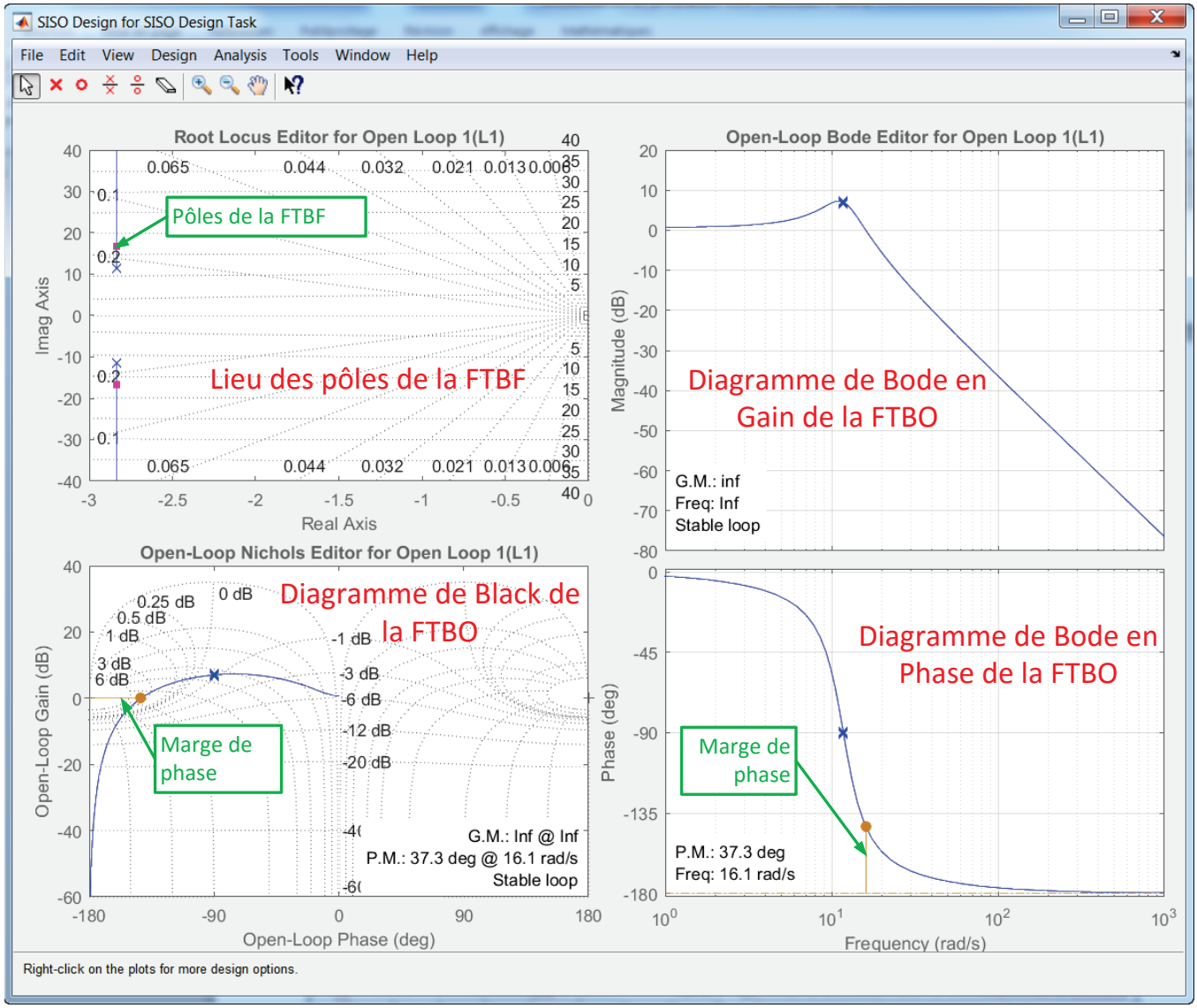


Figure 376 : SISO Design for SISO Design Task

A noter que la fenêtre **Control and Estimation Tool Manager** reste ouverte également. Elle permet à tout moment de revenir sur le choix des tracés en utilisant l'onglet **Graphical Tuning** pour modifier les tracés de la fenêtre **SISO Design for SISO Design Task** et l'onglet **Analysis Plots** pour modifier les tracés de la fenêtre **LTI View for SISO Design Task**.

Dans un premier temps les réglges seront effectués uniquement à l'aide de la réponse indicielle de la fonction de transfert en boucle fermée vis-à-vis de la consigne.
Dans la fenêtre **Control and Estimation Tool Manager** sélectionner l'onglet **Analysis Plots** et décocher la case correspondant à la réponse indicielle vis-à-vis de la perturbation comme indiqué sur la

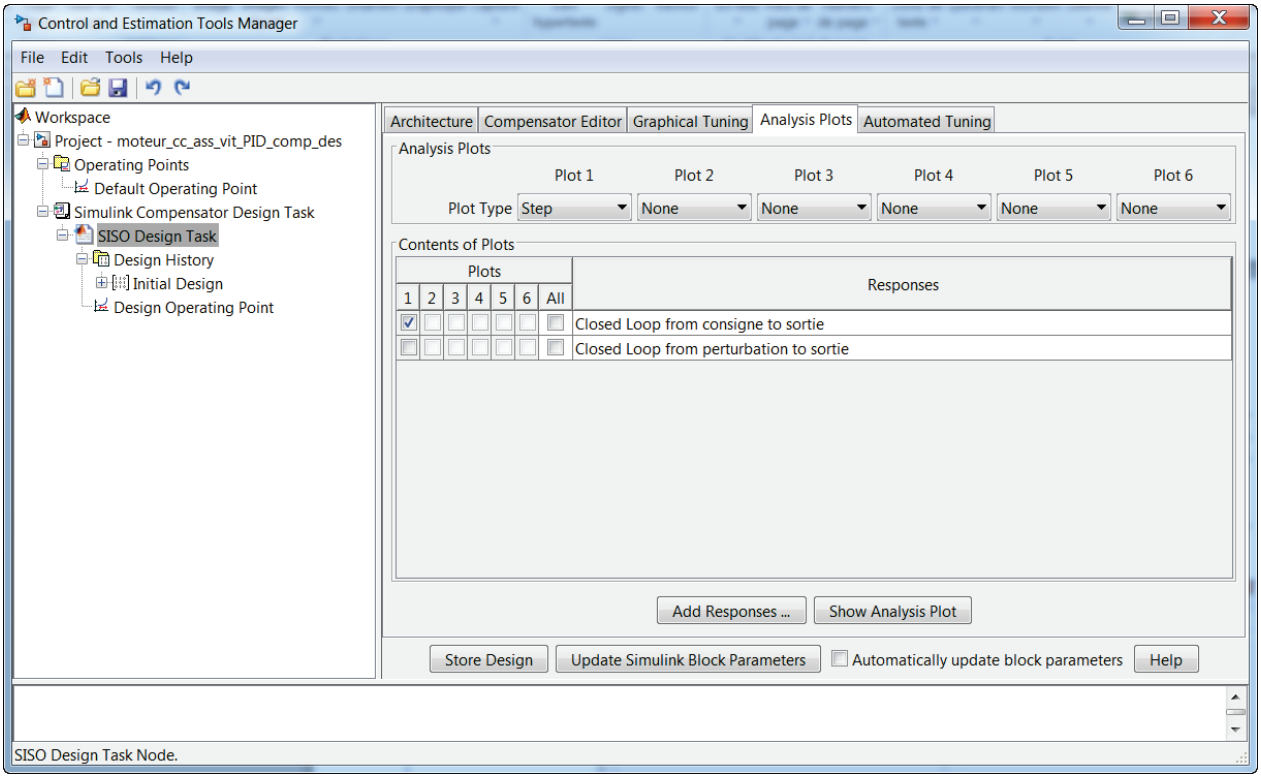


Figure 377 : modification du choix des Analysis Plots

La fenêtre **Linear Analysis for SISO Design Task** est mise à jour conformément à ce nouveau paramétrage (

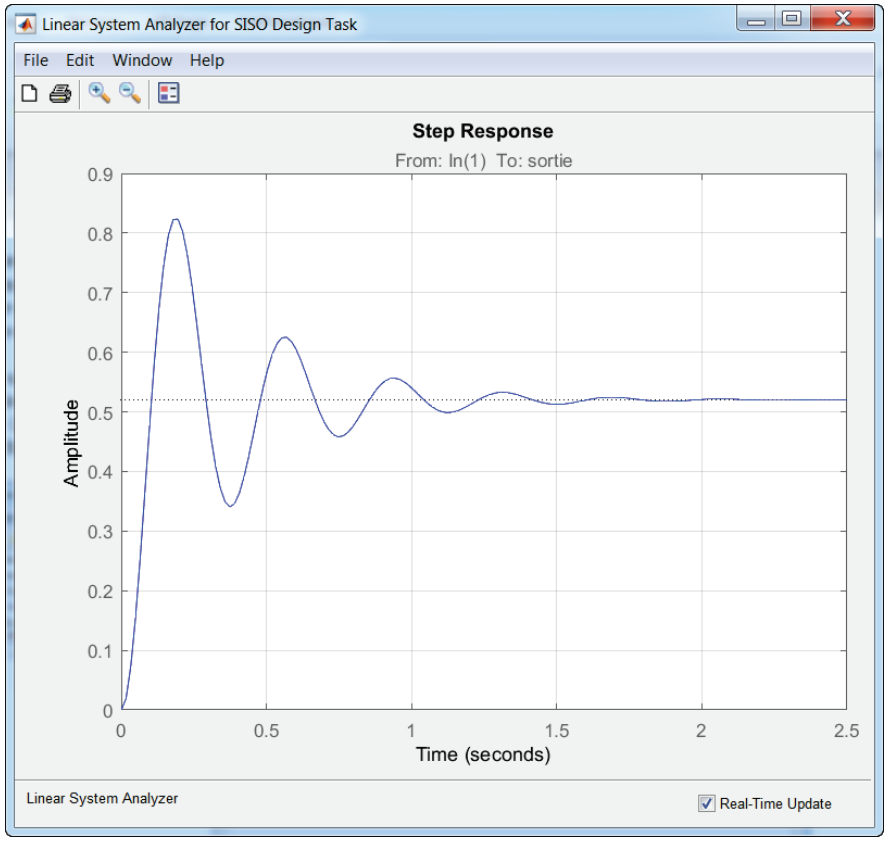


Figure 378 : mise à jour de la fenêtre Linear Analysis for SISO Design Task

Dans la fenêtre **Control and Estimation Tool Manager** sélectionner l'onglet **Compensator Editor** (Figure 379). Cette fenêtre permettra de visualiser et régler les paramètres du PID

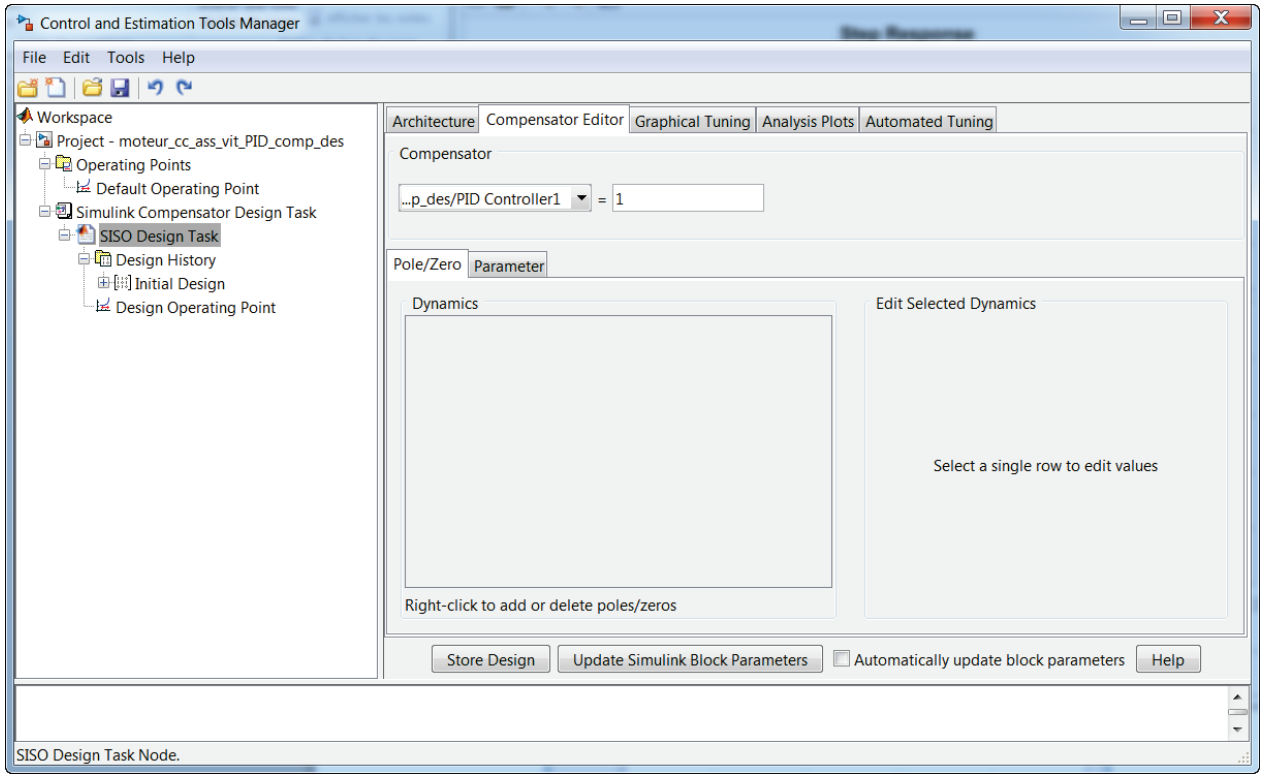


Figure 379 : onglet Compensator Editor de la fenêtre Control and Estimation Manager

Pour bien utiliser l'outil, il est nécessaire de pouvoir visualiser les trois fenêtres simultanément :

- **Control and estimation Tool Manager** pour effectuer les réglages
- **SISO Design for SISO Design Task** pour voir l'influence des réglages sur le comportement de la boucle ouverte
- **Linear Analysis for SISO Design Task** pour voir l'influence des réglages sur le comportement de la boucle fermée.

Il sera plus confortable de disposer soit d'un grand écran, soit de travailler avec deux écrans.

6. Réglage du PID

Dans la fenêtre **Control and Estimation Tools Manager**, onglet **Compensator design**, cliquer sur l'onglet **Parameter**.

Il est maintenant possible de procéder aux réglages du PID et de voir évoluer de manière dynamique l'ensemble des tracés.

Faire glisser les différents curseurs et observer l'influence du réglage sur les **Design Plots** et sur les **Analysis Plots**.

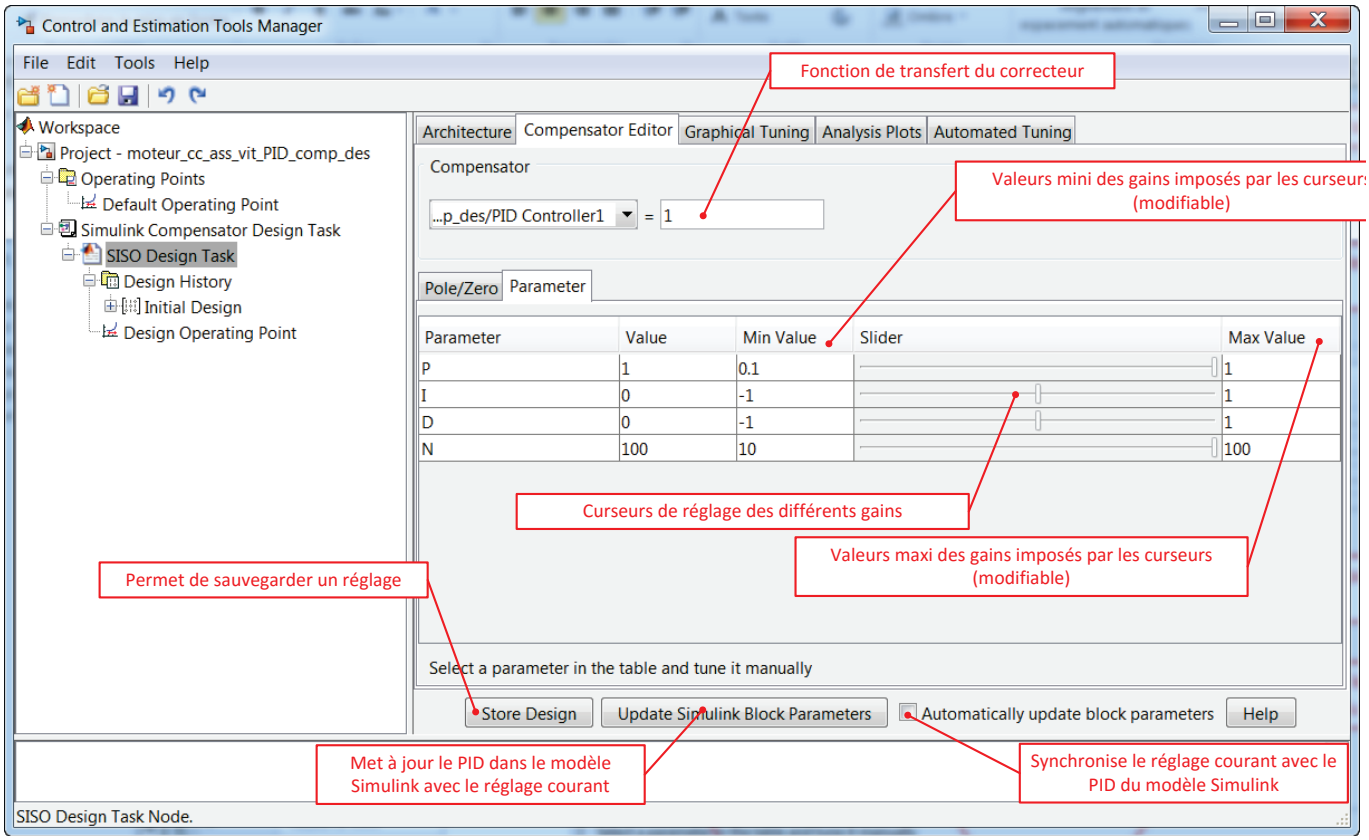


Figure 380 : les fonctions de la fenêtre de réglage du PID

7. Définition et visualisation des critères de performance

Le réglage du correcteur ne peut se faire que si des critères de performance sont définis. Il est possible de définir et de visualiser ces critères sur les courbes de la fenêtre **Linear Analysis for SISO Design Task**.

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF et choisir **Properties**, onglet **Options** puis définir que le critère de rapidité est le temps de réponse à 5%.

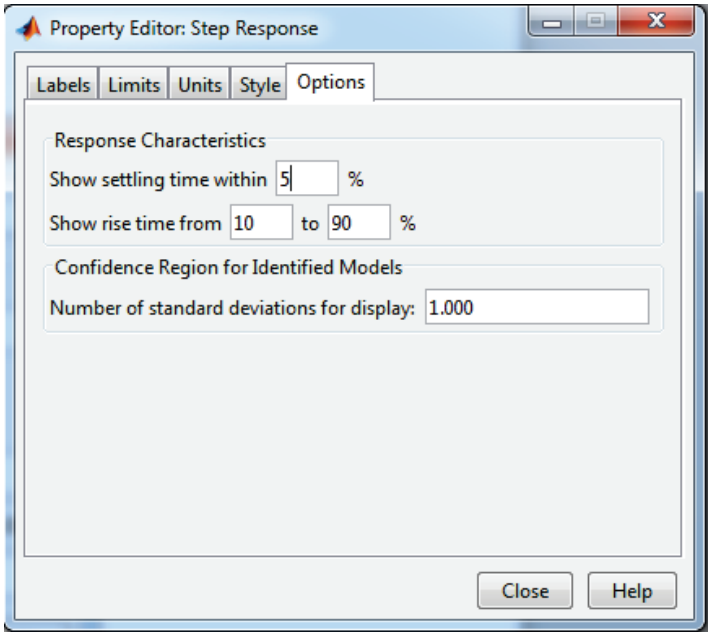


Figure 381 : définition des critères de rapidité

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF et choisir **Characteristics** puis **Settling Time** pour afficher le temps de réponse à 5%.

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF et choisir **Design Requirement New** et entrer les critères de performance pour le réglage du PID conformément à la Figure 382.

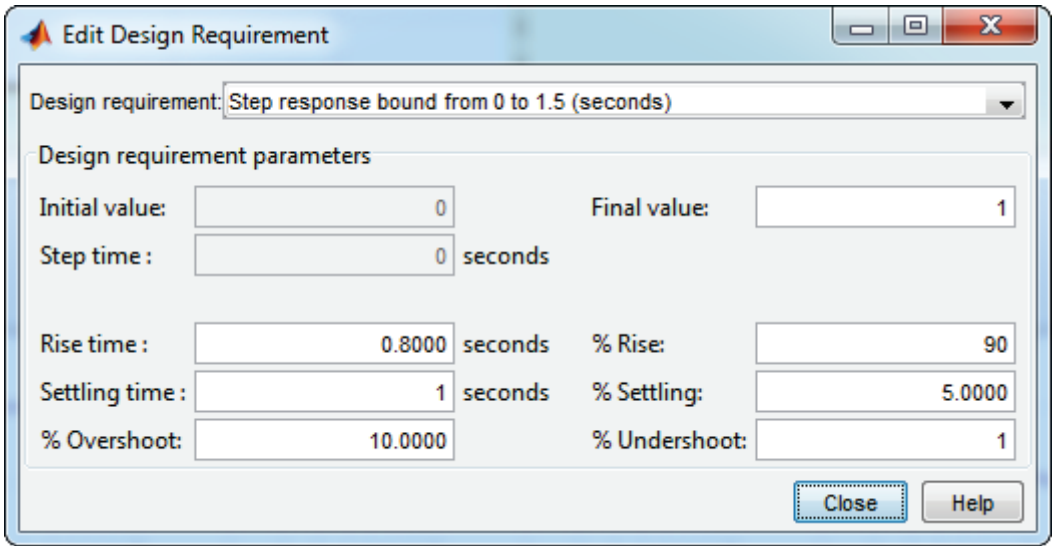


Figure 382 : définition des critères de performances

On impose ici :

- un temps de montée (de 0 à 90%) de 0.8 s
- un temps de réponse à 5% de 1 s
- un dépassement maxi de 10%

La fenêtre graphique permet de visualiser la zone dans laquelle doit se situer la réponse indicielle pour satisfaire le cahier des charges (Figure 383).

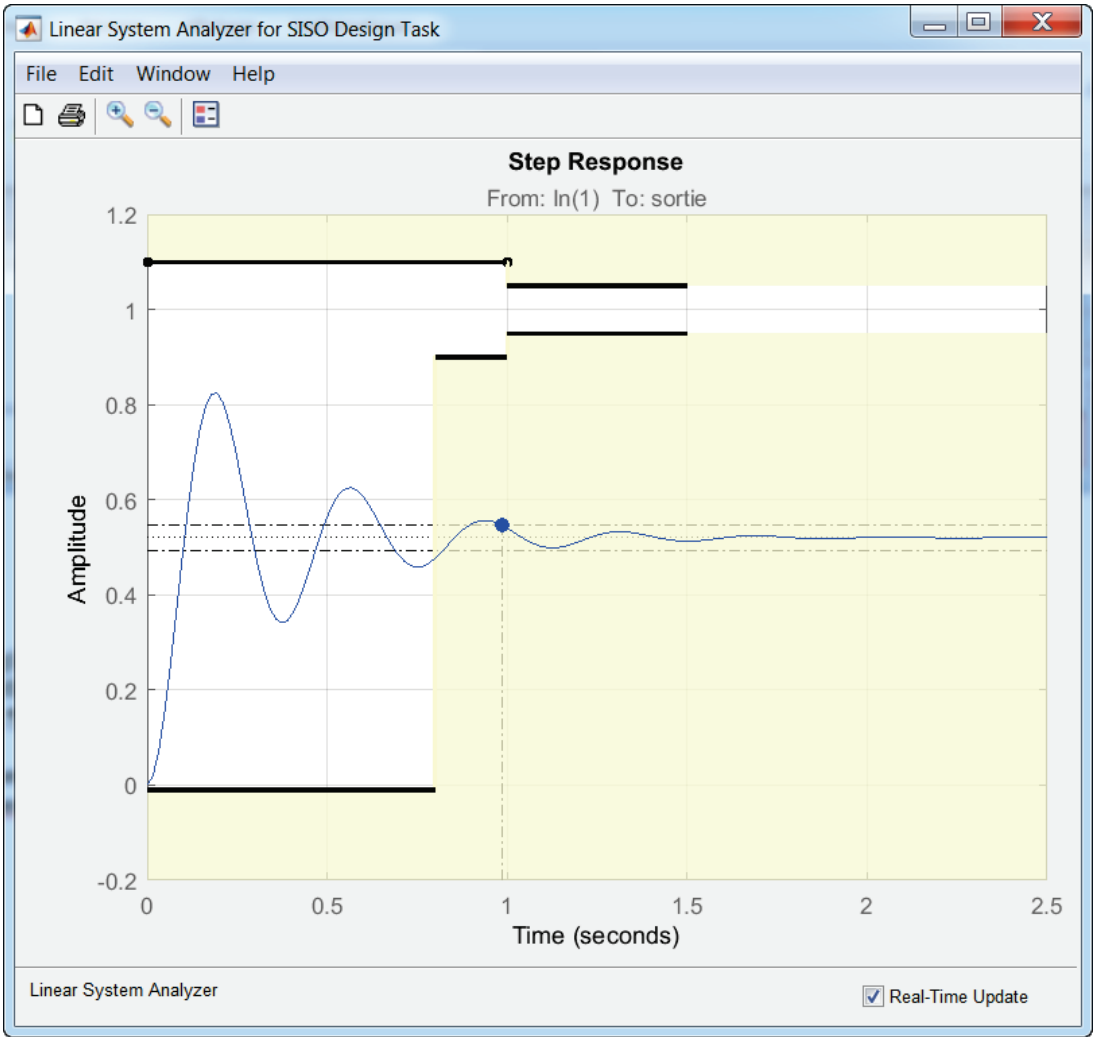


Figure 383 : visualisation des critères de performances dans la fenêtre graphique

La réponse indicielle doit se trouver en dehors de la zone jaune qui est définie par les différents critères pour que le cahier des charges soit respecté.

8. Réglage du PID à l'aide des curseurs

La réponse n'est pas satisfaisante. A l'aide des curseurs, effectuer manuellement le réglage du PID. On peut voir évoluer dynamiquement les tracés de la fenêtre **SISO Design for SISO Design TASK**.
La fenêtre de la Figure 384 montre un réglage satisfaisant du PID.

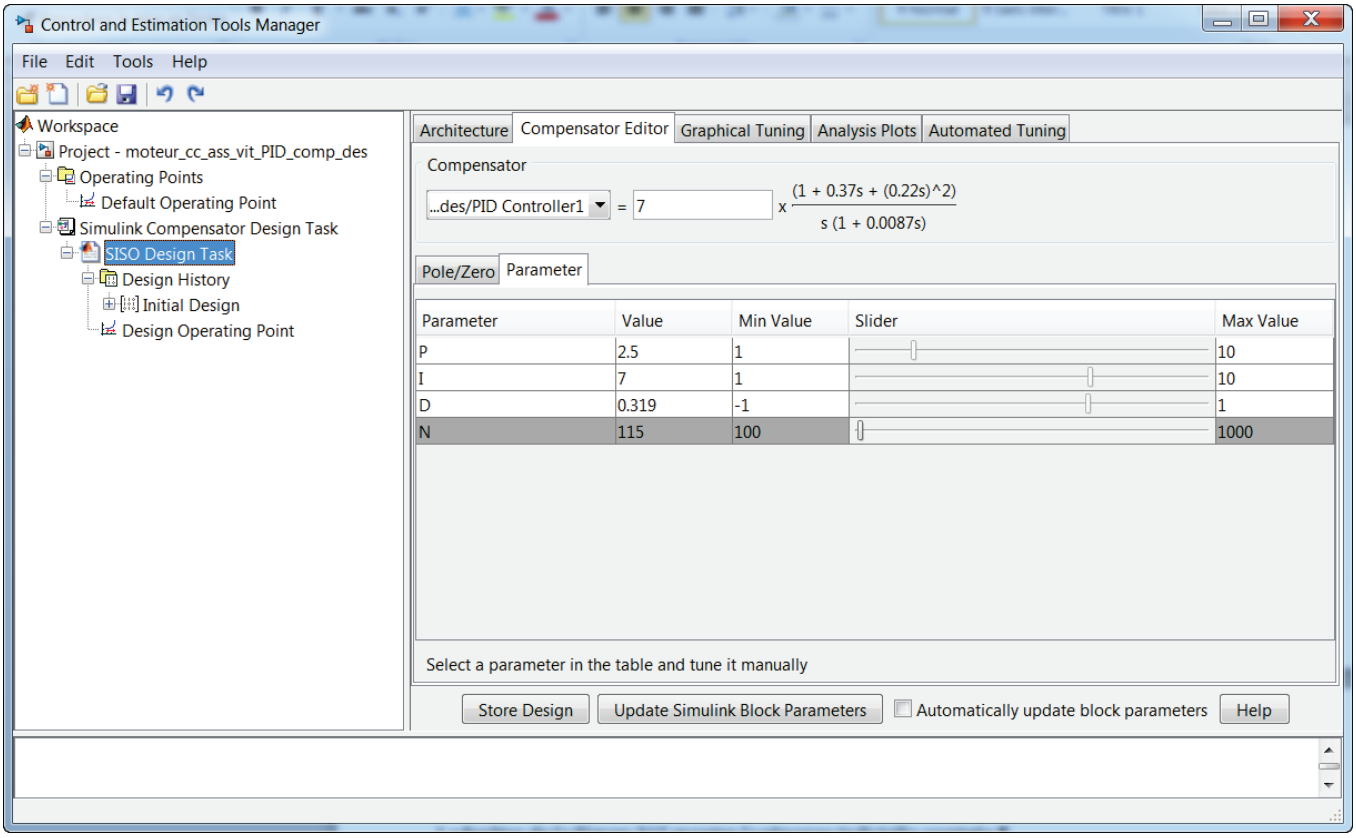


Figure 384 : réglage satisfaisant le cahier des charges du PID

La fenêtre de la Figure 385 montre la réponse indicielle corrigée.

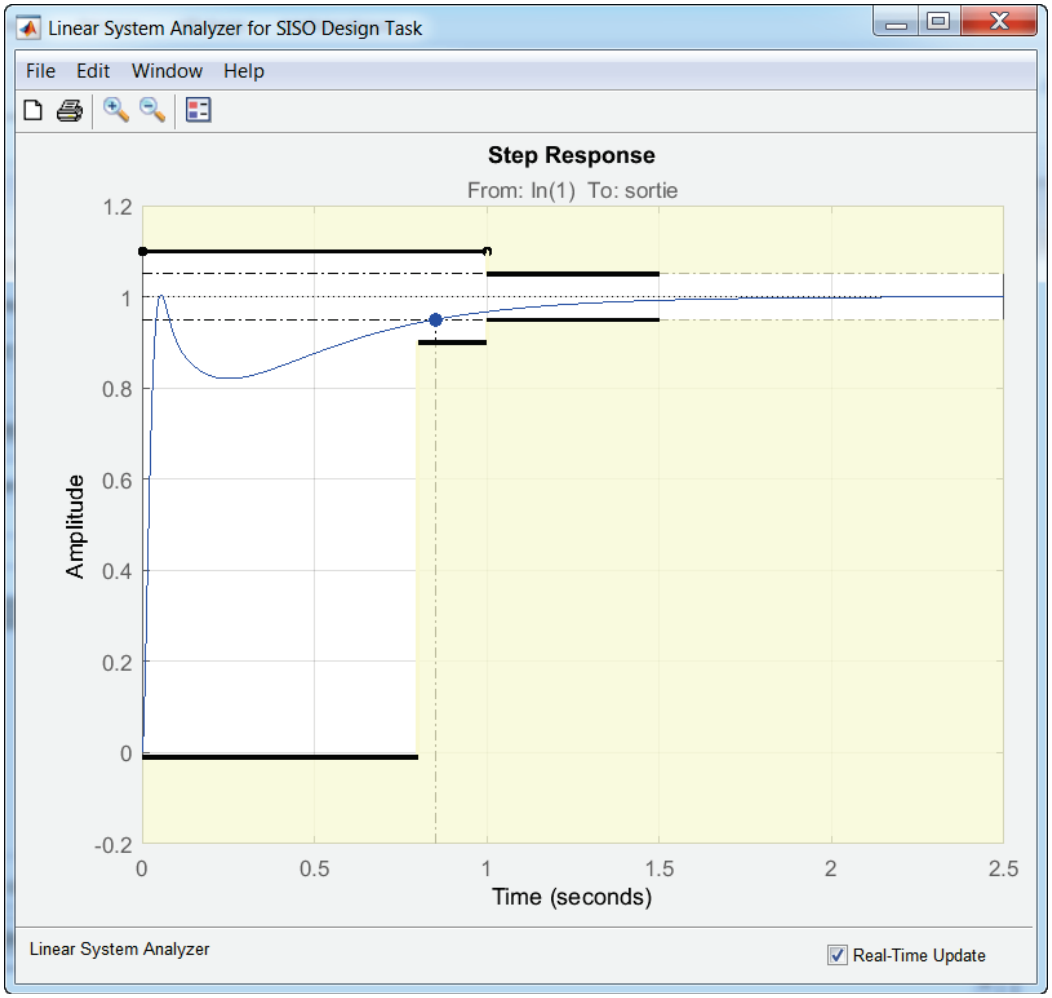


Figure 385 : résultat du réglage manuel du PID sur la réponse temporelle

Le réglage étant satisfaisant vis-à-vis de la consigne, nous pouvons maintenant vérifier le comportement vis-à-vis de la perturbation.

Pour cela dans la fenêtre **Control and Estimation Tool Manager** sélectionner l'onglet **Analysis Plots**, puis sélectionner la réponse indicielle vis-à-vis d'un échelon de perturbation comme indiqué sur la Figure 386.

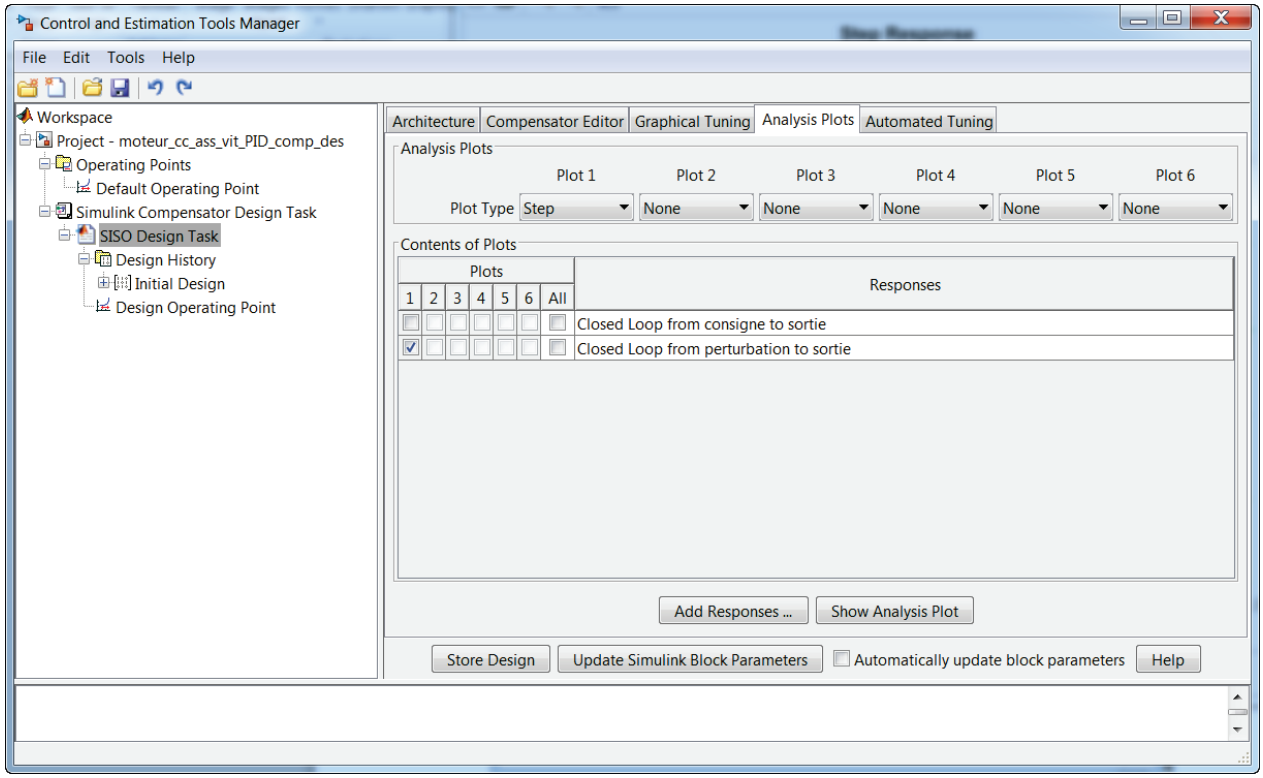


Figure 386 : sélection de la réponse indicielle vis-à-vis de la perturbation

La fenêtre **Linear Analysis for SISO Design Task** permet de visualiser la réponse indicielle vis-à-vis de la perturbation. Les critères de performances de la réponse indicielle vis-à-vis de la consigne apparaissent encore. Vous pouvez les supprimer en cliquant avec le bouton droit de la souris dans la partie jaune de la fenêtre graphique et choisir **Delete**.
On obtient alors la réponse vis-à-vis de la perturbation comme le montre la Figure 387.

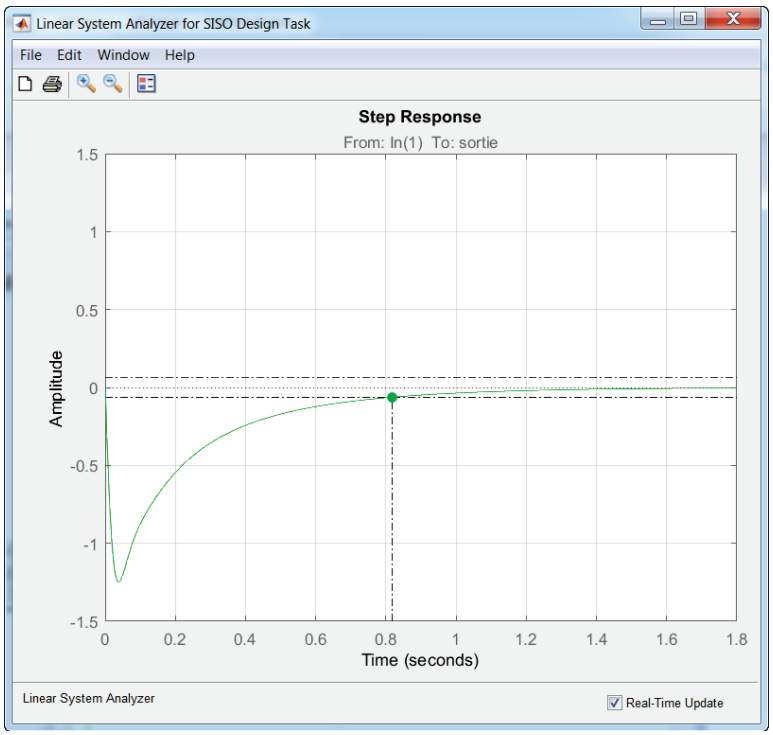


Figure 387 : réponse indicielle du système vis-à-vis d'un échelon de perturbation

Nous constatons que la perturbation est rapidement rejetée et que son influence peut être considéré comme négligeable au bout de 0.8 s . Le réglage effectué est donc satisfaisant.

La fenêtre **SISO Design for SISO Design Task** permet de visualiser l'influence de ces réglages sur les **Design Plots** et d'apprécier les critères de marge de gain et de marge de phase.

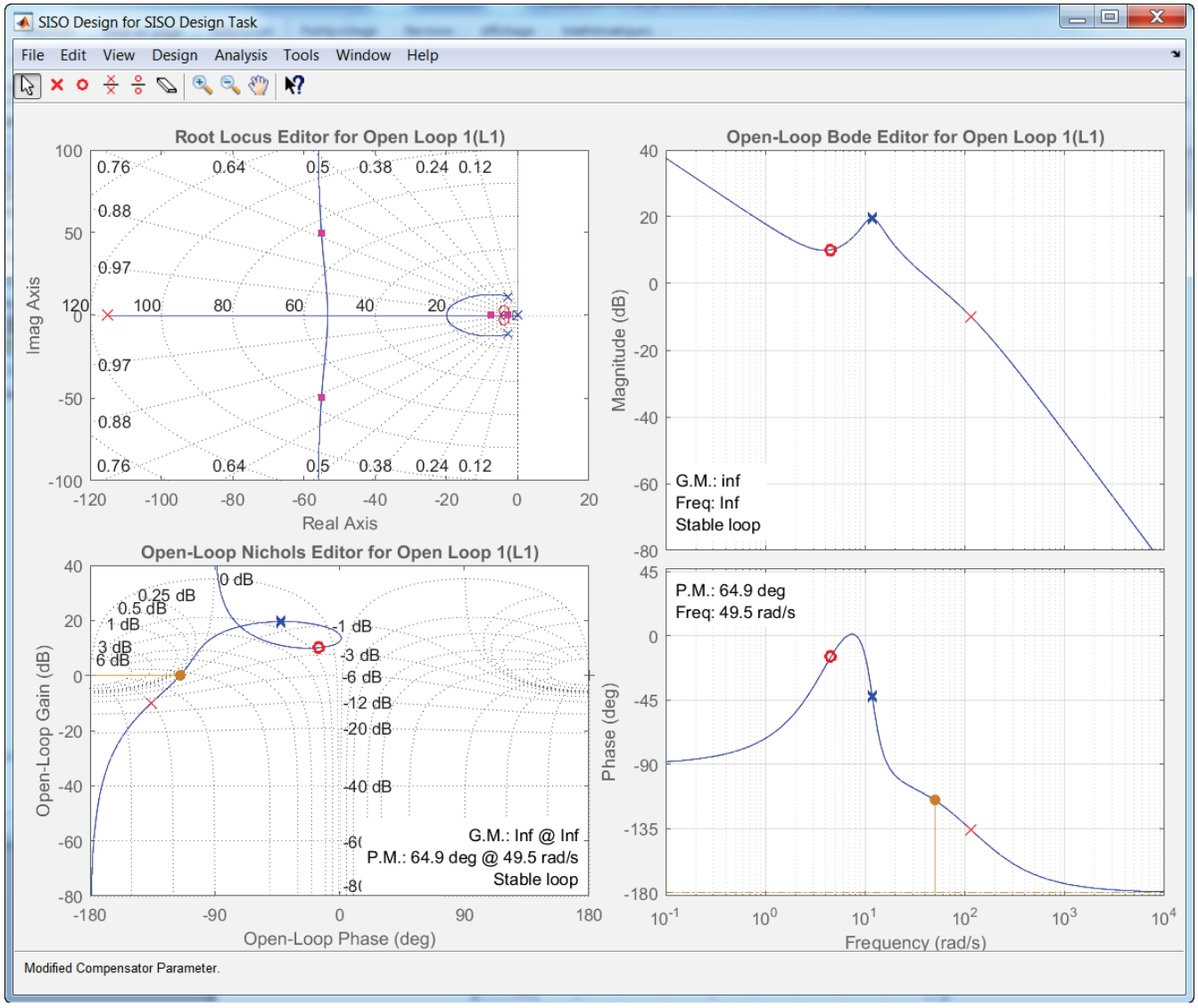


Figure 388 : influence des réglages effectués sur les Design Plots

9. Exportation du réglage dans le modèle Simulink

Cliquer maintenant sur **Update Simulink Block Parameter** dans la fenêtre **Compensator Editor** afin d'exporter les réglages du PID dans le bloc PID du modèle Simulink.
Une fenêtre confirme l'importation des paramètres de réglage dans Simulink (Figure 389).

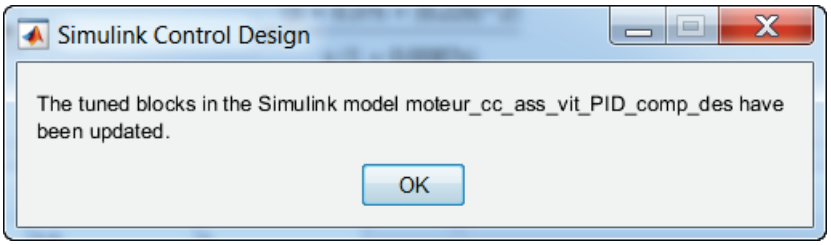


Figure 389 : confirmation de l'importation des paramètres de réglages dans Simulink

Retourner dans le modèle Simulink puis **Lancer** la simulation et visualiser la réponse du moteur avec le PID réglé.

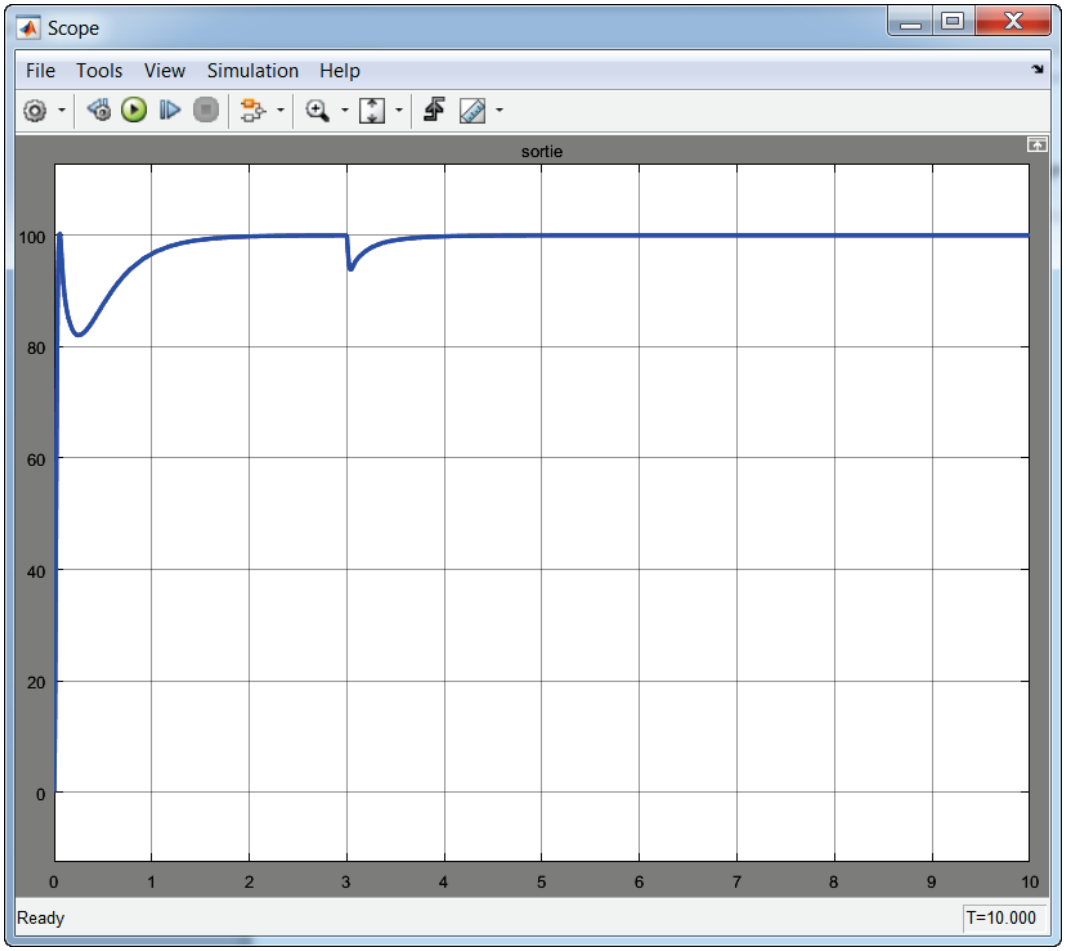


Figure 390 : réponse temporelle après réglage du PID

Nous constatons sur la Figure 390 que la réponse temporelle est plus rapide, bien amortie et que la perturbation est bien rejetée par le système. Les critères de performances sont respectés.

IV. Conception et réglage d'un correcteur de forme quelconque

Dans de nombreuses applications de contrôle commande, le correcteur peut prendre d'autres formes que celle d'un simple correcteur PID . **MATLAB** offre la possibilité de construire un correcteur de forme quelconque à partir de l'outil « **Control System Designer** ». La démarche sera sensiblement la même que celle présentée dans la partie III. La fonction de transfert du correcteur sera construite par étape en lui ajoutant de nouveaux éléments dans le but d'améliorer les performances du système. L'outil « **Control System Designer** » permet d'intégrer dans la fonction de transfert du correcteur :

- des intégrateurs
- des pôles complexes
- des pôles réels
- des zéros complexes
- des zéros réels
- des correcteurs à avance de phase (**Lead**)
- des correcteurs à retard de phase (**Lag**)
- des filtres rejecteurs (**Notch**)

A. Ouverture du modèle

Ouvrir le fichier « ***moteur_cc_ass_vit_control_sys_des_tf*** »

Ce fichier contient le modèle de l'asservissement en vitesse d'un moteur à courant continu (Figure 391). Le correcteur a été remplacé par une fonction de transfert que l'on va construire à l'aide de l'outil « **Control System Designer** ». Pour l'instant cette fonction de transfert est un gain pur de valeur 1.

La consigne imposée est de 100 rad/s et une perturbation de couple est appliquée au système à l'instant t=3 s.

Les points de linéarisation sont déjà placés dans le modèle.

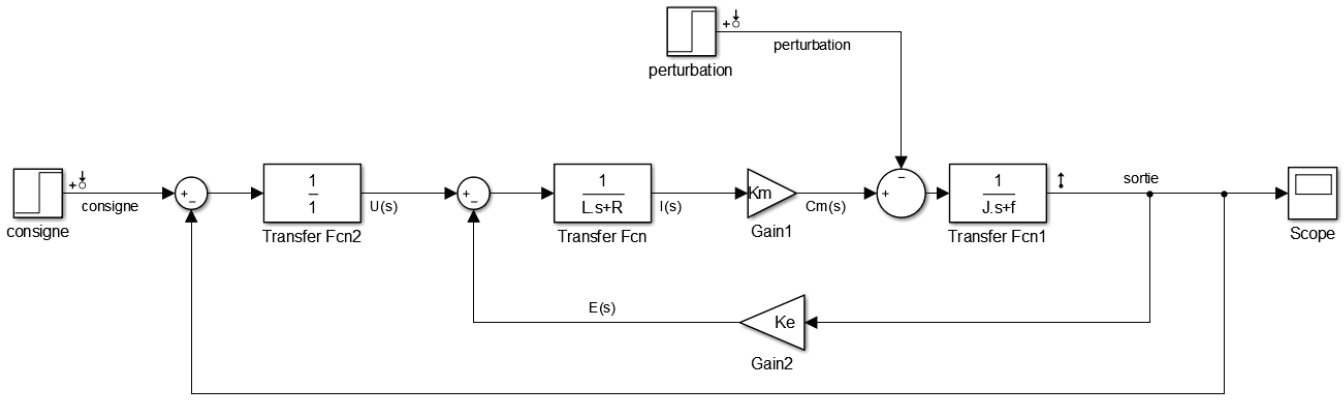


Figure 391 : modèle de l'asservissement du moteur à continu avec correction par fonction de transfert à régler

Exécuter le script ***parametres_moteur_cc.m.***

Lancer la simulation et **visualiser** la réponse dans le scope.

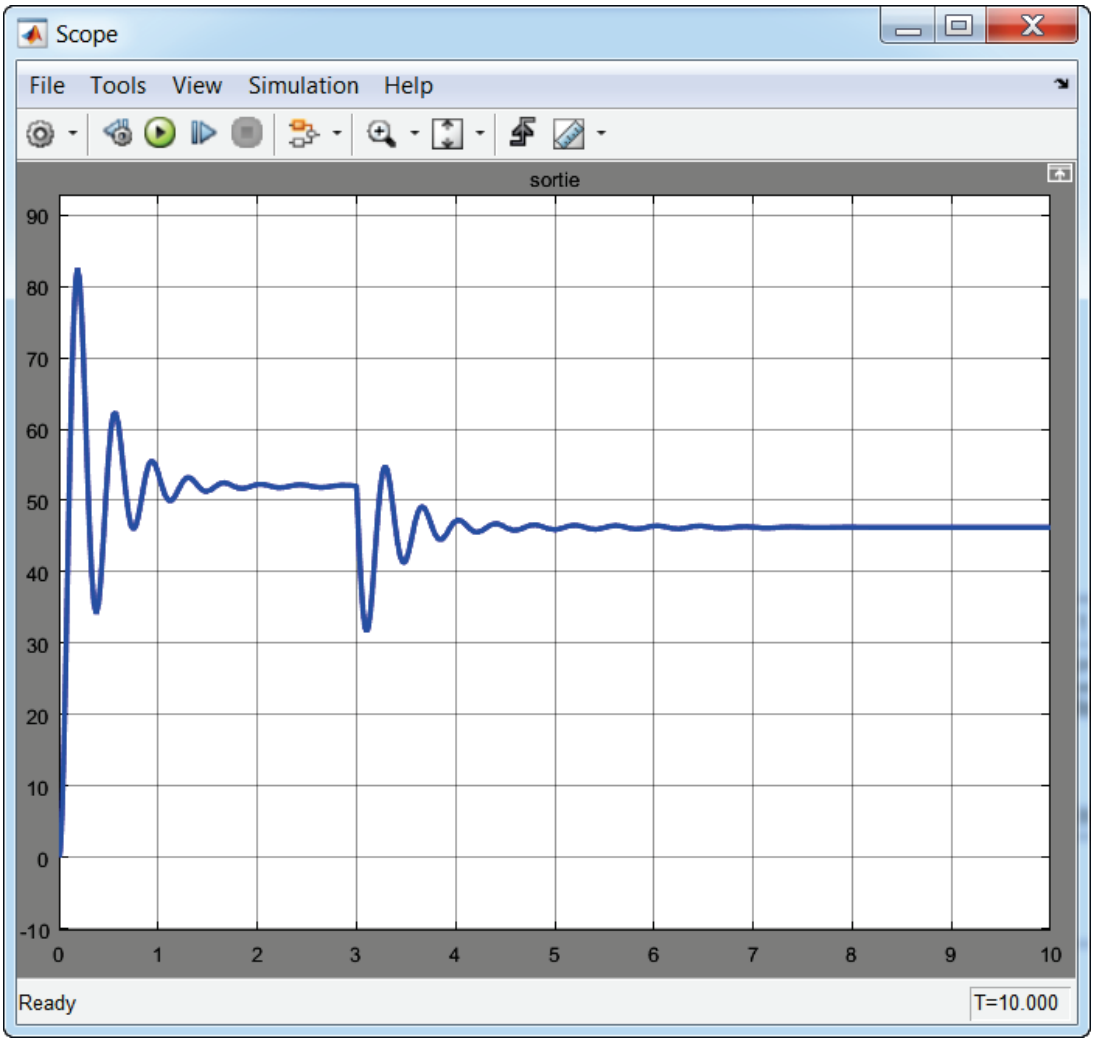


Figure 392 : réponse du système non corrigé

Nous pouvons constater que la réponse du système n'est pas satisfaisante. Le système n'est pas précis, mal amorti et ne rejette pas correctement la perturbation.

B. Conception du correcteur

A partir de la barre de commande, sélectionner **Analysis/Control Design/Control System Designer.**

Cette commande entraîne l'ouverture de la fenêtre **Control and Estimation Tools Manager** qui propose de choisir les blocs que l'on veut régler, de visualiser les points d'entrée et de sortie de la boucle fermée.

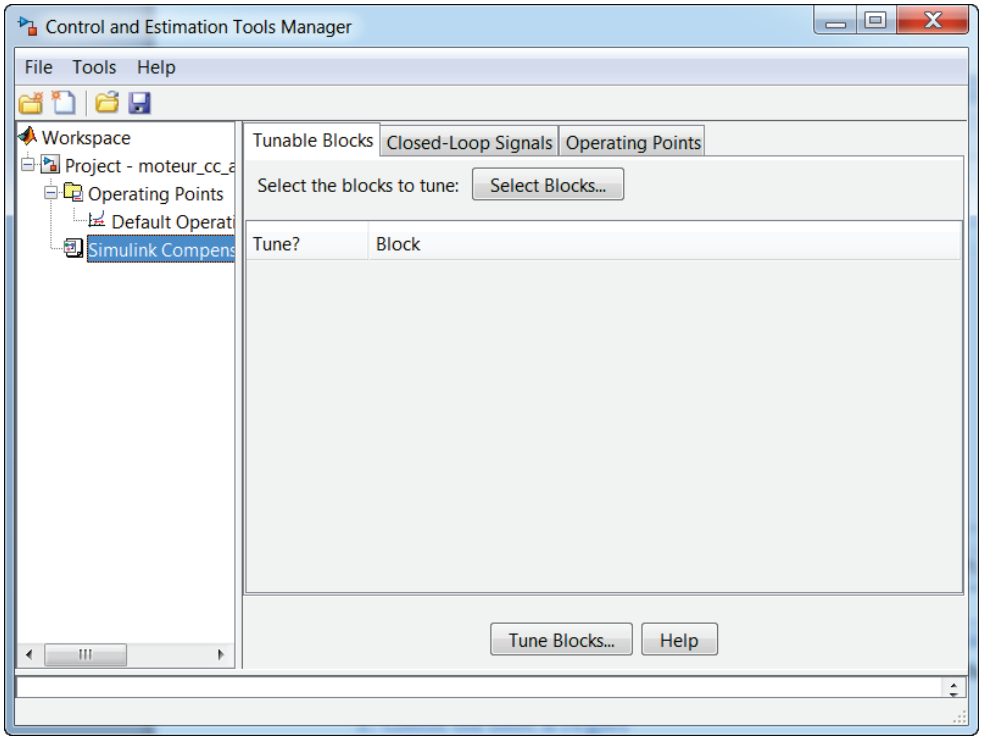


Figure 393 : fenêtre Control and Estimation Tools Manager

1. Choix du bloc à régler

Sélectionner l'onglet **Tunable Blocks** pour choisir le bloc à régler puis cliquer sur **Select Blocks.**

La fenêtre **Select Blocks to Tune** s'ouvre et propose tous les blocs du modèle qui peuvent être réglés.

Sélectionner **Correcteur** (c'est le nom donné dans le modèle **Simulink** à la fonction de transfert du correcteur que nous souhaitons concevoir) et cliquer sur **OK** afin de pouvoir procéder à la conception du correcteur.

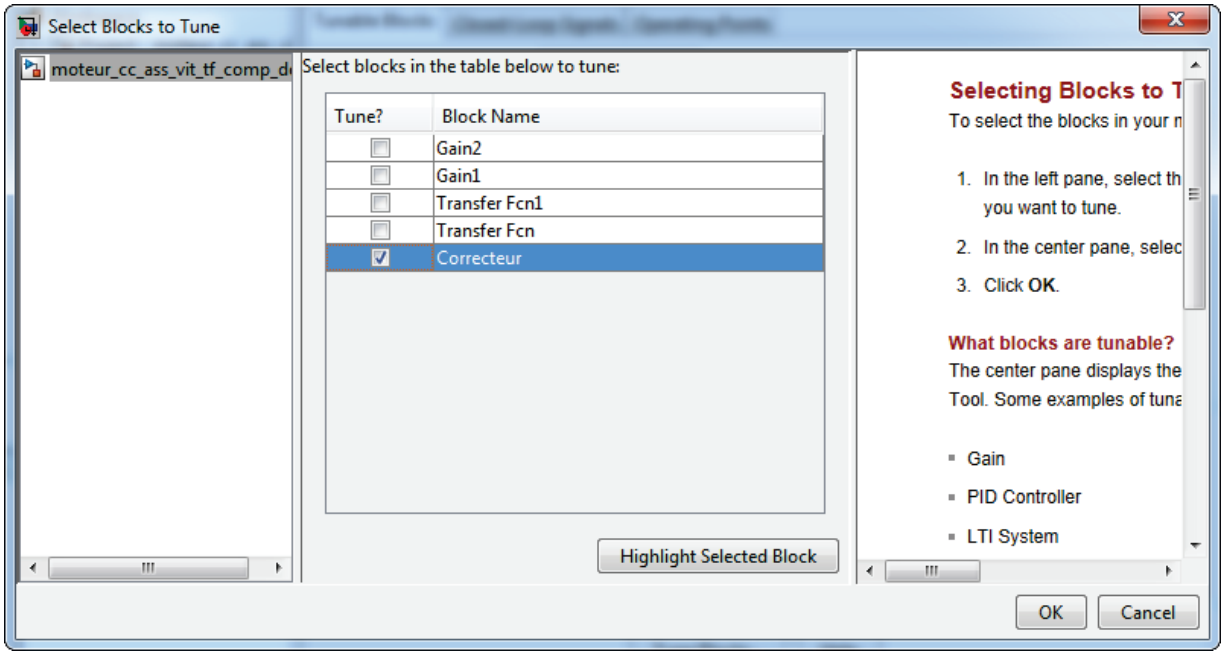


Figure 394 : choix du bloc à régler

Cliquer ensuite sur **Tune Block** pour commencer les réglages.

Une fenêtre d'informations (Figure 395), vous permet de prendre connaissance des fonctionnalités de l'outil **Control System Designer.**

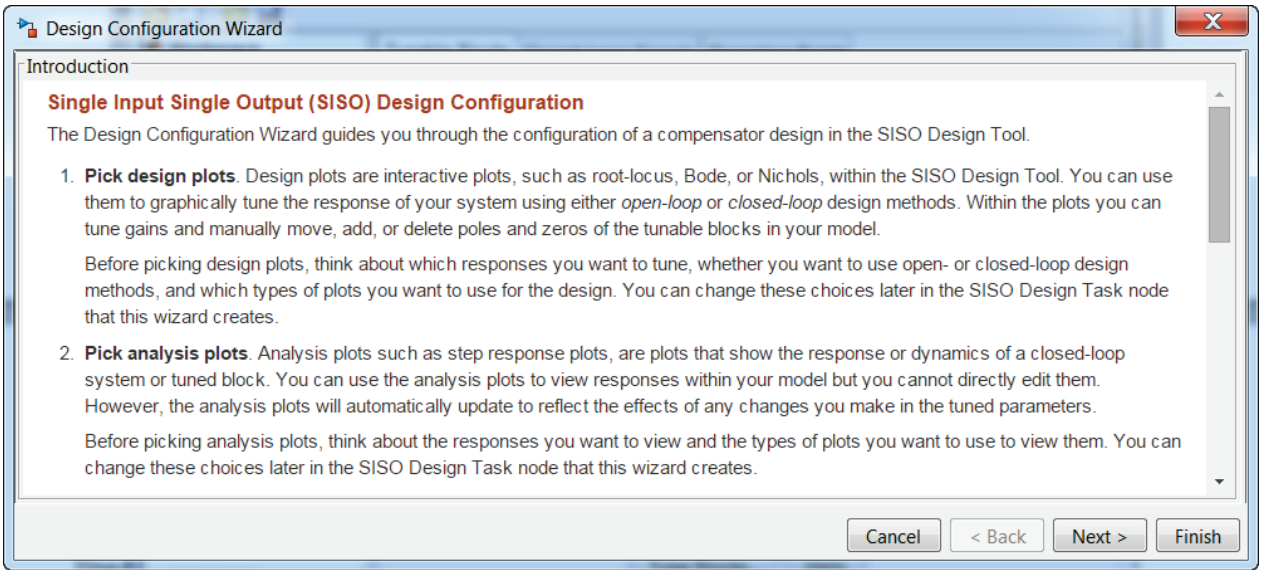


Figure 395 : fenêtre d'information de l'outil Control System Designer

C**liquer** sur **Next**.

2. Choix des tracés à visualiser pour la boucle ouverte

Les procédures de choix des tracés ont été décrites dans le paragraphe III de ce chapitre.

La fenêtre **Design Configuration Wizard** s'ouvre et vous permet de choisir les tracés sur lesquels il sera possible d'agir pour effectuer les réglages.

Choisir les tracés conformément à la Figure 396.

- Root locus (lieu des pôles de la **fonction de transfert en boucle fermée**)
- Diagramme de Bode de la **fonction de transfert en boucle ouverte**
- Diagramme de Nichols de la **fonction de transfert en boucle ouverte**

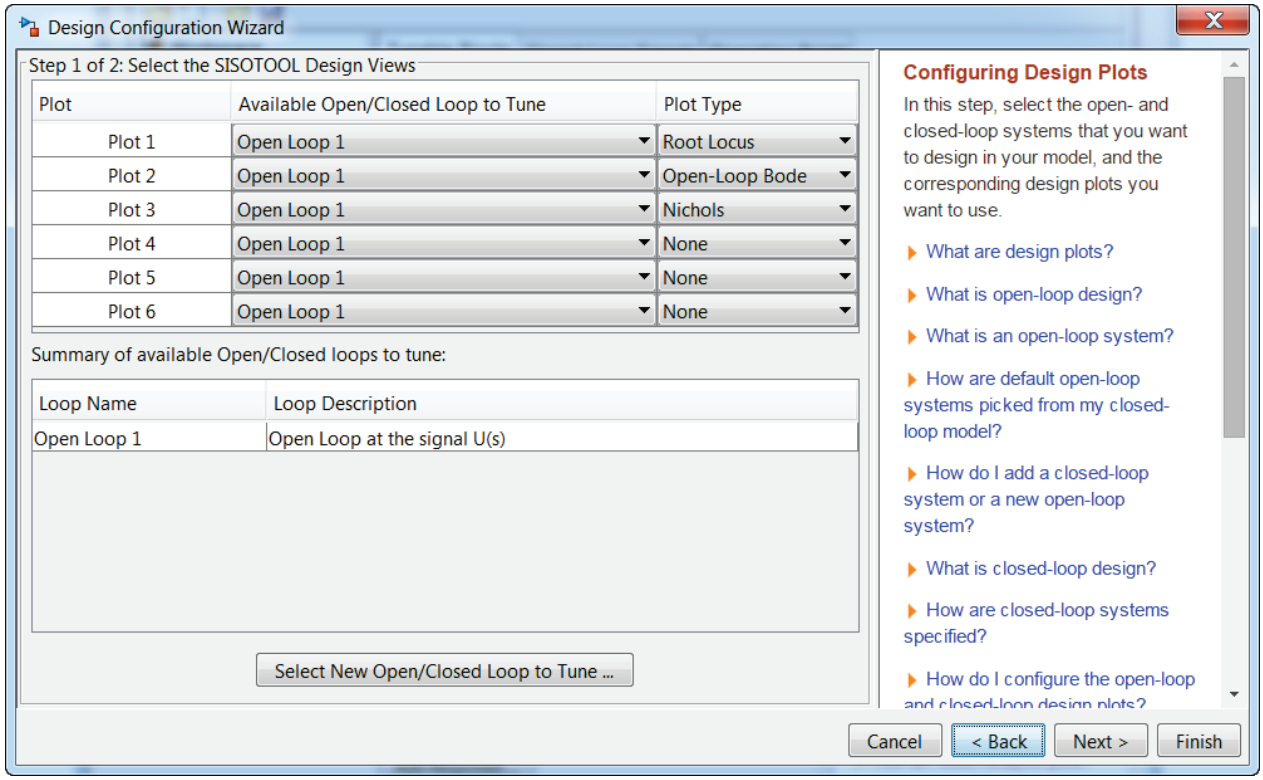


Figure 396 : choix des tracés à visualiser pour la boucle ouverte

3. Choix des tracés pour visualiser les performances de la boucle fermée

Pour visualiser les performances de la fonction de transfert en boucle fermée, nous pouvons afficher :

- La réponse temporelle à un échelon de la **fonction de transfert en boucle fermée**
- Le diagramme de Bode de **la fonction de transfert en boucle fermée**

Nous nous intéresserons ici qu'au comportement du système bouclé vis-à-vis de l'entrée.

Configurer la fenêtre **Design Configuration Wizard** conformément à la Figure 397.

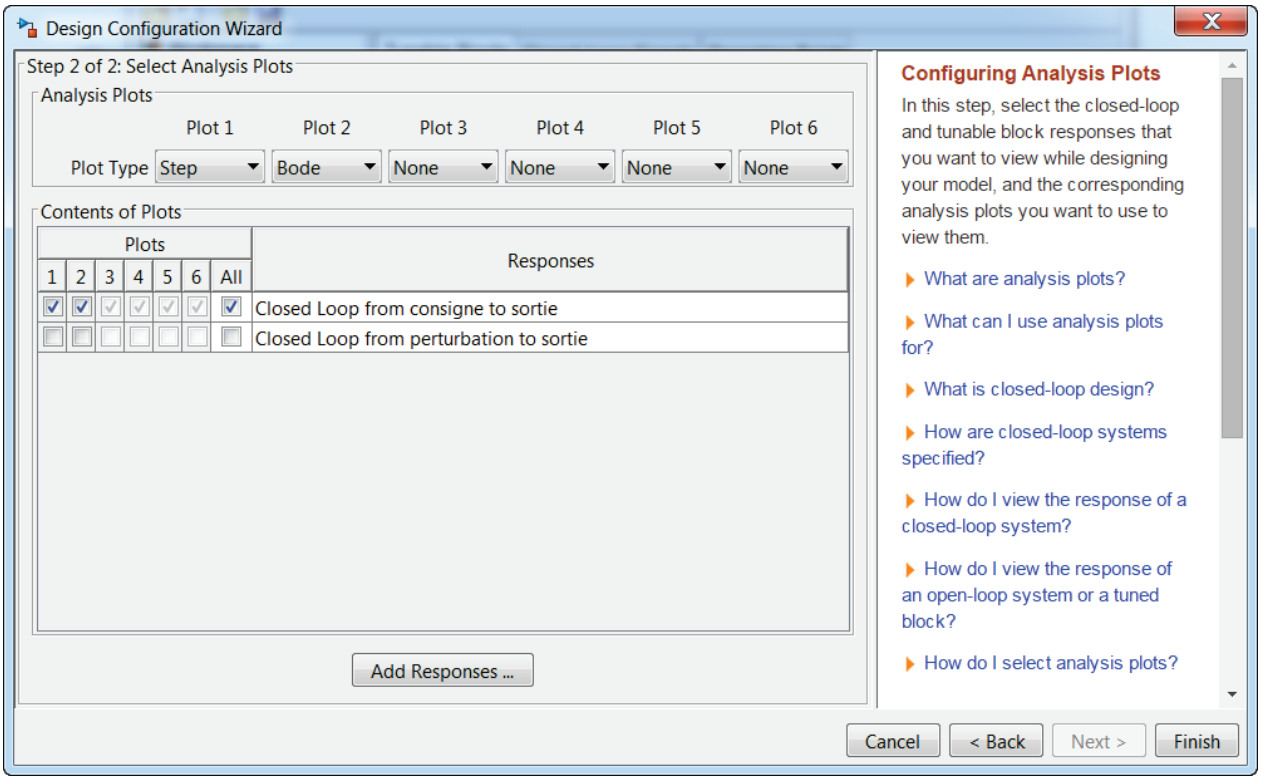


Figure 397 : sélection des tracés de visualisation des performances de la boucle fermée

Cliquer sur **Finish**

4. Analyse des fenêtres graphiques de l'outil « compensator design »

Les deux fenêtres **Linear System Analyzer for SISO Design Task** (permet de visualiser les performances de la fonction de transfert en boucle fermée) et **SISO Design for SISO Design Task** (permet de visualiser le comportement de la fonction de transfert en boucle ouverte) apparaissent alors à l'écran.

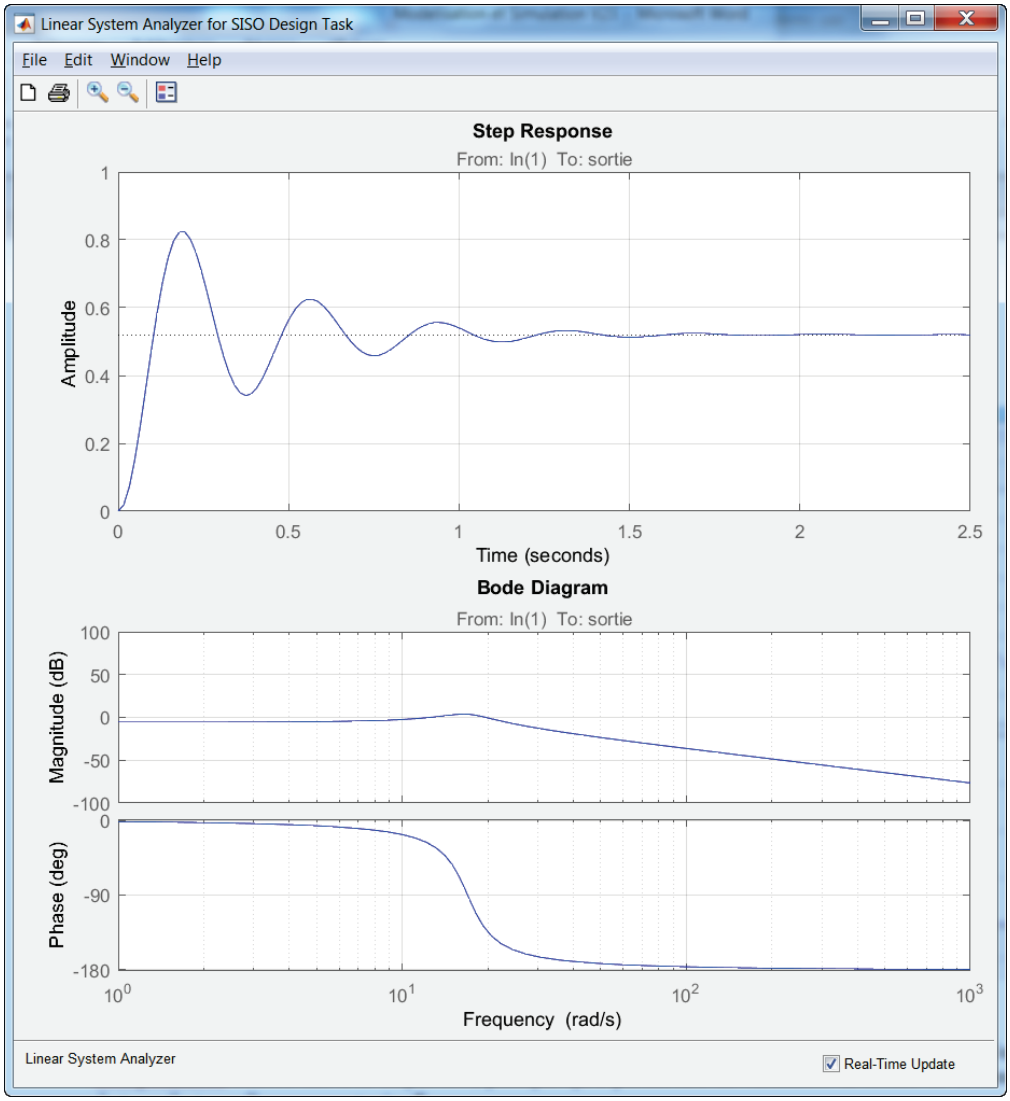


Figure 398 : visualisation des performances de la fonction de transfert en boucle fermée

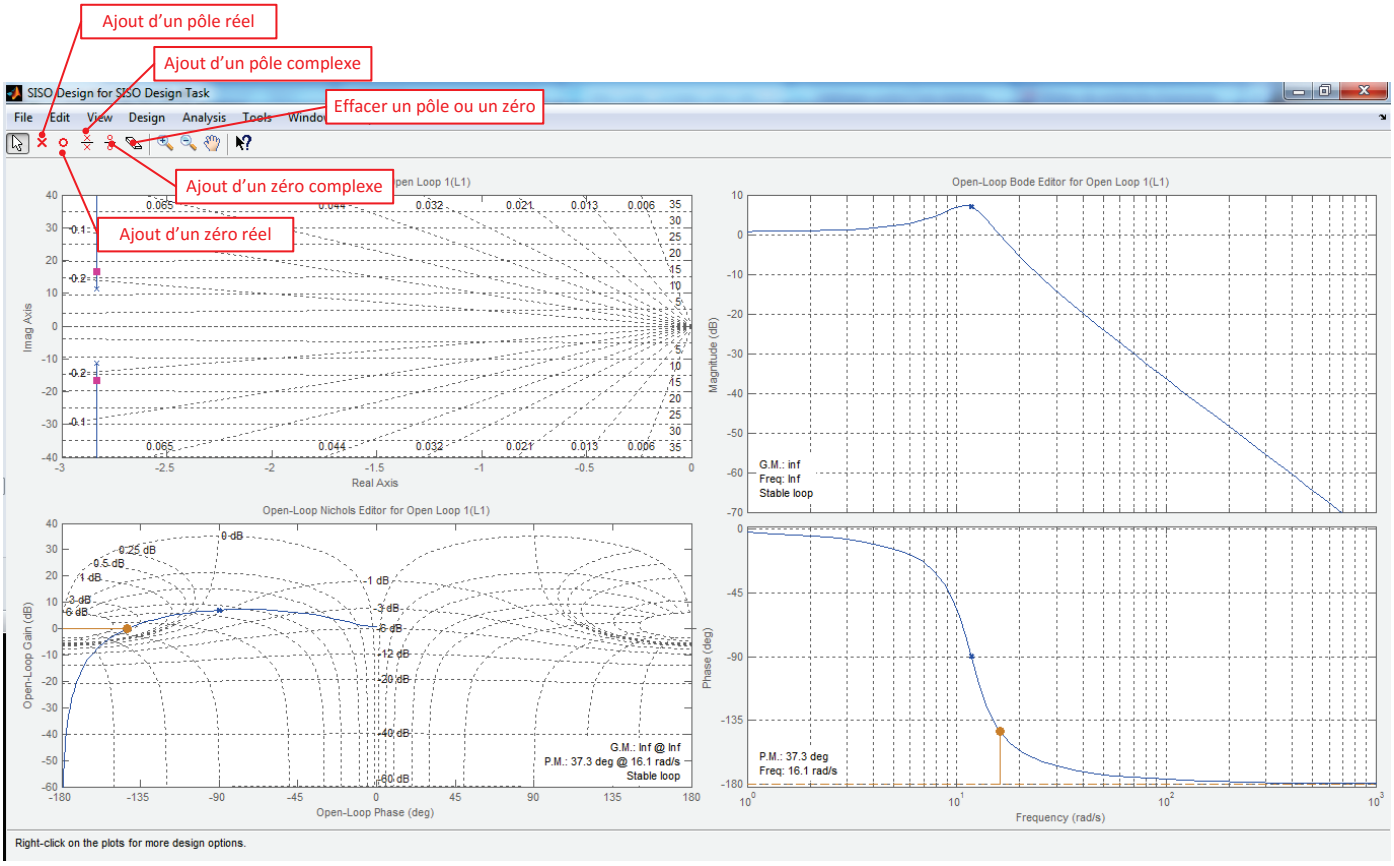


Figure 399 : la barre de commande de la fenêtre SISO Design for SISO Design Task

A noter que la fenêtre **Control and Estimation Tool Manager** reste ouverte également. L'onglet **Compensator Editor** est très important, puisque qu'il permettra de visualiser la forme de la fonction de transfert que l'on va construire.

Pour bien utiliser l'outil, il est préférable de pouvoir visualiser les trois fenêtres simultanément :

- **Control and estimation Tool Manager** pour effectuer les réglages
- **SISO Design for SISO Design Task** pour voir l'influence des réglages sur le comportement de la boucle ouverte
- **Linear System Analyzer for SISO Design Task** pour voir l'influence des réglages sur le comportement de la boucle fermée.

5. Visualisation de l'influence du gain de la fonction de transfert en boucle ouverte

Il est possible dans un premier temps de faire varier le gain du correcteur et de visualiser son influence sur l'ensemble des diagrammes.

Dans la fenêtre **SISO Design for SISO Design Task**, sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte, **déplacer** le pointeur de la souris sur la courbe de gain. Lorsqu'il prend la forme d'une main, utiliser le **glisser déposer** pour faire translater la courbe verticalement.

Cette translation aura comme influence de modifier la valeur du gain du correcteur que nous concevons.

Déplacer la courbe de gain afin d'obtenir un gain basse fréquence de l'ordre de **20dB** pour la fonction de transfert en boucle ouverte.

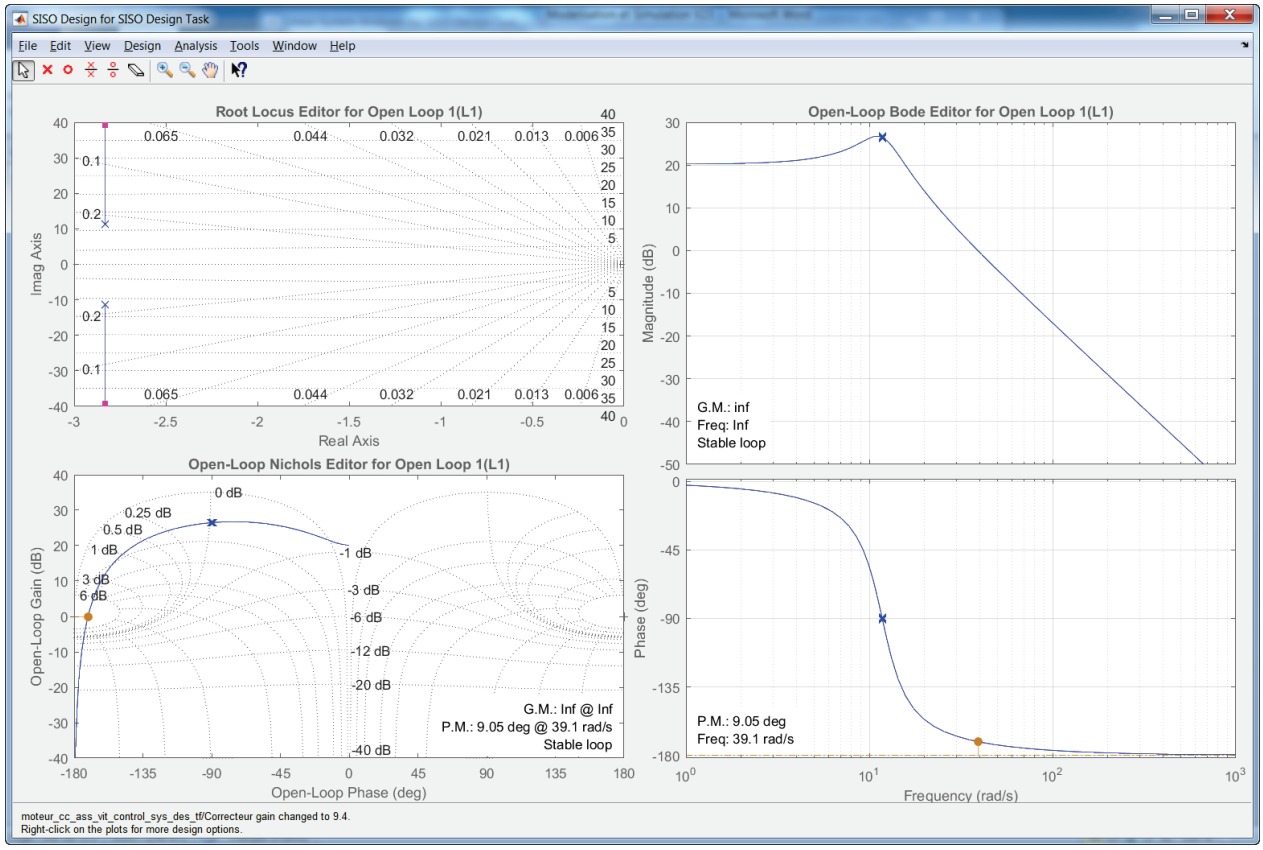


Figure 400 : variation du gain de la fonction de transfert en boucle ouverte

Nous constatons que les autres diagrammes se mettent à jour de manière dynamique en prenant en compte l'augmentation du gain de la fonction de transfert en boucle ouverte.

L'influence de l'augmentation du gain proportionnel du correcteur sur le comportement de la boucle fermée est visualisable dans la fenêtre **Linear System Analyzer for SISO Design Task**. Sur la réponse indicielle, la précision et la rapidité augmentent, mais l'amortissement diminue laissant apparaître de plus fortes oscillations. Sur le diagramme de Bode, on observe une augmentation de la bande passante à -3dB (amélioration de la rapidité) et une résonnance plus importante.

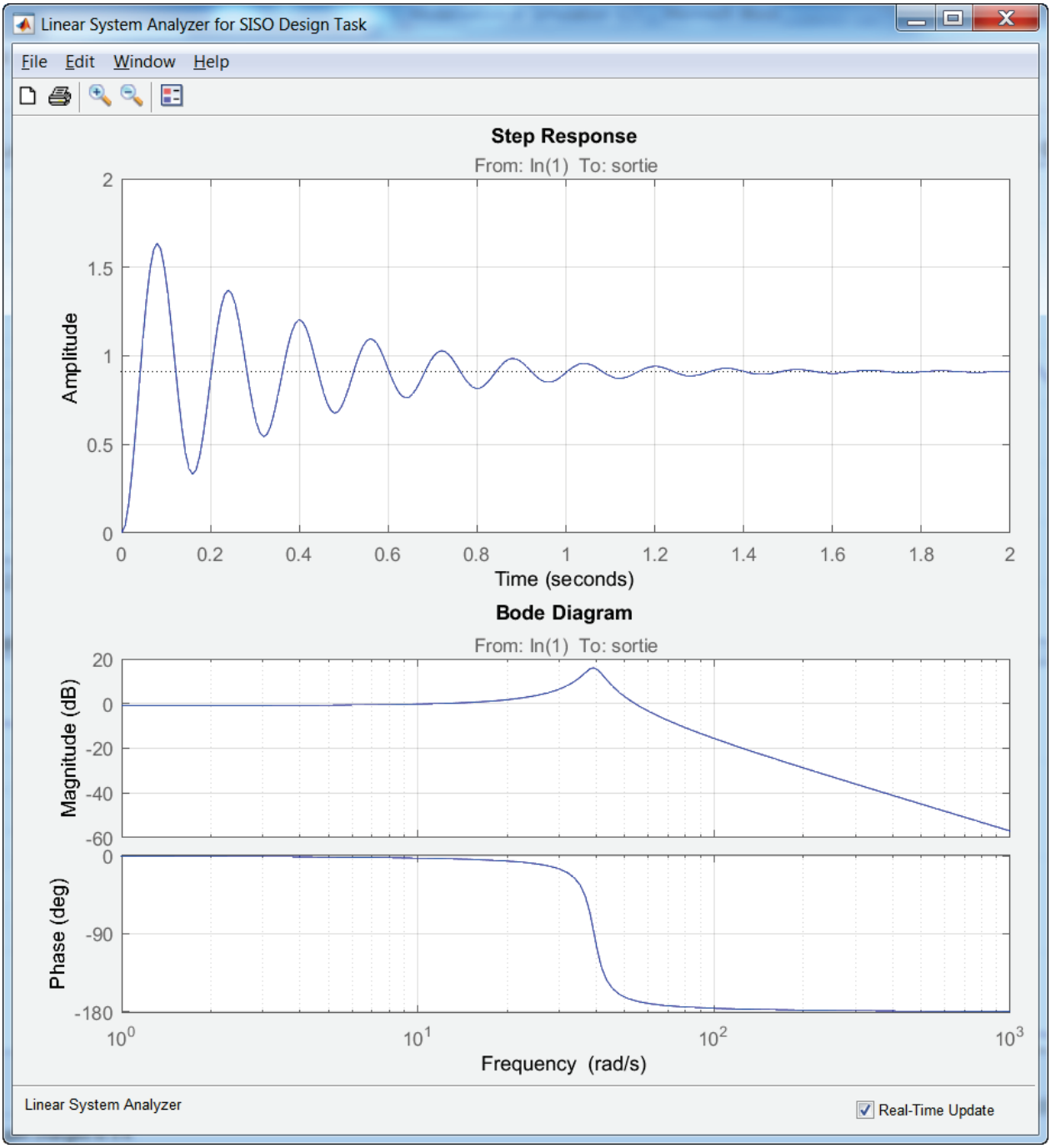


Figure 401 : visualisation de l'influence sur les performances de la boucle fermée

Il est également possible de visualiser la nouvelle fonction du correcteur dans la fenêtre **Control and Estimation Tool Manager**, onglet **Compensator Design**. Nous pouvons visualiser un gain pur de l'ordre de 10.

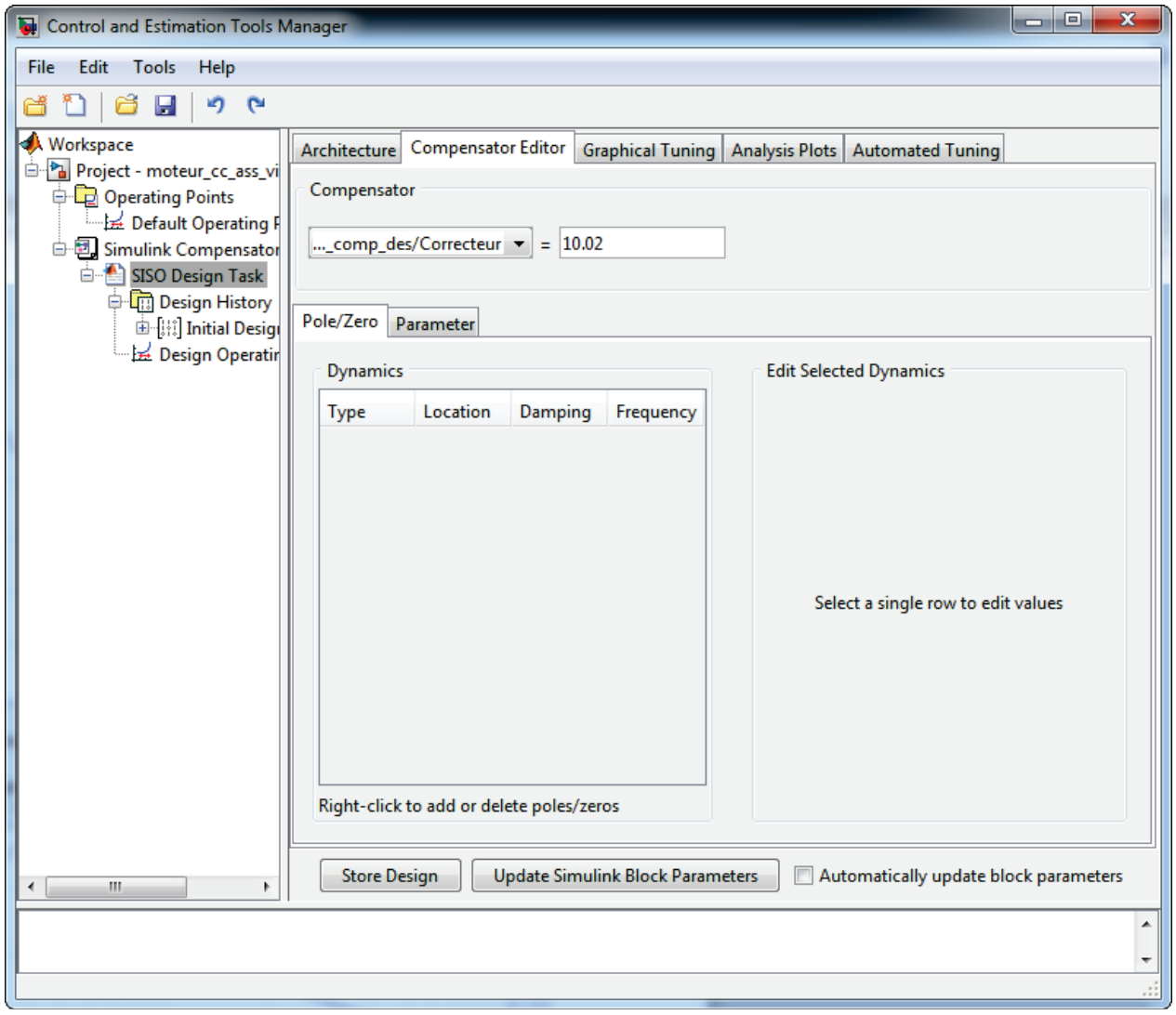


Figure 402 : visualisation de la fonction de transfert du correcteur

Il est également possible de modifier le gain du correcteur en utilisant d'autres procédés :

- En déplaçant les pôles de la FTBF le long du lieu des pôles (root locus)
- En faisant translater verticalement le tracé de Nichols
- En indiquant directement la valeur du gain dans l'onglet **Compensator Design** de la fenêtre **Control and Estimation Tool Manager**.

Dans tous les cas la mise à jour des autres diagrammes est réalisée automatiquement.

6. Ajout d'un intégrateur

Afin de rendre le système précis, il est possible d'ajouter un intégrateur dans le correcteur.
Pour cela **cliquer** avec le bouton droit de la souris, dans la fenêtre **SISO Design for SISO design Task** puis sélectionner **Add Pole zero/Integrator.** Un intégrateur est ajouté à notre correcteur et toutes les courbes sont mises à jour.

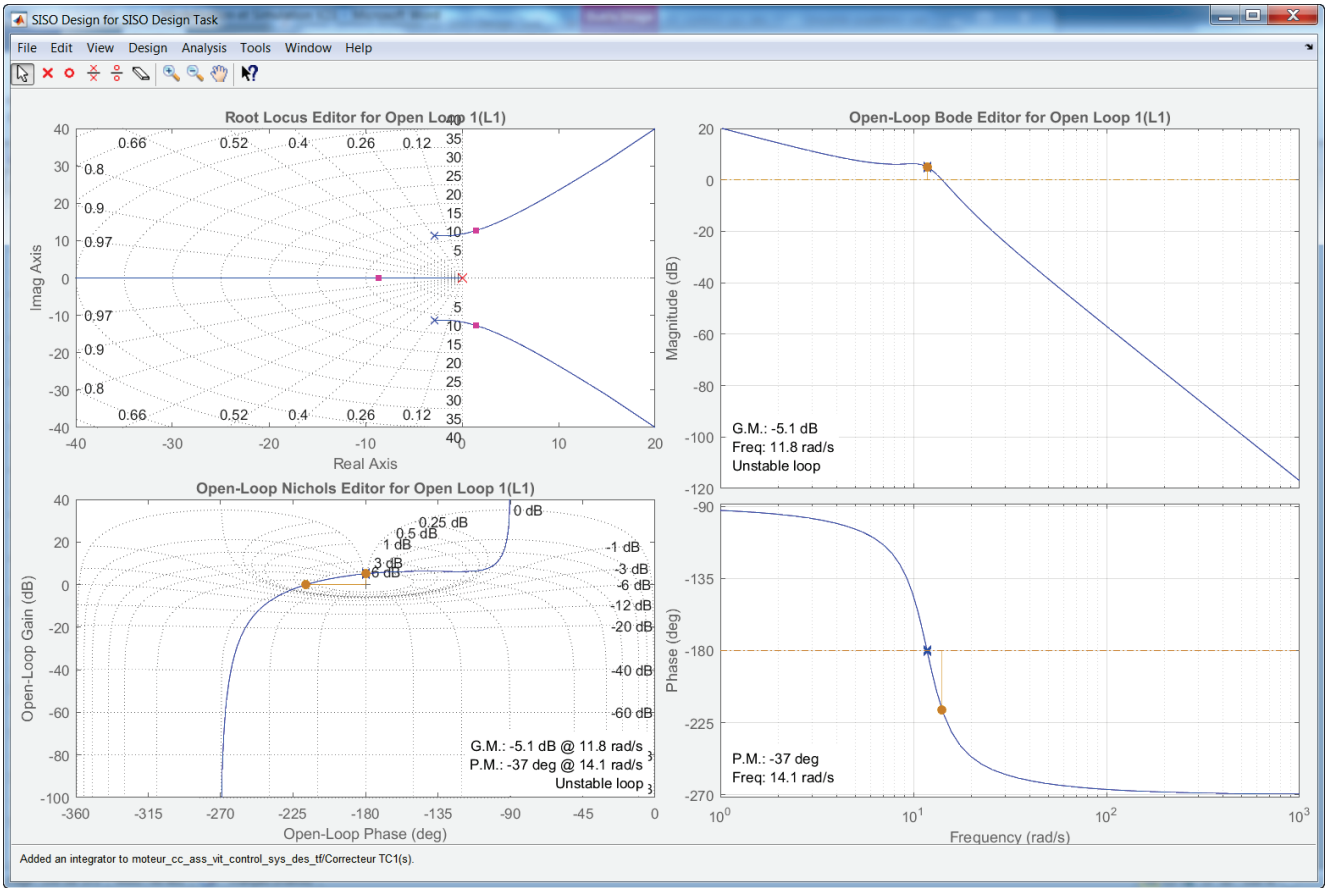


Figure 403 : ajout d'un intégrateur

On remarque sur le lieu des pôles, que des pôles de la fonction de transfert en boucle fermée sont à partie réelle positive ce qui rend le système instable. Les marges de gain et de phase sont négatives et sont visibles sur le diagramme de Bode et le tracé de Nichols.
La réponse indicielle met en évidence cette instabilité (Figure 404).

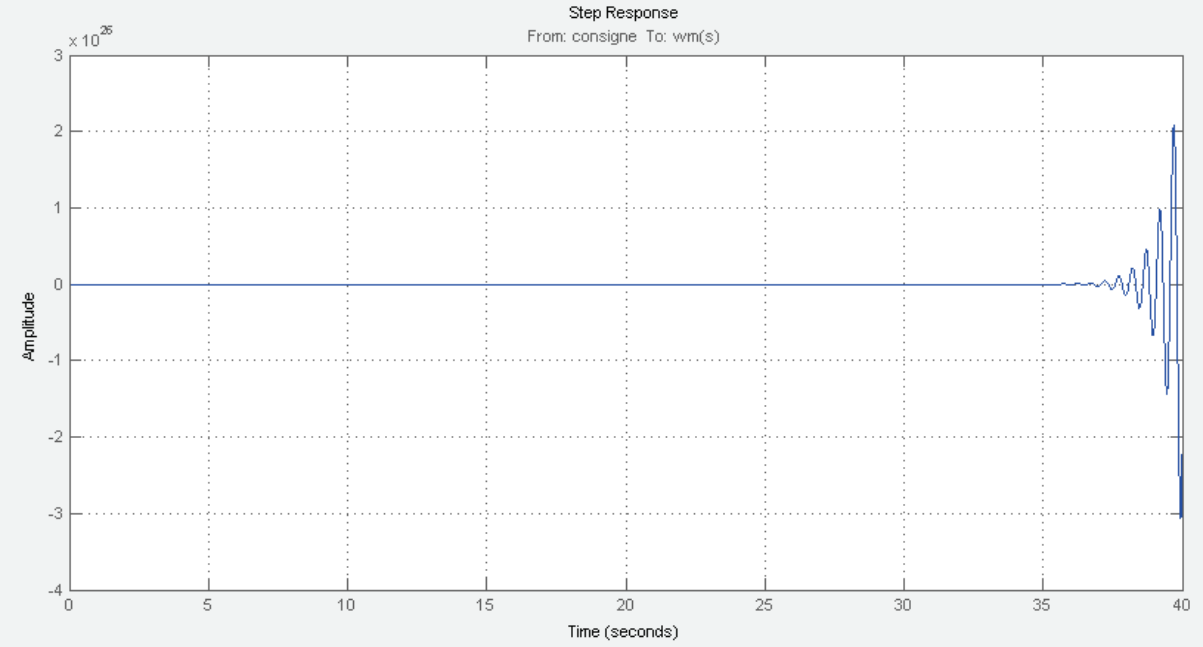


Figure 404 : visualisation de l'instabilité de la réponse indicielle

7. Ajout d'un correcteur à avance de phase (Lead)

Afin d'augmenter les marges de gain et de phase, il est possible d'ajouter un correcteur à avance de phase (Lead).

Pour cela **cliquer** avec le bouton droit de la souris, puis sélectionner **Add Pole zero/Lead.**

A l'aide du pointeur de la souris cliquer ensuite au niveau de la pulsation de coupure sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte. Le pôle du correcteur à avance de phase sera alors placé au niveau de la pulsation de coupure. Le zéro sera placé automatiquement à une pulsation inférieure.

Il est possible de visualiser les pôles et les zéros sur toutes les courbes (Figure 405).

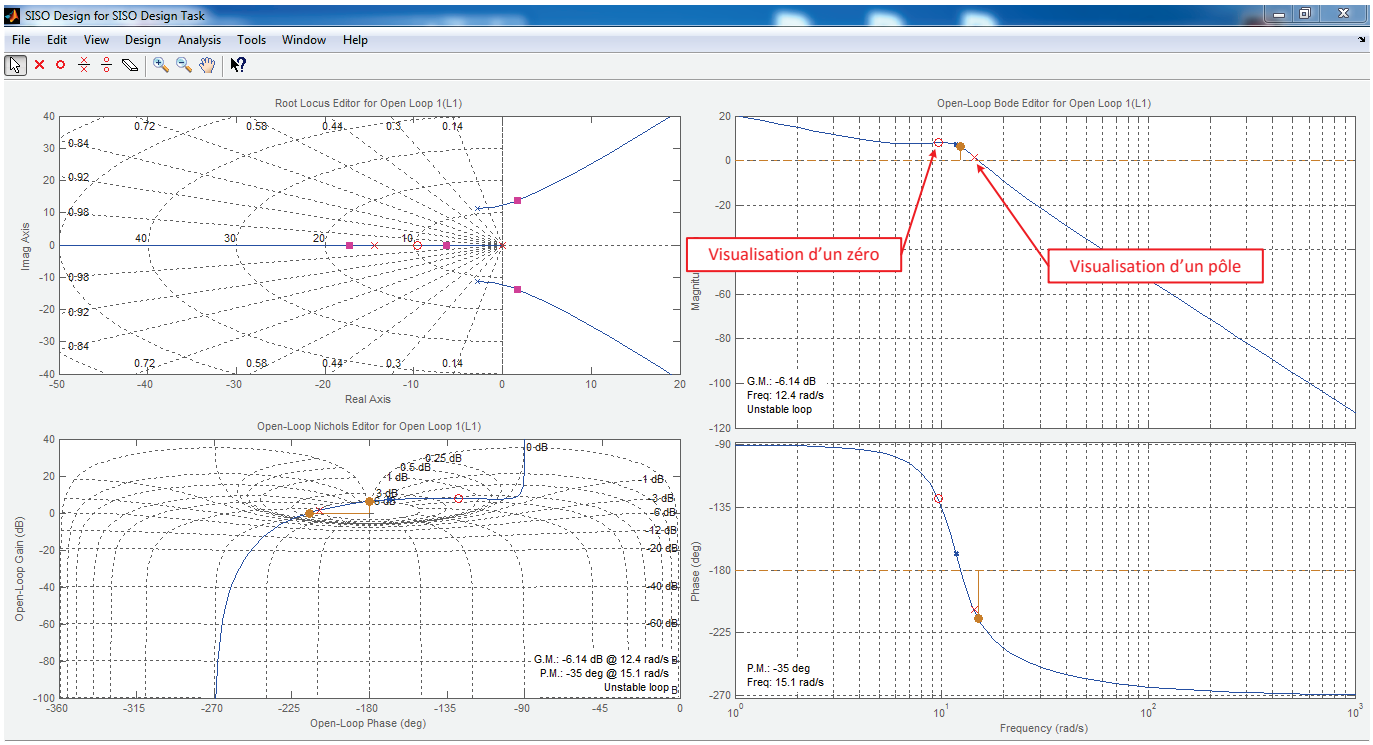


Figure 405 : visualisation des pôles et des zéros sur les courbes

Pour régler le correcteur à avance de phase, il suffit de déplacer à l'aide du pointeur de la souris le pôle et le zéro ainsi ajoutés pour obtenir une marge de gain et de phase positive afin de rendre le système stable.

Positionner le **pôle** autour de la pulsation **1000 rad/s** et le **zéro** autour de la pulsation **4 rad/s**.

Les marges de gain et de phase sont maintenant positives et le système est stable. Les pôles de la fonction de transfert en boucle fermée sont à partie réelle négative (Figure 406).

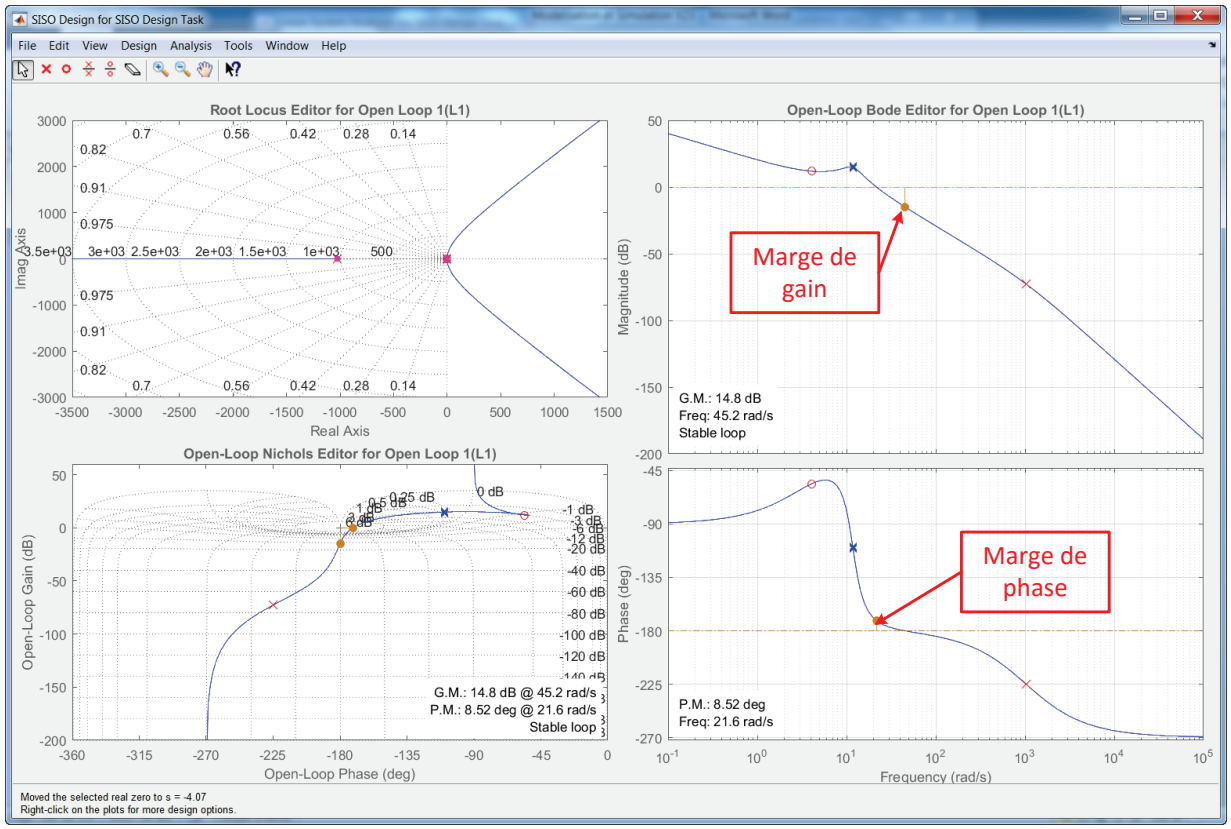


Figure 406 : stabilisation du système par ajout d'un correcteur à avance de phase

La réponse indicielle converge et la précision est bonne.

Le diagramme de Bode en gain de la fonction de transfert en boucle fermée, fait apparaître une forte résonance autour de la pulsation 22 rad/s qui témoigne de la mauvaise qualité de l'amortissement de la réponse indicielle (Figure 407).

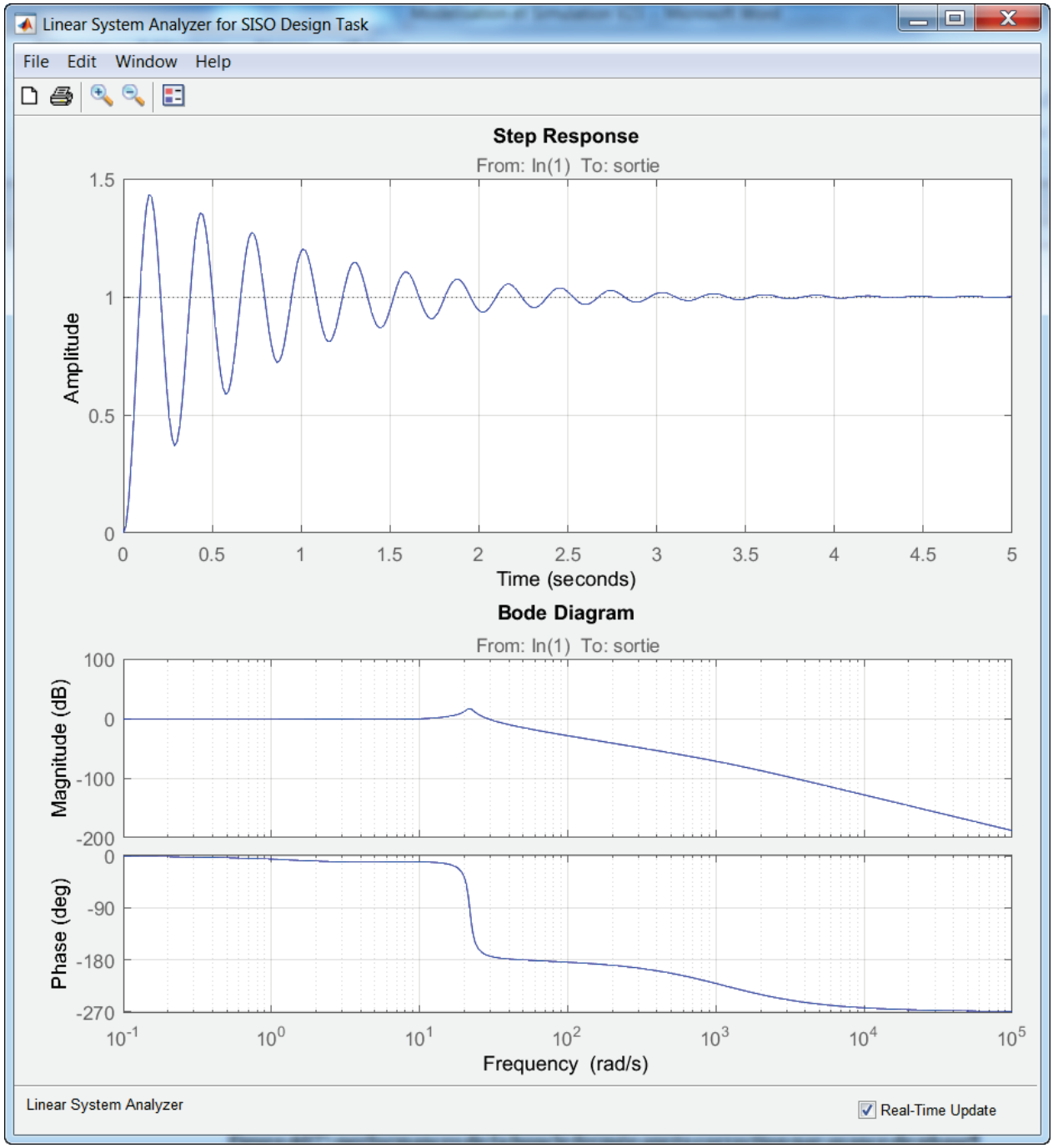


Figure 407 : performances de la boucle fermée après correction par avance de phase

Il est possible de visualiser à ce stade la fonction de transfert du correcteur dans la fenêtre **Control and Estimation Tool Manager** onglet **Compensator Design** (Figure 408)**.**

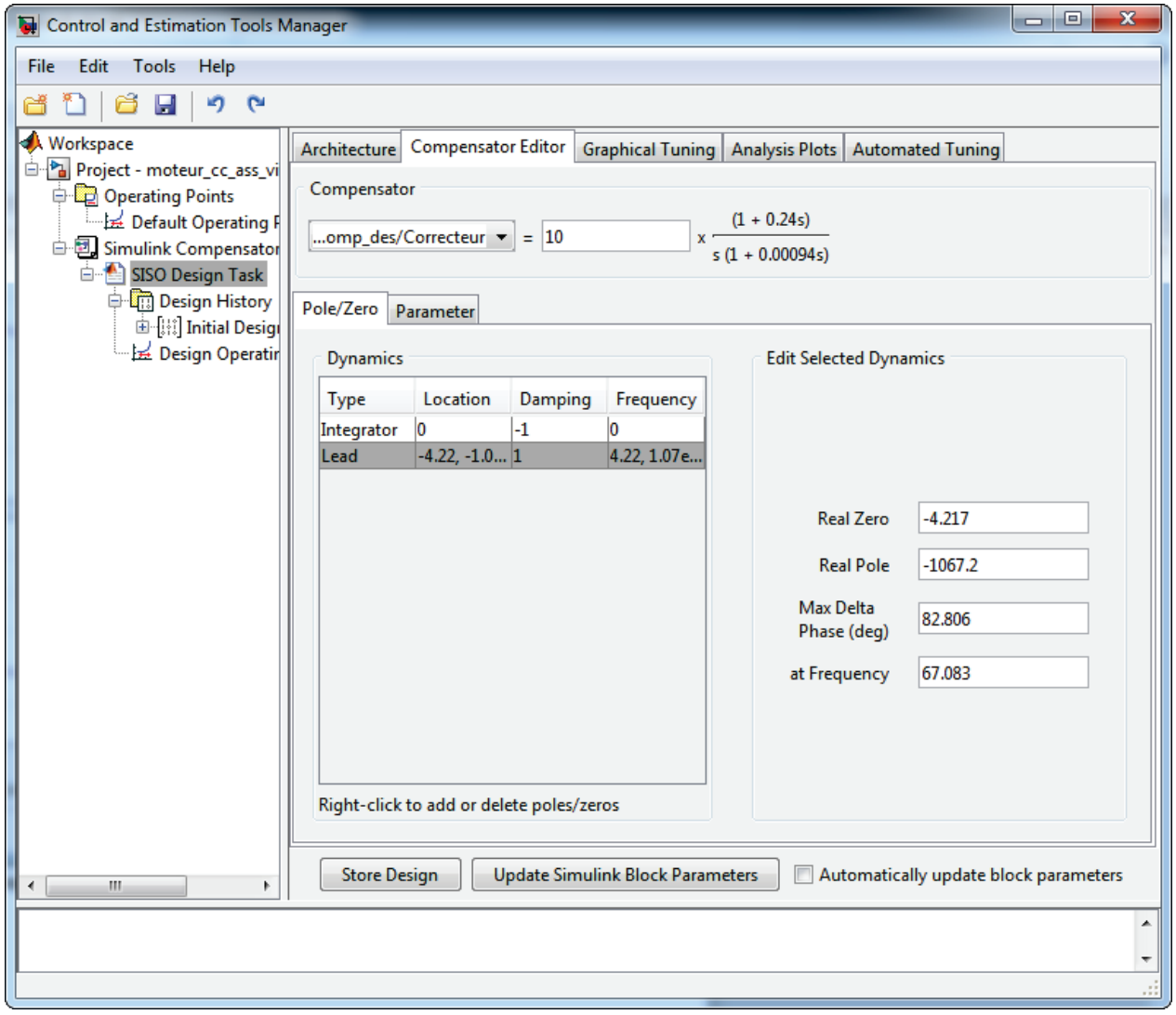


Figure 408 : visualisation de la fonction de transfert du correcteur

8. Ajout d'un filtre rejecteur (Notch)

Afin de compenser le phénomène de résonance constaté sur le diagramme de Bode de la fonction de transfert en boucle fermée (pic à la pulsation 20 rad/s). Il est possible d'ajouter un filtre rejecteur (**Notch**). Ce filtre permettra d'atténuer la résonance et de diminuer les oscillations.

Pour cela **cliquer** avec le bouton droit de la souris, puis sélectionner **Add Pole zero/Notch.**

Cliquer ensuite sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte légèrement en retrait du pic de résonnance constaté sur le diagramme de Bode de la fonction de transfert en boucle fermée (environ 10 rad/s).
Le filtre rejecteur est maintenant ajouté au correcteur et les diagrammes sont mis à jour (Figure 409).

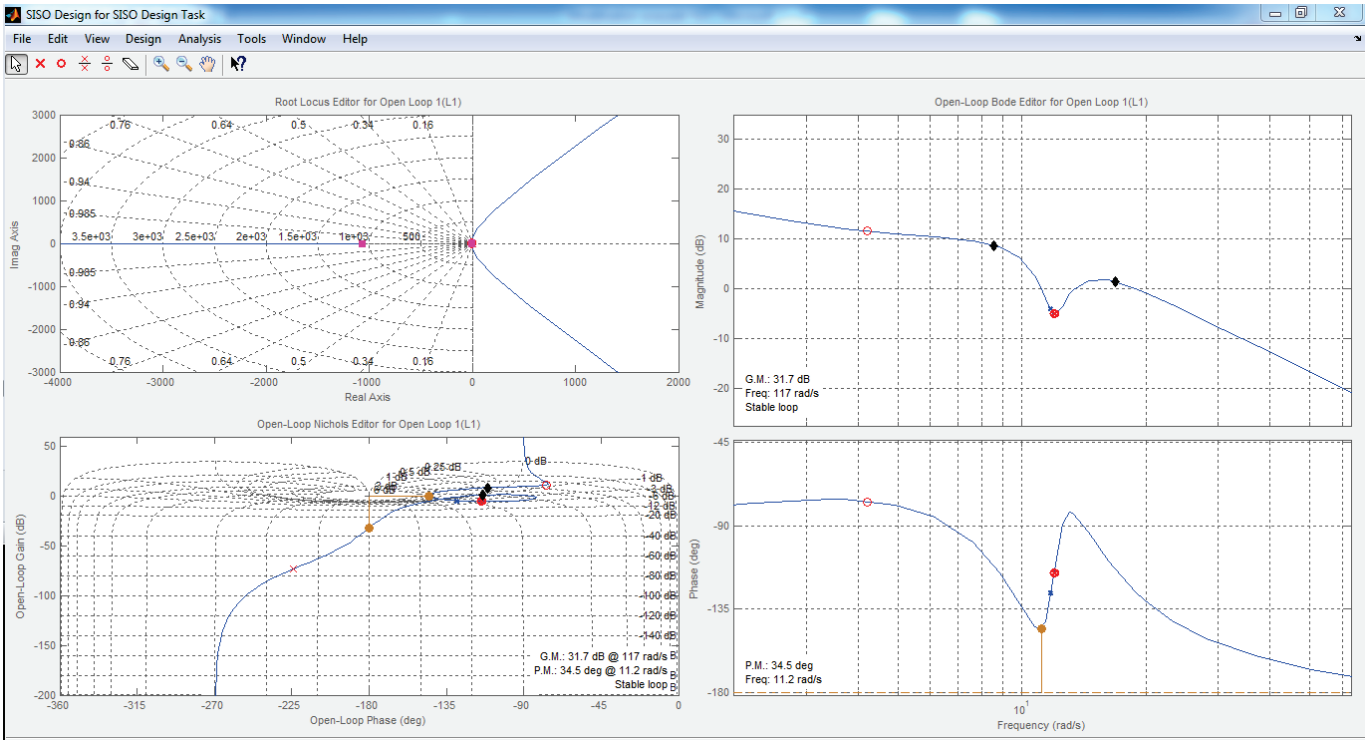


Figure 409 : ajout d'un filtre rejecteur au correcteur

Nous pouvons constater sur la Figure 410 l'influence sur la réponse indicielle et sur le diagramme de Bode de la boucle fermée. Il apparaît que le filtre n'est pas bien réglé dans la mesure où la résonnance du diagramme de Bode de la boucle fermée n'est pas totalement compensée et des oscillations importantes persistent.

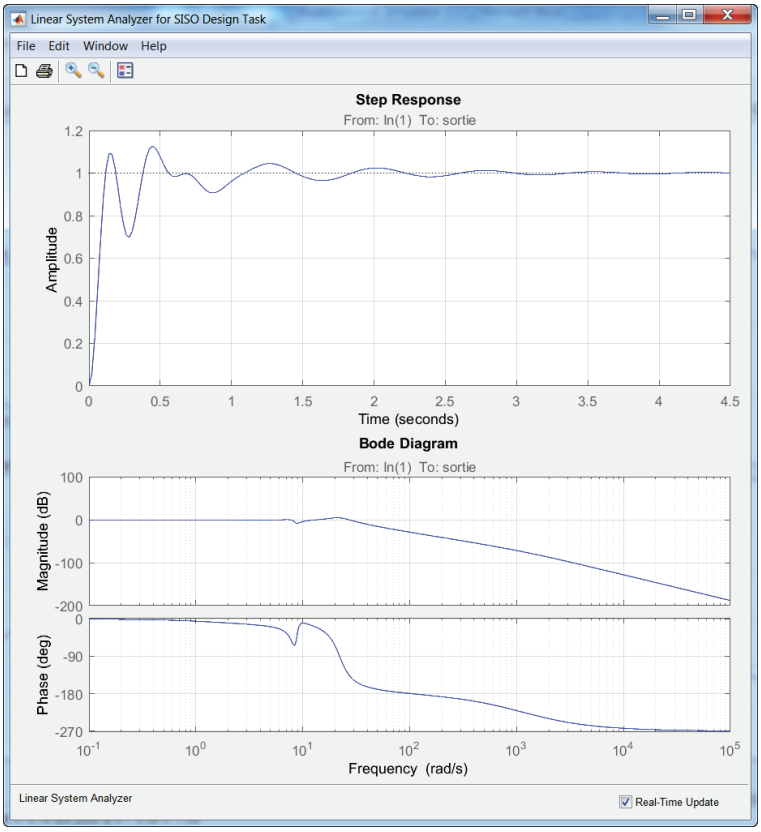


Figure 410 : influence du filtre rejecteur sur les performances de la boucle fermée

9. Réglage d'un filtre rejecteur

Pour régler le filtre rejecteur, il est possible de modifier graphiquement 3 paramètres :

- La fréquence à laquelle va agir le filtre (**Natural Frequency**)
- L'atténuation en dB que l'on souhaite obtenir (**Notch Depth**)
- La bande de fréquence dans laquelle le filtre doit agir (**Notch Width**)

Tous ces paramètres peuvent être réglés à partir du diagramme de Bode en gain de la fonction de transfert en boucle ouverte par simple glisser déposer des poignées représentant les paramètres du filtre indiqués sur la Figure 411.

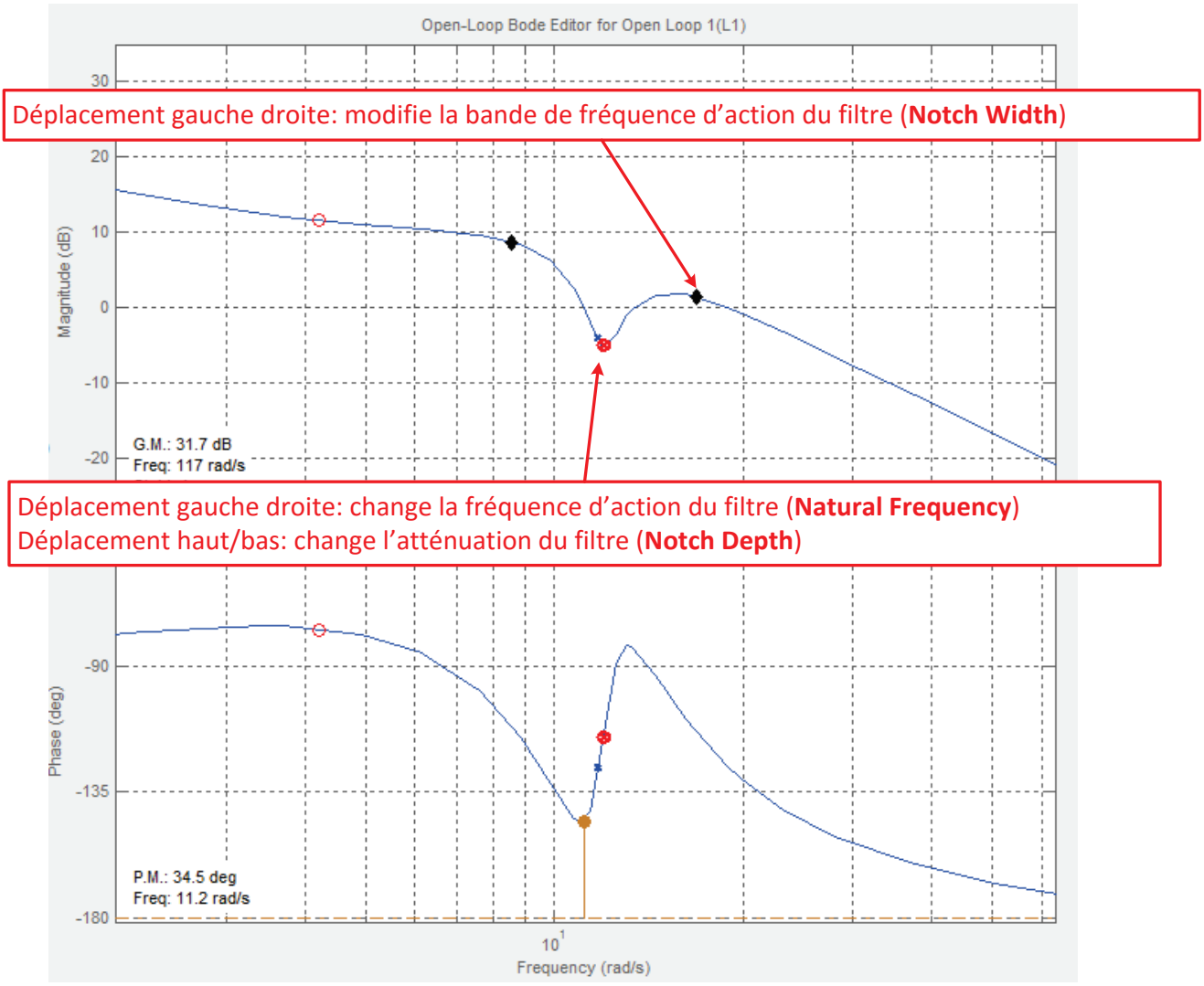


Figure 411 : réglage du filtre rejecteur

En visualisant simultanément le diagramme de Bode de la fonction de transfert en boucle fermée et le diagramme de Bode de la fonction de transfert en boucle ouverte, régler les trois paramètres du filtre de manière à voir disparaître la résonnance sur le diagramme de Bode de la fonction de transfert en boucle fermée.

Une fois les réglages effectuées les performances de la boucle fermée sont sensiblement améliorées (Figure 412).

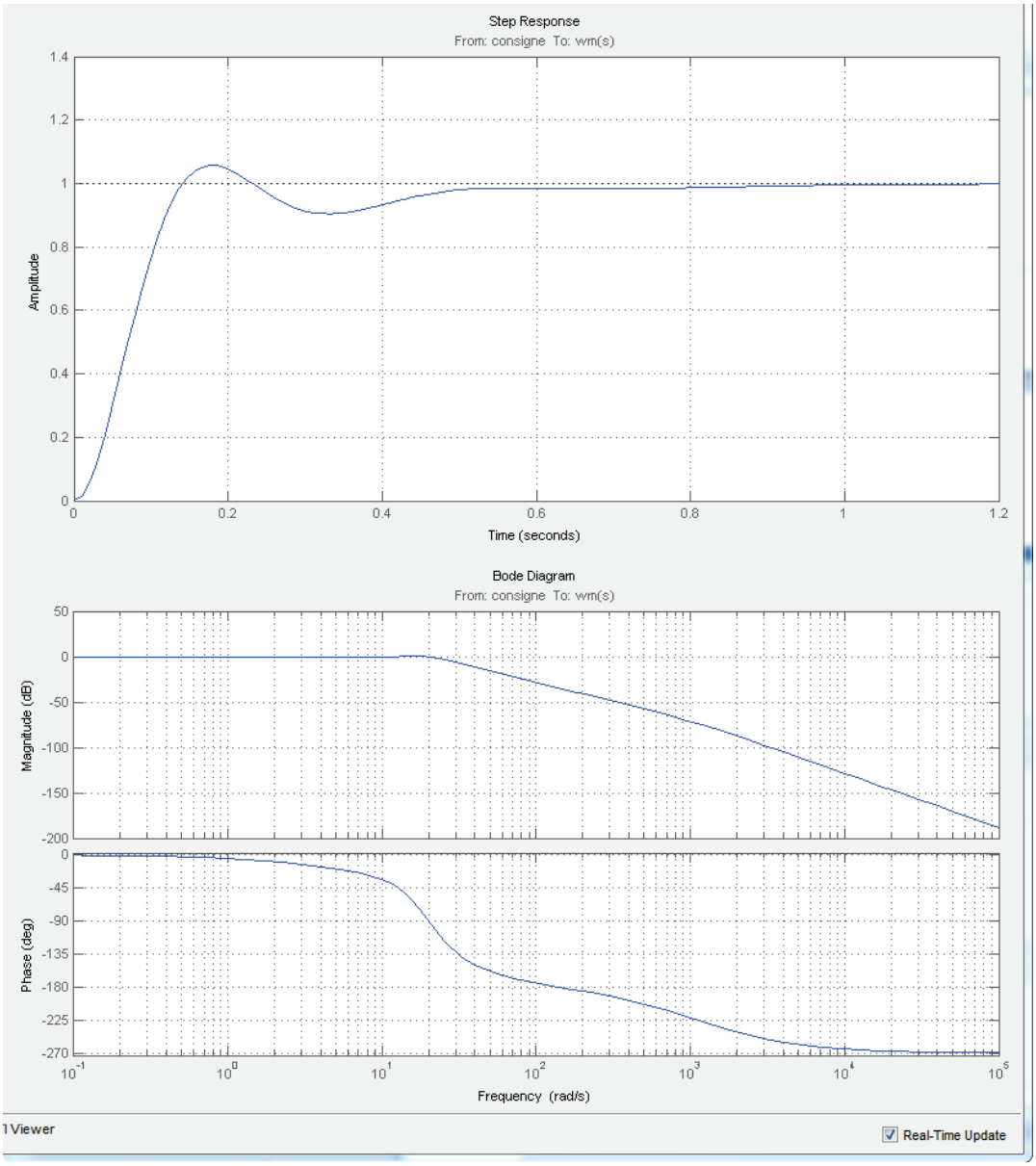


Figure 412 : performances de la boucle fermée après réglage du filtre

La réponse indicielle est précise, rapide et bien amorti. Le correcteur a permis de trouver un bon compromis entre toutes les performances recherchées.

Il est possible de visualiser la fonction de transfert du correcteur dans la fenêtre **Control and Estimation Tool Manager**, onglet **Compensator Design** et de voir la valeur des paramètres de réglages du filtre qui ont conduit à une amélioration des performances.

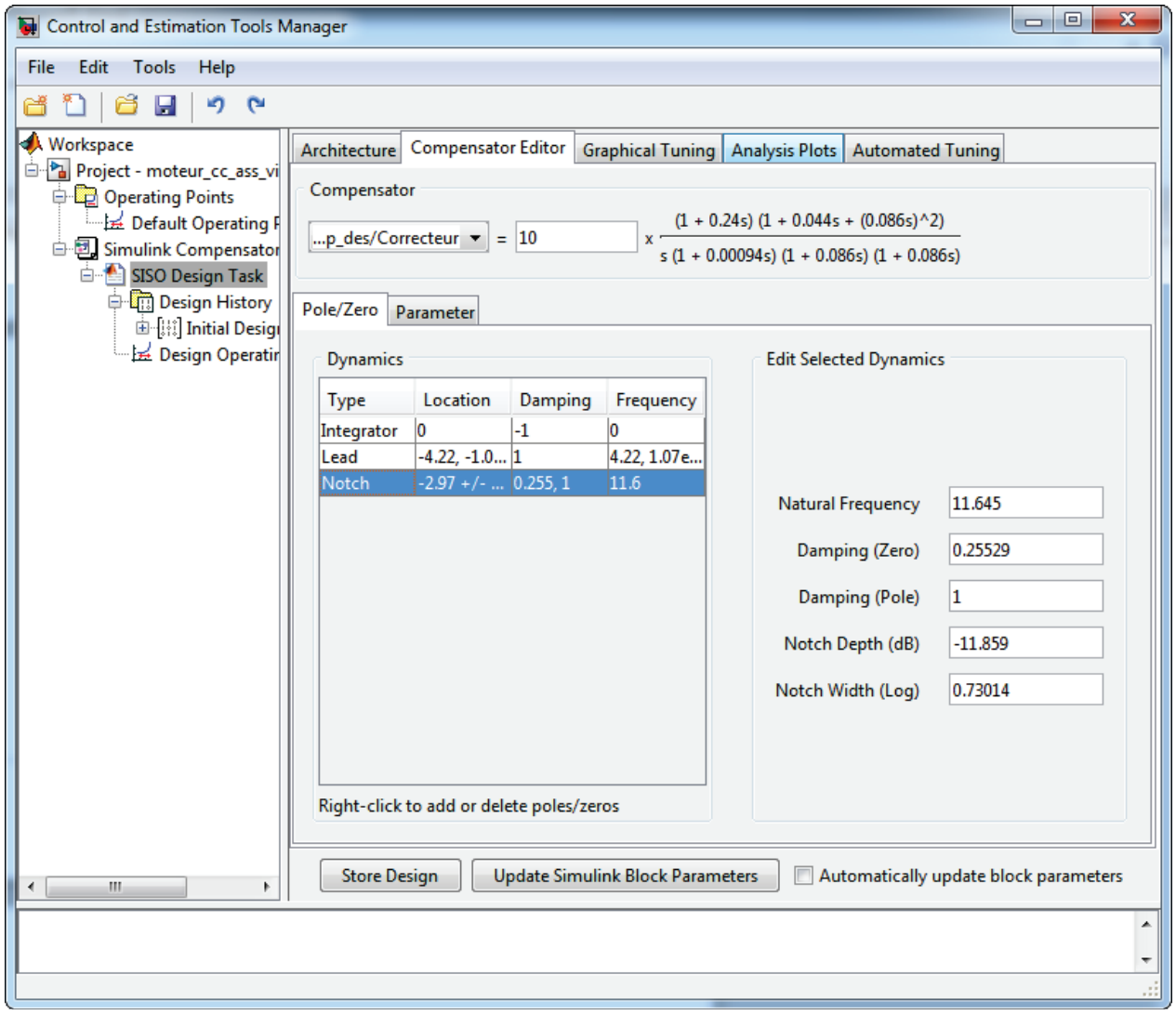


Figure 413 : visualisation de la fonction de transfert du correcteur

Il est également possible de voir l'influence du filtre rejecteur sur l'ensemble des courbes représentant le comportement de la boucle ouverte.

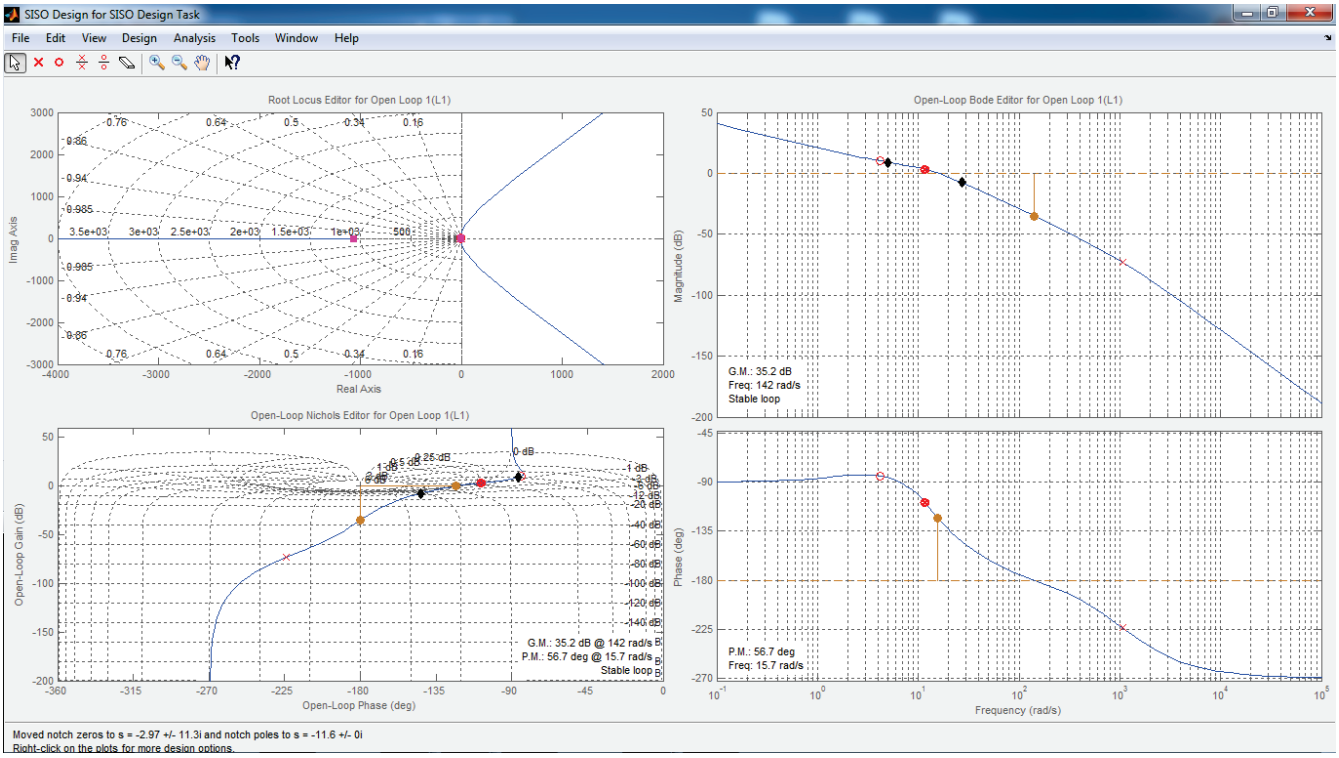


Figure 414 : influence du filtre sur le comportement de la boucle ouverte

10. Exportation de la fonction de transfert du correcteur vers le modèle Simulink

Cliquer maintenant sur **Update Simulink Block Parameter** dans la fenêtre **Compensator Editor** afin d'exporter la fonction de transfert du correcteur dans modèle Simulink.

Retourner dans le modèle Simulink, **lancer** la simulation et visualiser la réponse du système dans le scope.

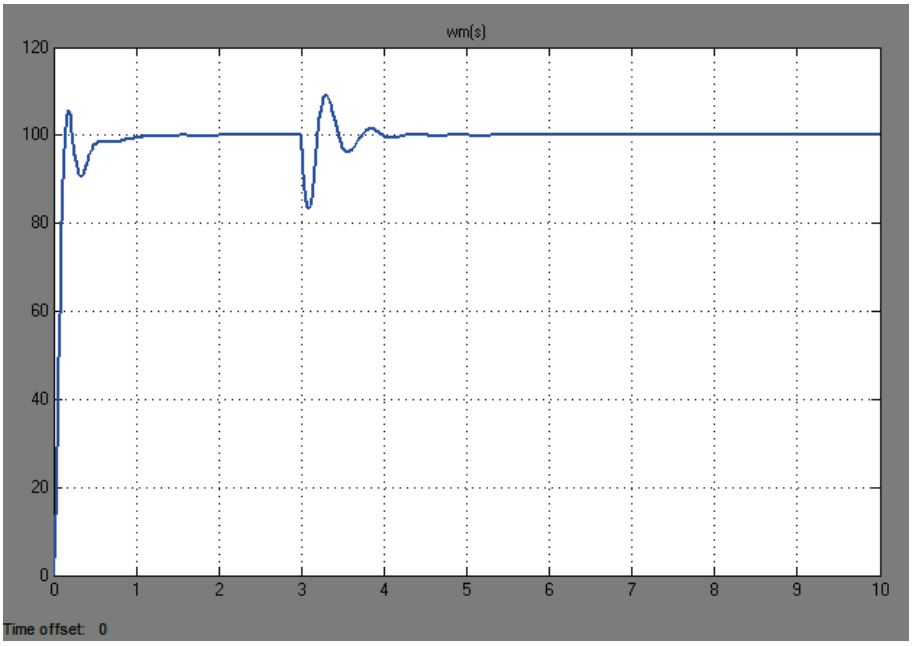
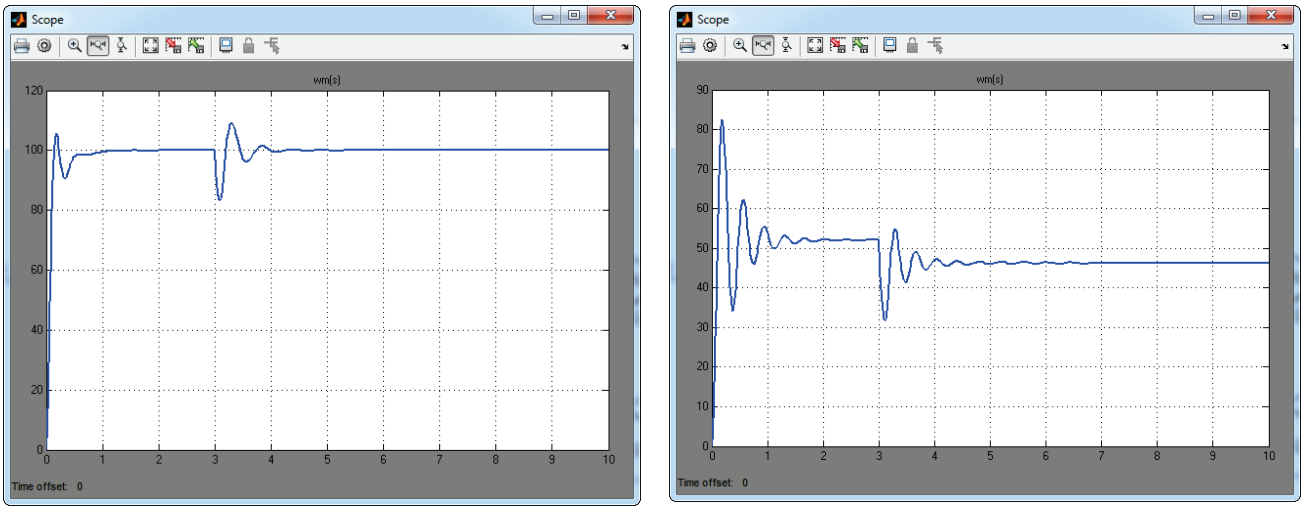


Figure 415 : réponse indicielle corrigée

La réponse indicielle corrigée (Figure 416) est précise, la rapidité et l'amortissement sont satisfaisants et la perturbation bien rejetée.

La comparaison des réponses corrigée et non corrigée permet de visualiser l'apport du correcteur.



Réponse indicielle corrigée **Réponse indicielle non corrigée**

Figure 416 : comparaison des réponses indicielles corrigée et non corrigée

Annexe 1 : paramétrage des scopes

Ouvrir le fichier « **circuit_RL.slx** » et **lancer** la simulation.

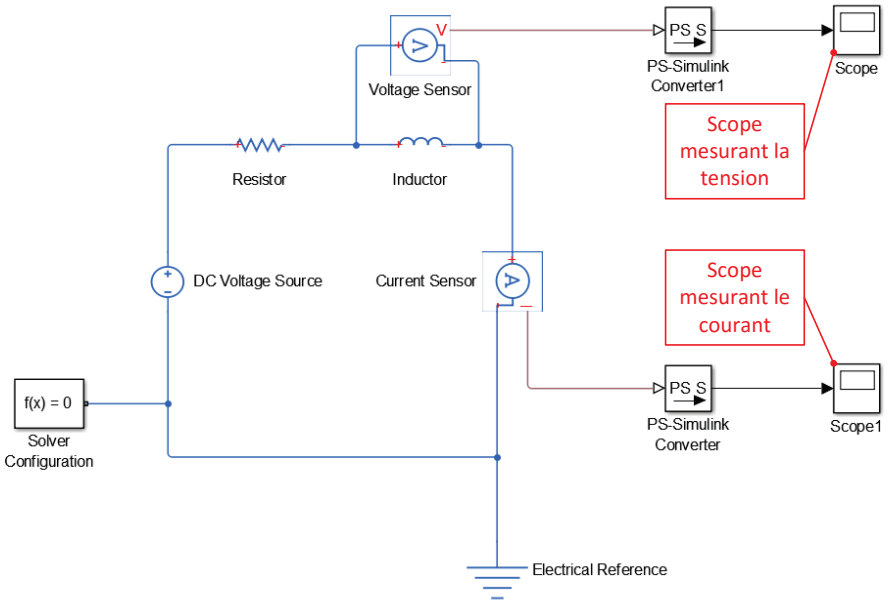


Figure 417 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine

Ouvrir le scope mesurant la tension aux bornes de la bobine et visualiser l'allure et la mise en forme de la courbe.

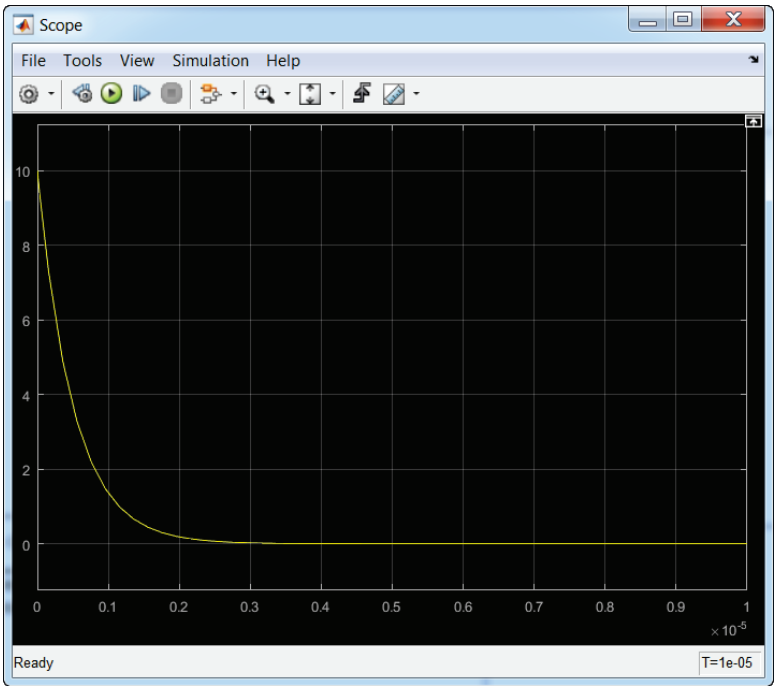


Figure 418 : mesure de la tension aux bornes de la bobine

Lors de l'ouverture du scope, la barre de commande est disponible sur la partie supérieure de la fenêtre (Figure 419).

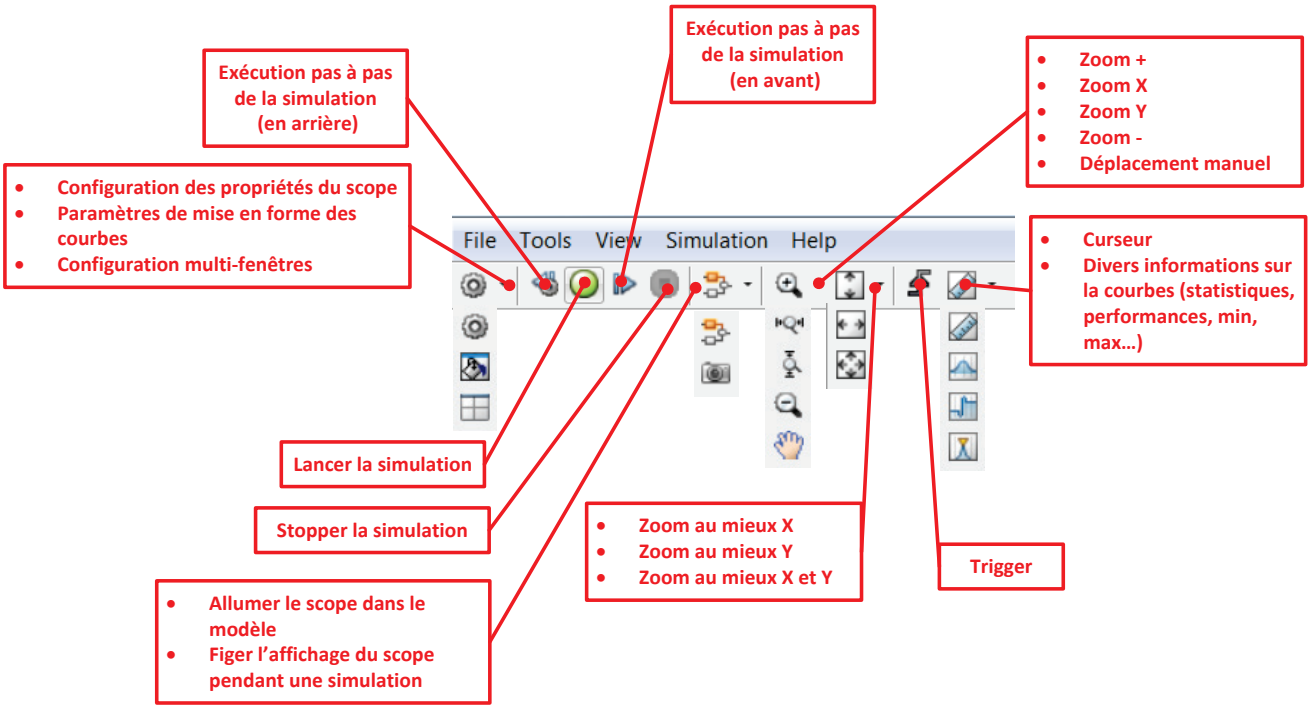


Figure 419 : barre de commande du scope

La commande la plus utile est la mise à l'échelle automatique des axes  . A la fin d'une simulation , il faut penser à l'utiliser dès l'ouverture du scope pour visualiser la courbe. Il est possible que lors de l'ouverture du scope la courbe ne soit pas dans la zone affichée par les axes et que rien ne soit visible sur le scope.

Il est possible de modifier la mise ne forme des courbes en cliquant sur l'icône  accessible depuis le menu déroulant **Configuration Properties** 

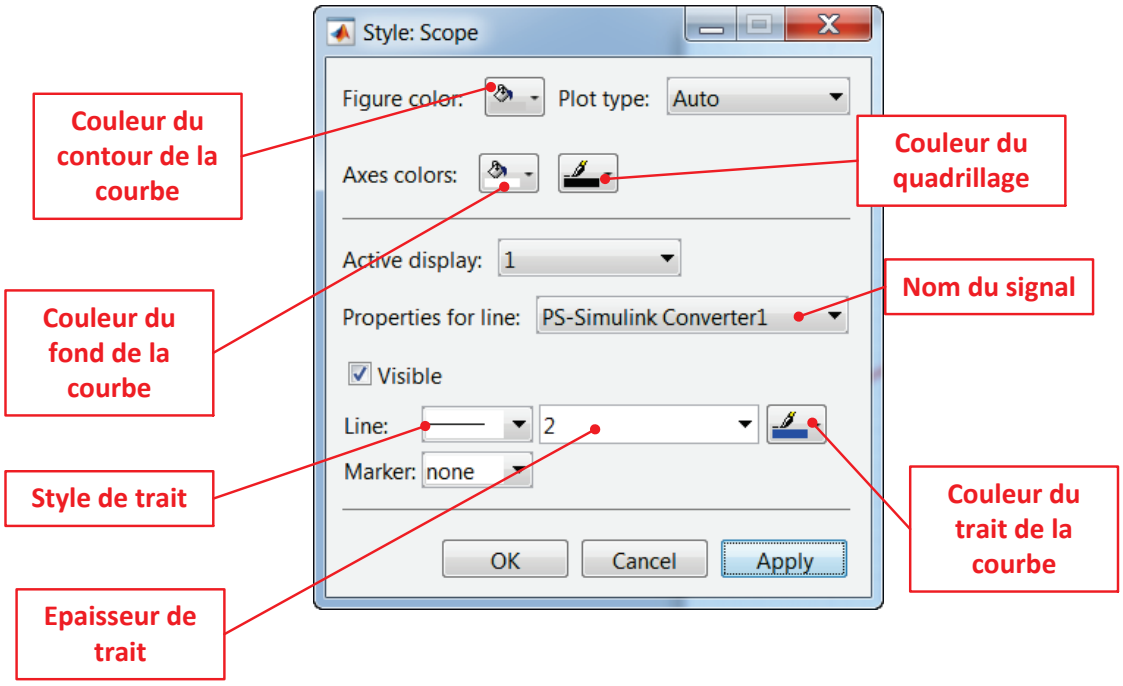


Figure 420 : mise en forme des courbes dans un scope

Modifier les paramètres de mise en forme comme indiqué sur la Figure 420 et observer le résultat obtenu (Figure 421).

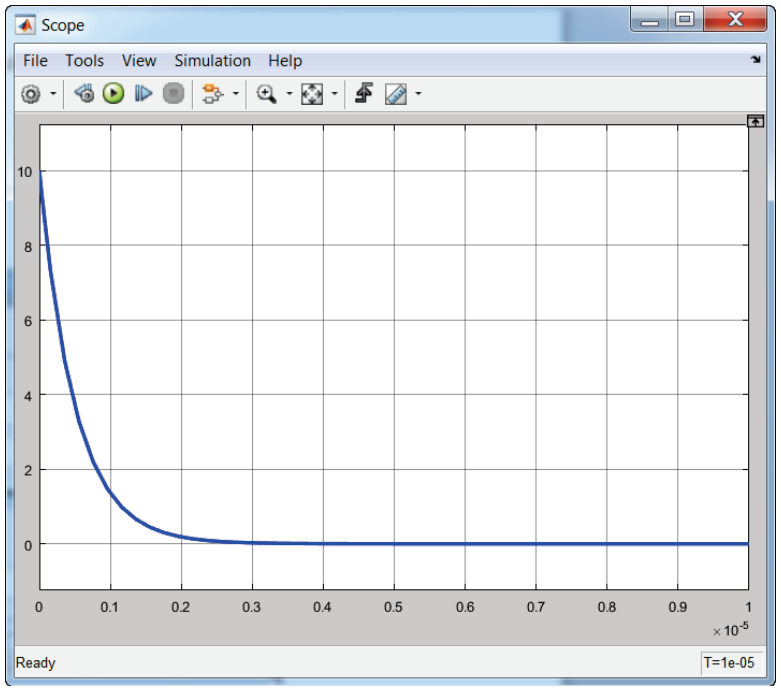


Figure 421 : modification de la mise en forme d'une courbe dans un scope

Pour afficher plusieurs courbes dans le même scope cliquer sur **Configuration Properties** et modifier le champ **Number of input ports** à **2** afin de définir deux entrées pour le scope (Figure 422).

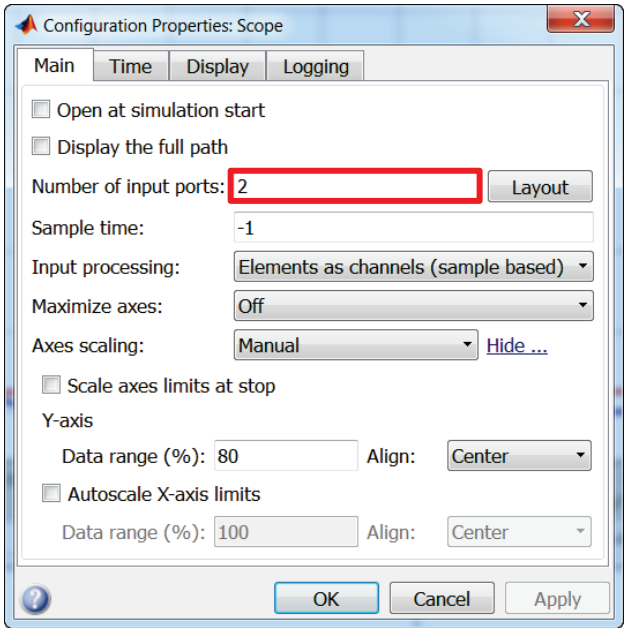


Figure 422 : modification du nombre d'entrées d'un scope

Le scope a maintenant 2 entrées. **Raccorder** la seconde entrée au signal mesurant le courant dans le circuit comme indiqué sur la Figure 423.

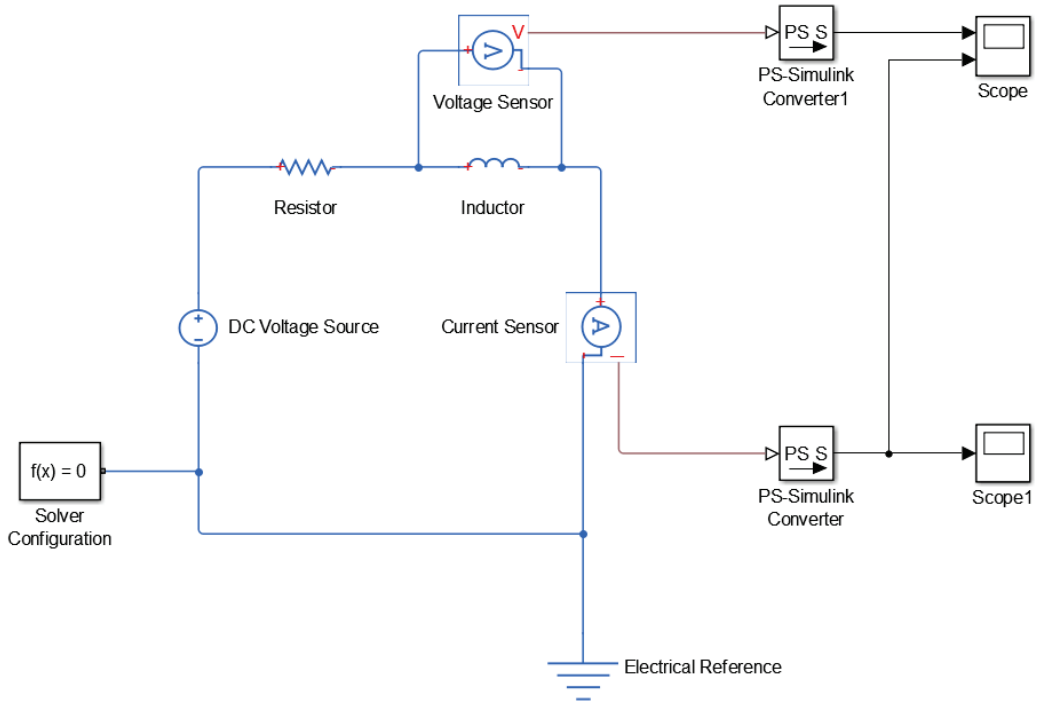


Figure 423 : raccordement d'un scope à deux entrées

Lancer la simulation et ouvrir le scope pour obtenir la courbe de la Figure 424.

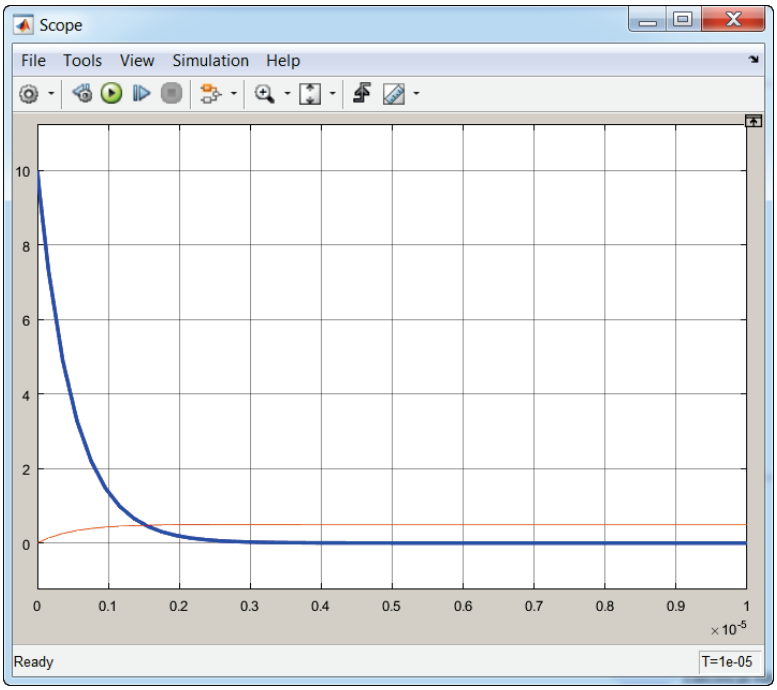


Figure 424 : visualisation des deux courbes dans le même scope

Modifier les caractéristiques de la seconde courbe en utilisant l'icône  , comme indiqué sur la Figure 425.

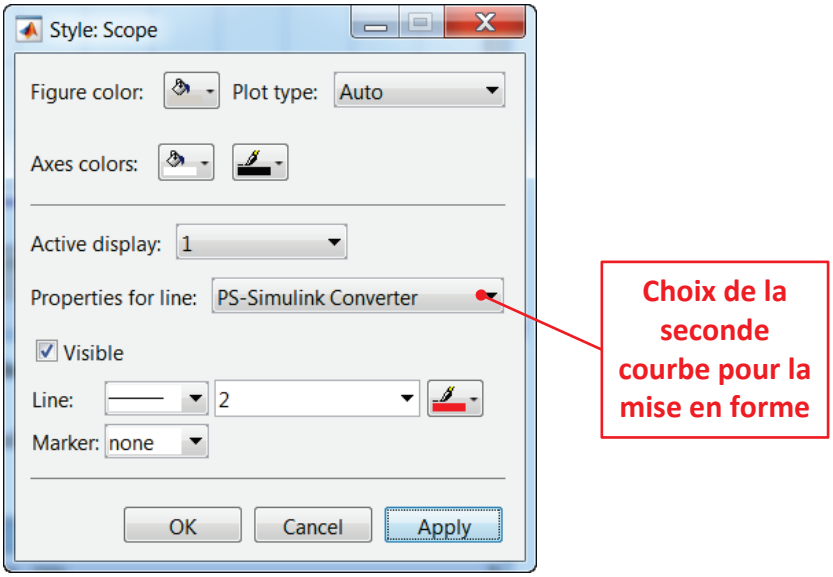


Figure 425 : modification des caractéristiques de la deuxième courbe

Vous devez obtenir la configuration de la Figure 426.

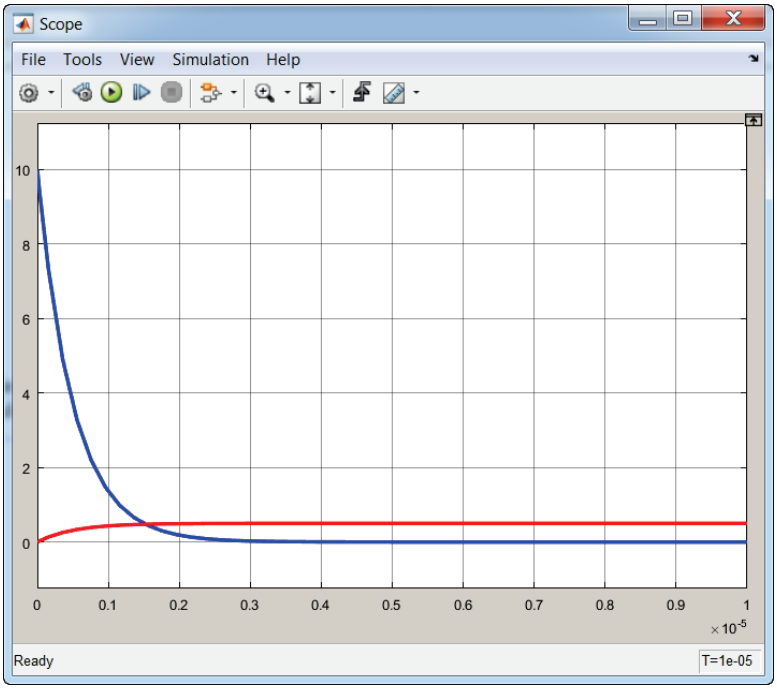


Figure 426 : modification des caractéristiques de la deuxième courbe

Si vous souhaitez afficher ces deux courbes dans des graphes différents, cliquer sur l'icône **layout** et choisir la disposition des courbes que l'on souhaite (Figure 427).

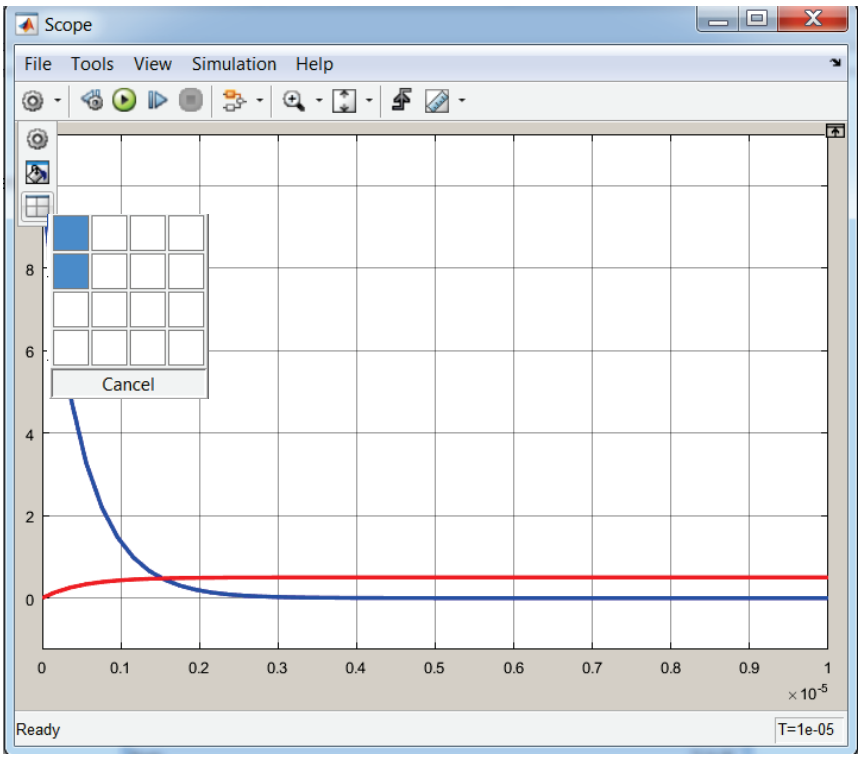


Figure 427 : modification du nombre de fenêtre dans le scope

Après mise à l'échelle des courbes à l'aide de la commande  , on obtient l'affichage représenté sur la Figure 428.

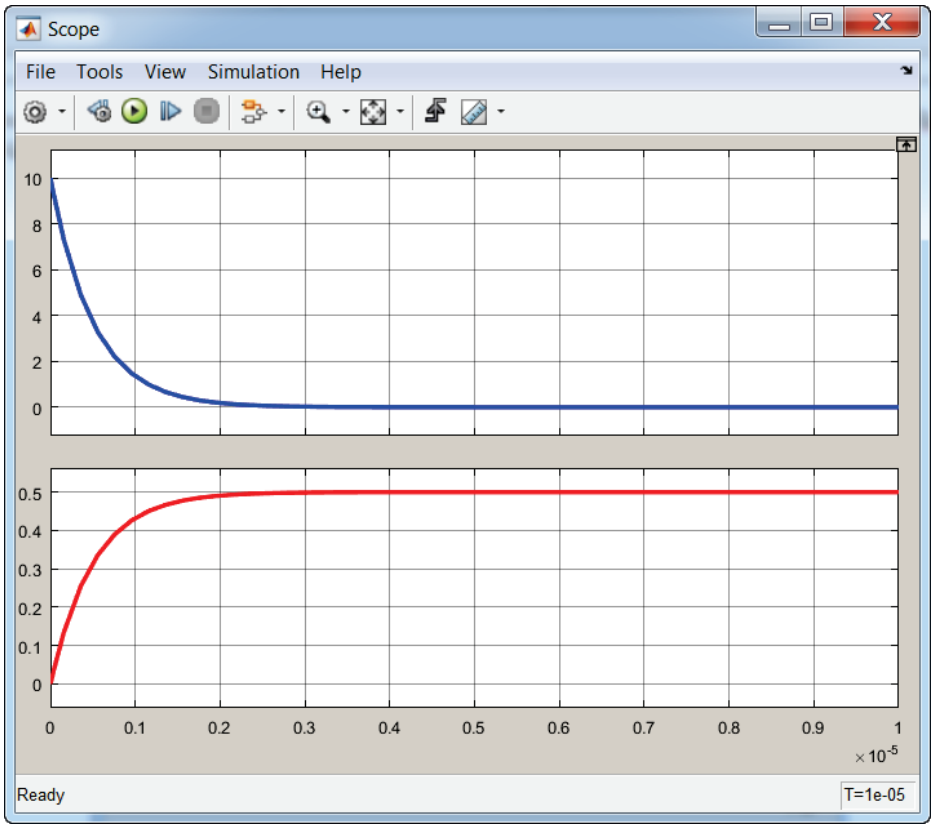


Figure 428 : affichage des courbes dans un scope sur plusieurs fenêtres

Un outil curseur est très pratique à utiliser dans les scope, pour l'ouvrir cliquer sur l'icône **Cursor Measurement**  afin de faire apparaître les curseurs. Il est alors possible d'avoir accès à tous les paramètres de mesure utiles ΔT, ΔY, coefficient directeur de la droite reliant les deux curseurs…(Figure 429).

Pour plus d'information sur l'utilisation des scopes vous pouvez utiliser l'aide de MATLAB.

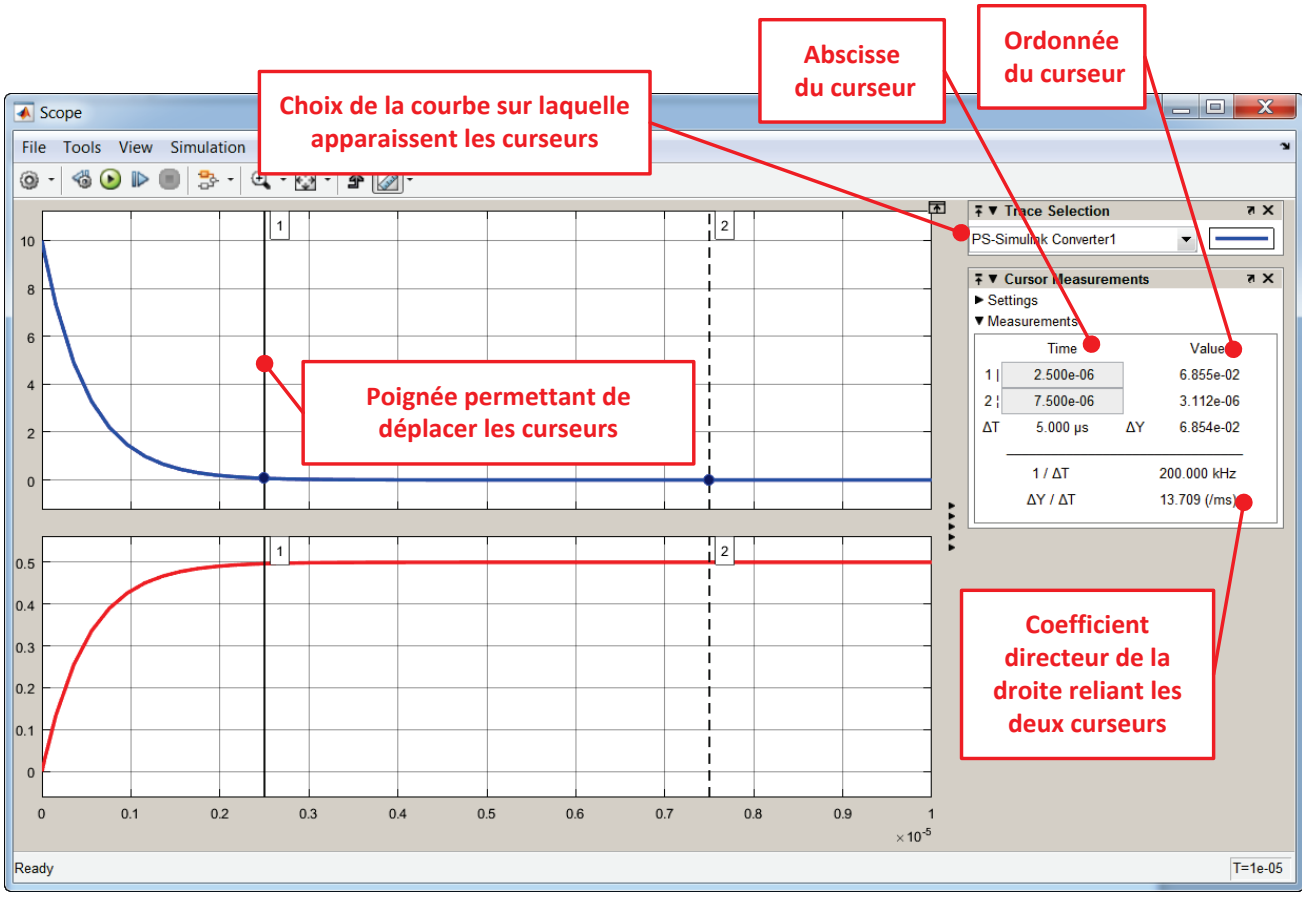
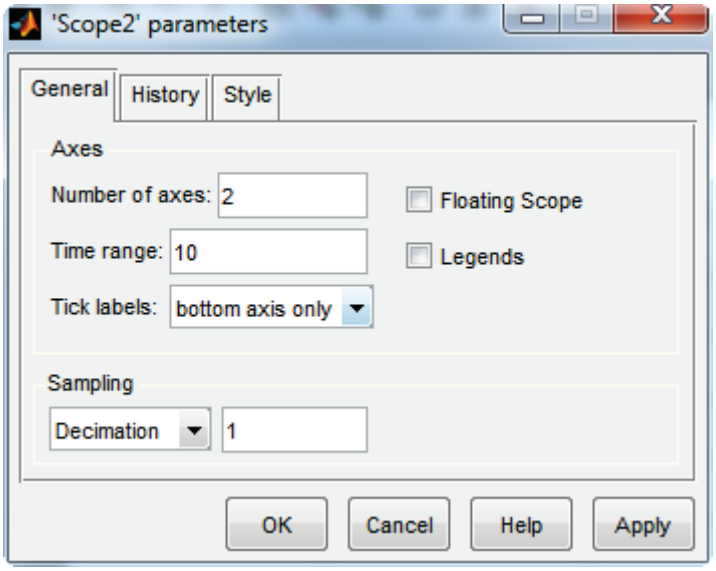


Figure 429 : utilisation des curseurs dans un scope

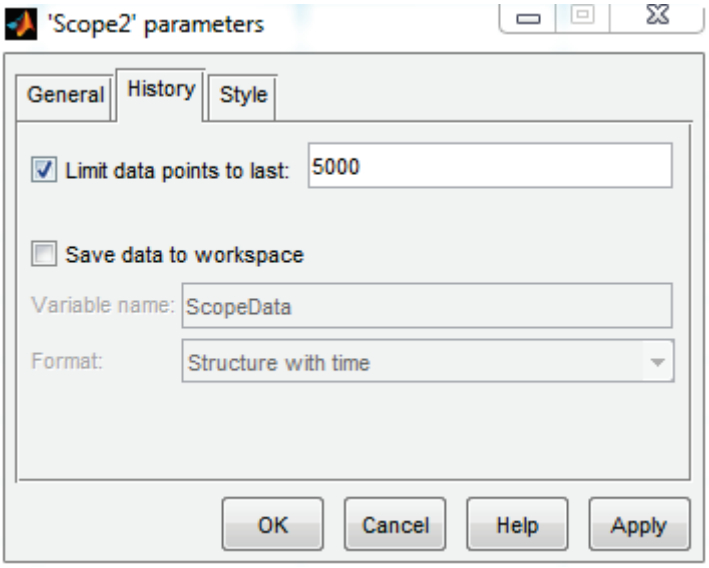
Pour paramétrer un scope à plusieurs entrées, faire un double-clique sur le scope et cliquer sur **Parameters**  .



Dans l'onglet **General, c**hoisir le nombres d'entrées du Scope dans le champ **Number of axes.**

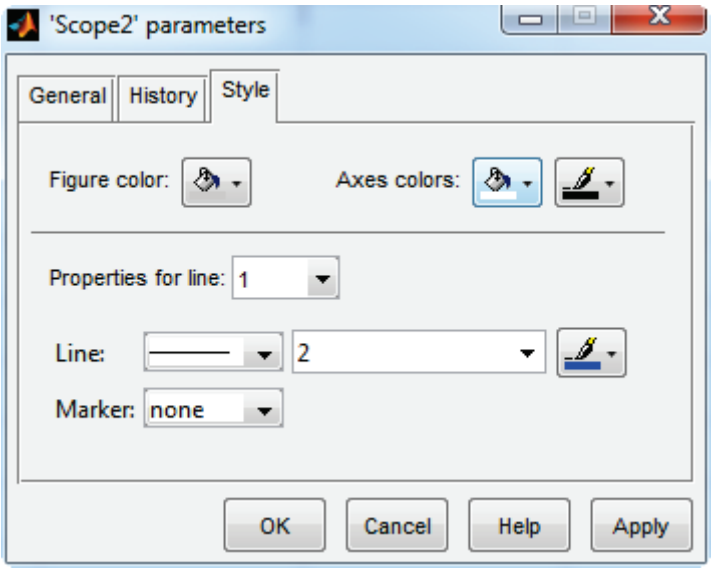
Il est également possible de demander l'affichage d'une légende en cochant la case **Legends**.

Dans l'onglet **History**, il est possible de spécifier le nombre de points maximal que le scope enregistrera. Il est souvent nécessaire de décocher cette case afin de pouvoir visualiser l'ensemble des résultats d'une simulation.



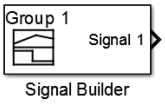
A partir de cet onglet, il est également possible de sauvegarder les résultats de la simulation dans le **workspace** sous la forme d'une matrice que l'on pourra exploiter avec **MATLAB**.
Pour cela cocher la case **Save data to workspace** et indiquer le nom de la variable et son format (par défaut **Structure with time** permet de sauvegarder à la fois les valeurs de tous les points des signaux, mais également des informations comme le nom du signal, la dimension des matrices…)

Dans l'onglet **Style**, il est possible de choisir les couleurs d'affichage des courbes et du fond, d'ajouter des marqueurs, de modifier l'épaisseur du tracé…



Annexe 2 : utilisation du signal builder

Le « signal builder » est un générateur de signaux que l'on peut paramétrer manuellement. Ce bloc est très utile pour imposer des lois de commandes.

Fonction du composant	Représentation	Bibliothèque
Source de signal paramétrable		Simulink/Sources

Glisser-déposer un bloc « Signal Builder » depuis la bibliothèque **Simulink** et **ouvrir** ce bloc.

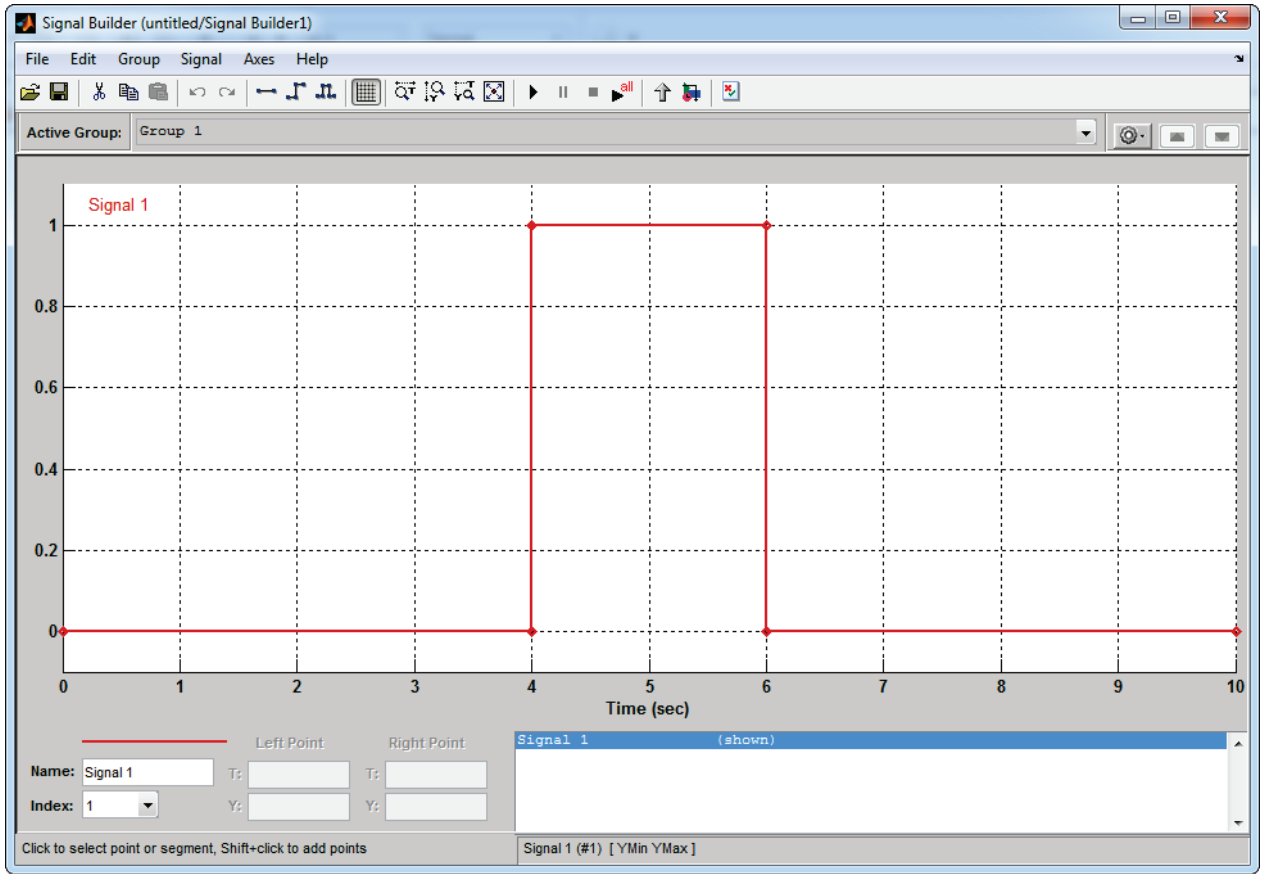


Figure 430 : fenêtre de paramétrage du Signal builder

La barre de commande du signal buider permet d'accéder à de nombreuses fonctions.

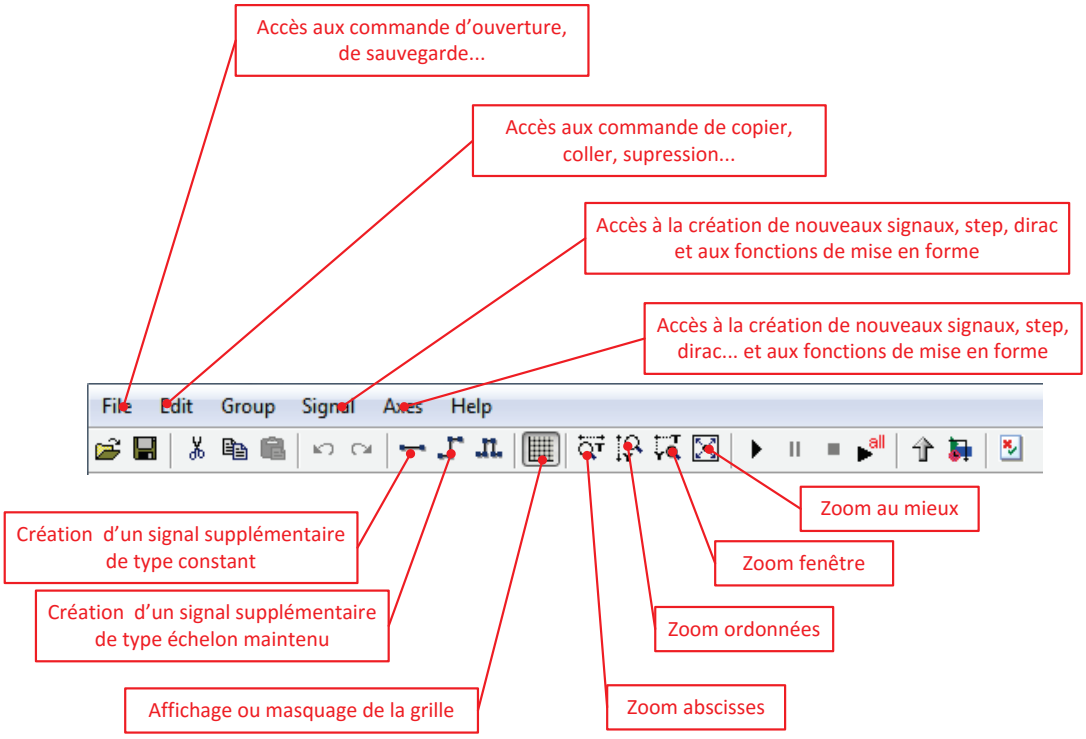


Figure 431 : barre de commande du signal builder

Dans un premier temps il faut indiquer la durée du signal. Pour cela sélectionner **Signal/Change Time Range** accessible depuis la barre de commande.

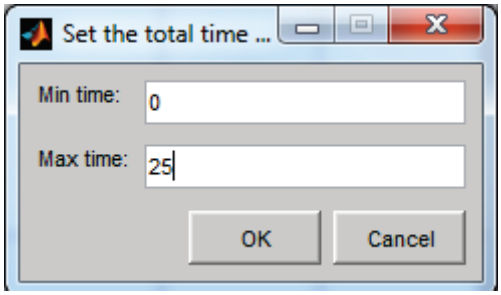


Figure 432 : choix de la durée du signal dans le « signal builder »

Choisir une durée de signal de 25 s.

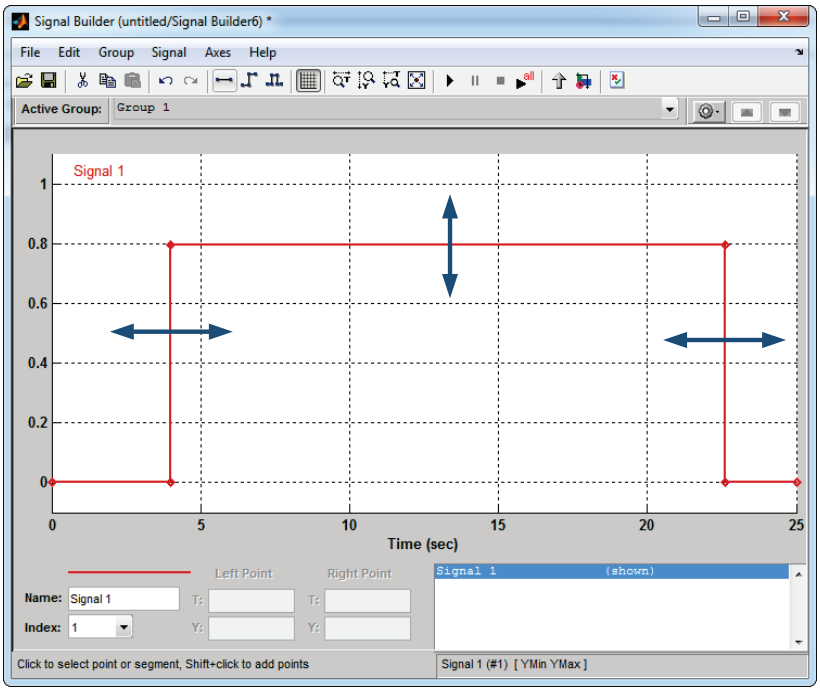


Figure 433 : déplacement manuel des segments dans le « signal builder »

Il est possible de déplacer manuellement les segments qui composent le signal.
Avec le curseur de la souris, en utilisant le **glisser déposer**, déplacer horizontalement les segments verticaux et verticalement les segments horizontaux.

Pendant le déplacement, on peut visualiser dans les cases « left point » et « right point » situées sous le signal la position du segment.
Si on fait un simple clique gauche sur un segment pour le sélectionner, il est possible de rentrer des positions exactes dans les cases « left point » et « right point ».
Il est également possible de déplacer uniquement un point pour obtenir des signaux de type rampe.

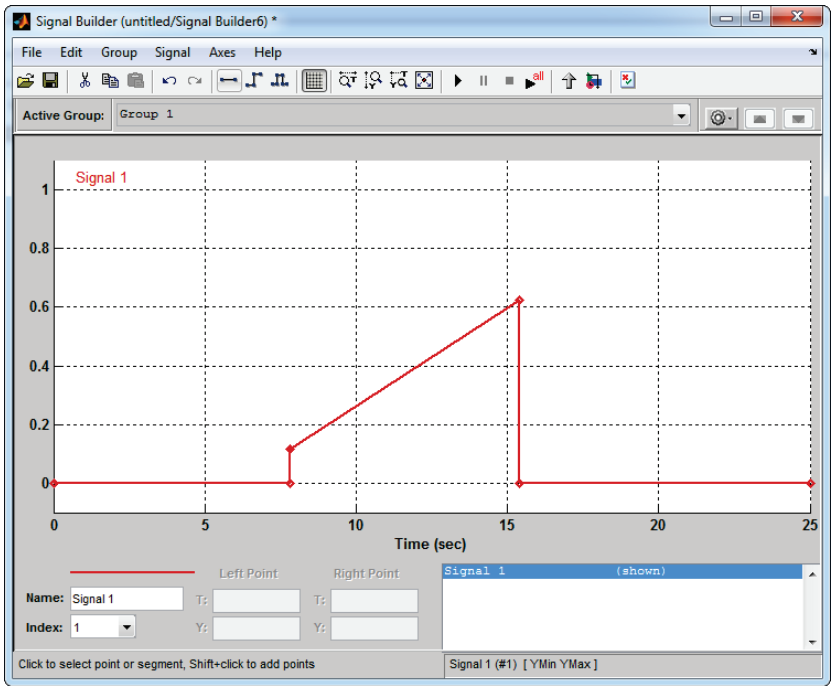


Figure 434 : déplacement d'un point dans le « signal builder »

Le principe de construction d'un signal quelconque est de créer des points qui seront reliés par des segments que l'on pourra ensuite déplacer librement.

Pour créer de nouveaux points, il faut utiliser un **shift+clique gauche** en plaçant les nouveaux points directement sur le signal.

Placer approximativement des nouveaux points conformément à la Figure 435.

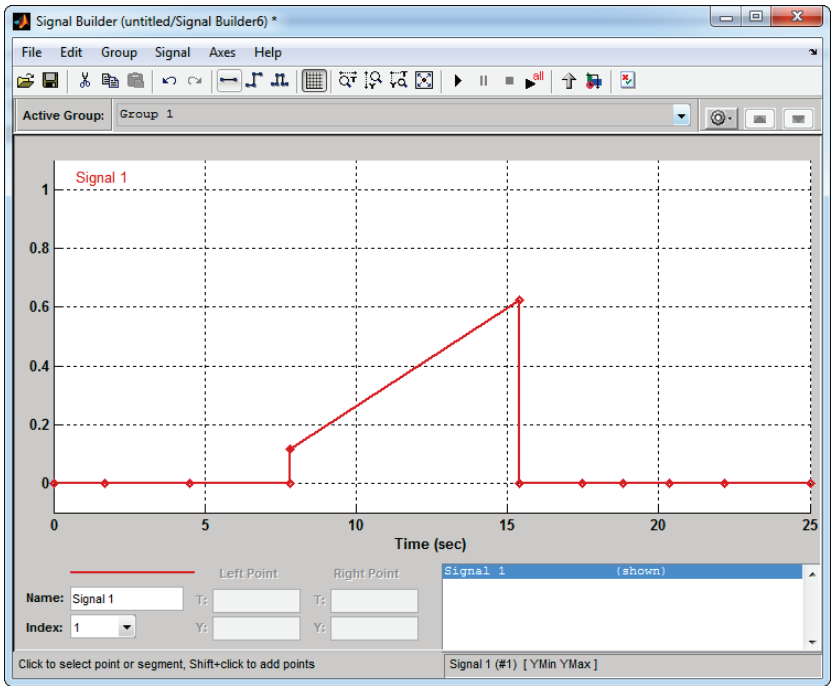


Figure 435 : placement de nouveaux points sur le signal

Déplacer les points et les segments pour obtenir approximativement le signal de la Figure 436.

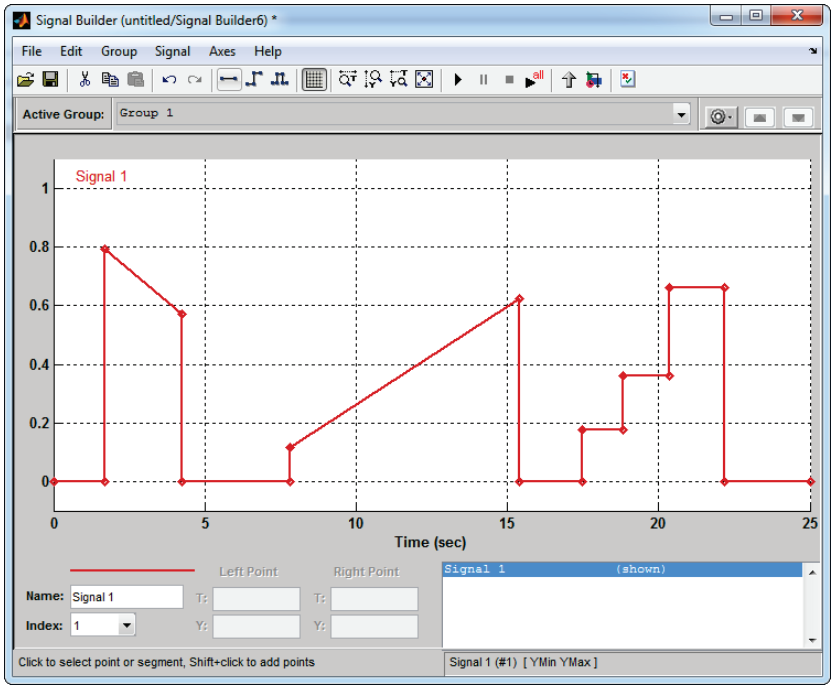
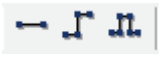


Figure 436 : mise en forme du signal par déplacement de segments et de points

Il est également possible de créer un bloc permettant de générer plusieurs signaux.

Cliquer sur l'un des trois signaux  accessible depuis la barre de commande et choisir d'introduire de nouveaux siganux.

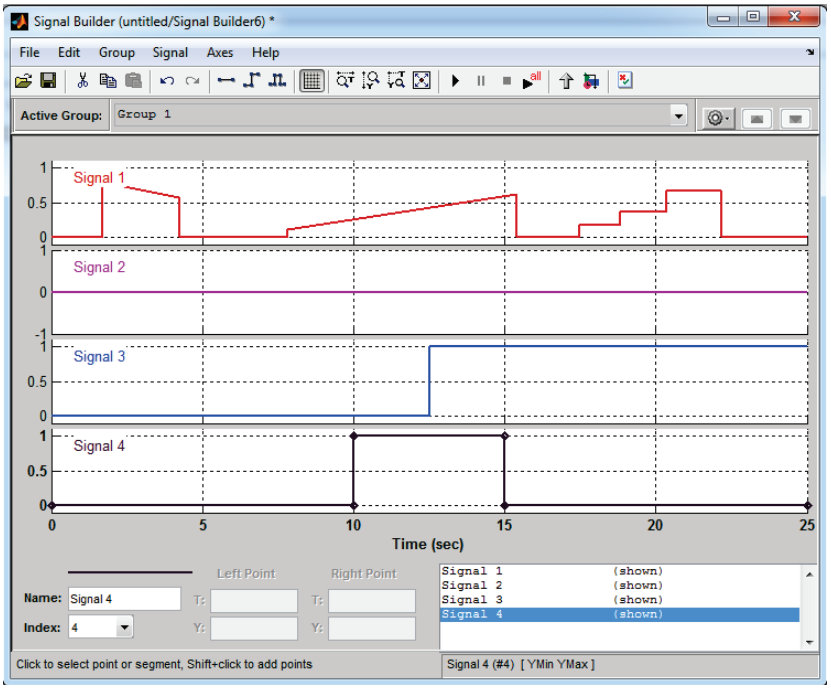


Figure 437 : génération de plusieurs signaux

Le bloc comporte maintenant 4 sorties.

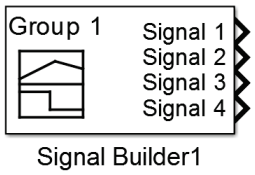


Figure 438 : visualisation des sorties multiples du « signal builder »

Table des figures