

Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB - Simulink (R2020b)

pour l'étudiant et l'ingénieur

Quatrième édition

Introduction au Model Based Design



- **MATLAB**
- **Simulink**
- **Simscape**
 - domaine électrique
 - domaine mécanique
 - domaine thermique
 - domaine hydraulique
- **Stateflow**
 - diagramme d'états
- **MATLAB Online**
- **MATLAB Mobile**



*Contrôle commande des systèmes asservis, réglage des correcteurs
Ingénierie numérique en langage MATLAB, les algorithmes fondamentaux*

Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB/Simulink R2020b 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 des Sciences Appliquées (INSA) de Lyon
MASTER en conception de structures Aéronautiques et Spatiales (ISAE-SUPAERO)
Agrégé de Sciences de l'Ingénieur
Ancien élève de l'Ecole Normale Supérieure 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/Simulink.

Vos remarques et vos suggestions seront les bienvenues et me permettront de faire évoluer et d'améliorer cet ouvrage. Cette version est la quatrième édition du livre et fait suite aux éditions de 2013, 2015 et 2018.

Toute utilisation, même partielle, du contenu de ce document devra obligatoirement faire référence à l'ouvrage et à son auteur. L'hébergement sur un site internet des fichiers contenant le livre numérique et les modèles associés est soumise à l'autorisation de l'auteur.



Ivan LIEBGOTT @ 2020

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 maîtrise 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-phérique. 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 modèles validés qui reproduisent fidèlement le comportement des systèmes réels.

Cet ouvrage vous présente une approche de la modélisation multi-phérique 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 2020b.

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 illustré par de très nombreux exemples dans différents domaines de la technologie (électrique, hydraulique, mécanique, thermique...) 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 (Foundation Library)
- Simscape Fluids
- Simscape Multibody
- Simscape Electrical
- Stateflow

Dans cette quatrième édition, un tout nouveau chapitre est consacré à l'ingénierie numérique en langage MATLAB. Vous pourrez trouver les algorithmes fondamentaux de l'analyse numérique (dérivation numérique, intégration numérique, résolution d'équations stationnaires, filtrage d'un signal, résolution d'équations différentielles, résolution de systèmes d'équations...). Pour chaque chapitre des exemples concrets d'utilisation de ces algorithmes sont présentés. Tous les scripts et les fonctions définis dans ce chapitre peuvent être facilement réutilisés et adaptés afin de gagner un temps précieux dans vos projets. La programmation en langage MATLAB s'avérera très utile en complément de l'utilisation des outils graphiques présentés également dans l'ouvrage.

Deux nouveaux chapitres sont consacrés à l'utilisation de MATLAB/Simulink Online et de MATLAB Mobile, vous pourrez découvrir les nouvelles méthodes de travail et de partage que proposent ces outils innovants. Prenez le temps de les découvrir.

L'ouvrage explore le potentiel des outils de modélisation et de programmation indispensables au déroulement d'un projet industriel de conception de système. Vous pourrez en percevoir toute la pertinence et l'exploiter en fonction des besoins spécifiques que vous rencontrerez dans votre démarche de conception et de modélisation.

Bonne lecture,

Ivan LIEBGOTT

**Modélisation et Simulation
des Systèmes Multi-Physiques
avec MATLAB/Simulink R2020b
pour l'étudiant et l'ingénieur
Introduction au Model-Based-Design**

Ivan LIEBGOTT

Table des matières

Chapitre 1 : Concepts et stratégies en modélisation

I. Introduction.....	19
II. Industrialisation et cycle de conception d'un système	20
A. Les compétences de l'ingénieur.....	21
B. Le triptyque des performances (*)	22
III. Mise en œuvre de la démarche- Introduction au « Model Based Design ».....	23
A. Architecture matérielle du projet	23
B. La phase d'Expression et de Spécification du Besoin.....	24
C. La phase de Conception-Modélisation-Simulation	25
1. La modélisation « white box »	26
2. La modélisation « Multi-Physique ».....	28
3. La simulation des modèles.....	29
4. La comparaison des performances simulées et mesurées	30
5. La modélisation « gray box ».....	30
6. Le Model-in-the-loop (MIL).....	34
D. La phase de Codage Implémentation	34
1. Le Software-in-the-loop (SIL).....	35
2. Le Processor-in-the-loop (PIL)	36
E. La phase d'Intégration Vérification	37
1. Le Hardware-In-the-Loop (HIL)	37
F. La phase de Validation Recette.....	40

Chapitre 2: Introduction et présentation des outils de modélisation

I. Le logiciel MATLAB-Simulink	41
A. Description et hiérarchie des outils utilisés.....	41
1. MATLAB.....	41
2. Simulink.....	43
3. Simscape.....	44
4. Stateflow	47
5. Utilisation des outils de modélisation.....	47
II. Présentation de l'environnement MATLAB – Simulink.....	49
A. Lancement du logiciel	49
B. La fenêtre de l'environnement MATLAB	49
1. La barre de commande MATLAB	50
C. La fenêtre de l'environnement Simulink.....	50
D. Configuration de MATLAB – Simulink	55
1. Nommer un fichier dans MATLAB/Simulink.....	55
2. Le « path » de MATLAB.....	55
3. Ajout de dossiers dans le « path » pour toutes les sessions	55
4. Ajout de dossiers dans le « path » pour la session courante.....	56

Chapitre 3: Stratégie de conception d'un modèle multi-physique

I. Stratégie de conception d'un modèle multi-physique	57
A. Lien avec le diagramme Chaîne d'énergie/Chaîne d'information	57
II. Application à un pilote hydraulique de bateau	59
A. Diagramme présentant la chaîne d'énergie et d'information du pilote hydraulique de bateau	60
B. Modèle multi-physique du pilote hydraulique de bateau réalisé avec MATLAB - Simulink	61
C. Chargement et simulation du modèle	62
D. Visualisation des résultats issus du modèle multi-physique.....	62
E. Exploration du modèle.....	68
1. Exploration du modèle de la chaîne d'information : Simulink et Stateflow.....	68
2. Exploration du modèle de la chaîne d'énergie : Simscape Electric Library	69
3. Exploration de la chaîne d'énergie : Simscape Fluids	70
4. Exploration de la chaîne d'énergie : Simscape Multibody	71
5. Exploration de la chaîne d'énergie : Simulink	72
F. Pilotage interactif du modèle	72
III. Exemples de modèles multi-physique et exploitations possibles.....	75
A. Le robot Maxpid.....	75
B. L'axe linéaire Control'X.....	79
C. Comment faire un modèle multi-physique avec MATLAB-Simulink ?.....	84

Chapitre 4: Prise en main de Simscape

I. Introduction à la modélisation acausale avec Simscape.....	85
A. Choix des composants	86
B. Placement et assemblage des composants	86
C. Les différents types de ports et de connexions	88
D. Paramétrage des composants.....	90
E. Lancement de la simulation et analyse des résultats	93
F. Utilisation du Data Inspector.....	96
II. Comparaison avec l'approche causale	103
A. Equation de comportement du système	103
B. Choix des composants	103
C. Placement et assemblage des composants.....	103
D. Paramétrage des composants.....	104
E. Lancement de la simulation et analyse des résultats	105
F. Avantage et inconvénients des approches causale et acausale.....	107
III. Les fondamentaux de la modélisation avec Simscape	108
A. Notions de domaines physiques.....	108
B. Les blocs importants de Simscape	109
C. Variables de type « Across » et « Throught » et positionnement des capteurs.....	110
D. L'orientation des composants	110
1. Utilisation de composants actifs	111
2. Implantation et orientation des capteurs.....	114
3. Utilisation de composants dont la dynamique est orientée	117

4. Utilisation de composants passifs.....	117
5. Choix du solveur	118
6. Les problèmes que peut rencontrer le solveur	119
IV. Exemples de modélisation multi-domaine	120
A. Domaine électromécanique – Axe linéaire	120
1. Choix des composants.....	121
2. Placement et assemblage des composants	123
3. Paramétrage des composants.....	124
4. Simulation du modèle en boucle ouverte	131
5. Utilisation du Simscape Results Explorer	133
6. Création de sous-systèmes.....	137
7. Modélisation de l'asservissement en position de l'axe.....	144
B. Domaines hydraulique– vérin hydraulique simple effet.....	149
1. Choix des composants.....	149
1. Placement et assemblage des composants	152
2. Paramétrage des composants.....	152
3. Simulation.....	159
4. Utilisation des fonctionnalités de routage des signaux.....	161
5. Remplacement de la source de pression par une source de débit	164
C. Domaine électrique –Commande PWM d'un moteur à courant continu	168
1. Utilisation du composant « Controlled PWM Voltage »	169
2. Commande PWM d'un moteur à courant continu	173
3. Utilisation du composant « H-Bridge » (pont en H).....	175
D. Rendre un modèle interactif, utilisation de la bibliothèque « dashboard »	183
1. Exemple de modèle interactif.....	183
2. Utilisation des blocs de la bibliothèque.....	185
E. Le domaine thermique	190
1. Rappels théoriques sur les échanges thermiques	190
2. Le phénomène de conduction.....	190
3. Le phénomène de convection	192
4. Le phénomène de rayonnement.....	193
5. La masse thermique	195
6. La résistance thermique.....	196
7. Présentation de la bibliothèque Thermal de Simscape.....	198
8. Chauffage d'une barre métallique par conduction	200
9. Choix des composants du domaine thermique	200
10. Les ports transmetteurs de puissance (PCP) du domaine thermique	204
11. Paramétrage des composants du modèle	204
12. Mise en place des capteurs du domaine thermique	207
13. Définition des conditions initiales du modèle	213
14. Réglage du temps de simulation	214
15. Lancement de la simulation et exportation des résultats	214
16. Interprétation des résultats de la simulation	215
17. Rappels théoriques sur les échanges thermiques à travers une paroi d'un bâtiment.....	222
18. Modélisation avec Simscape des échanges par conduction et convection dans une paroi monocouche	223

19. Utilisation du bloc Thermal Résistance.....	227
20. Modélisation avec Simscape d'une paroi multicouche d'un bâtiment.....	230
21. Modélisation de l'efficacité d'un double vitrage avec couche d'argon.....	241
22. Modélisation thermique d'une maison	253
V. Application pédagogique.....	257
A. Présentation du hacheur série	257
B. Objectifs pédagogiques	259
C. La construction du modèle.....	261
D. La didactisation du modèle.....	268
1. Création d'un sous-système et ajout d'une image	268
2. L'instrumentation du modèle	268
3. Conclusion sur la didactisation du modèle	272
4. Optimiser la didactisation du modèle en fonction de l'objectif d'apprentissage visé	274
E. Exploitation des résultats issus de la simulation du modèle	278
1. Objectif 1 : Comprendre la circulation du courant dans le circuit en phase active et en phase de roue libre	278
2. Objectif 2 : Visualiser et évaluer l'influence du rapport cyclique sur le courant moteur.....	280
3. Objectif 3 : Visualiser et évaluer l'influence de la fréquence de hachage sur l'ondulation du courant.....	282
4. Objectif 4 : Visualiser et évaluer l'influence de l'inductance de la charge sur l'ondulation du courant	284
F. Conclusion.....	287

Chapitre 5: Prise en main de MATLAB

I. Introduction.....	288
A. Création de variable.....	288
B. Création de vecteur	289
C. Indexation des composantes d'un vecteur	289
D. Tracés de courbes	290
E. Mise en forme élémentaires des courbes	292
F. Annotation des graphiques	294
G. Création de matrices	296
H. Opération éléments par éléments	301
I. Créer un script élémentaire	302
J. Les opérateurs de comparaison de MATLAB	305
K. Les structures de boucles usuelles	306
1. Syntaxe de la boucle if – elseif – else	306
2. Syntaxe de la boucle for	306
3. Syntaxe de la boucle while	306
II. Exemple d'exploitations	307
A. Interpolation d'une série de données	307
B. Le calcul symbolique avec MATLAB	309
1. Résolution d'une équation algébrique.....	309
2. Développer ou factoriser une expression	311
3. Dériver une fonction.....	312
4. Intégrer une fonction.....	312

5. Utiliser la transformée de Laplace.....	313
6. Utiliser la transformée inverse de Laplace	314
7. Décomposition en éléments simples	314
1. Résolution d'une équation différentielle d'ordre 1	315
1. Résolution d'une équation différentielle d'ordre 2	316
C. Manipulation des fonctions de transfert.....	318
1. Création d'une fonction de transfert	318
2. Opérations sur les fonctions de transfert.....	319
3. Tracer les réponses temporelles d'un système.....	322
4. Tracer les réponses fréquentielles d'un système.....	323
5. Evaluer les marges de gain et de phase.....	325
6. Tableau récapitulatif des commandes utiles sur les fonctions de transfert	326

Chapitre 6: Prise en main de Simulink

I. Introduction.....	328
II. Régulation en température d'un four	328
A. Ouverture du modèle.....	329
B. Ouverture du script contenant la définition des variables	330
C. Lancement de la simulation.....	331
D. Tracer un diagramme de Bode avec Simulink	331
1. Tracer un digramme de Bode en boucle ouverte.....	332
2. Tracer un diagramme de Bode en boucle fermée.....	336
E. Tracer d'un diagramme de Black-Nichols	339
F. Ajout et paramétrage d'une saturation.....	340
G. Exportation des variables de la simulation vers le Workspace	342
1. Ecriture d'un script pour tracer une série de courbes	345
H. Linéarisation d'un modèle Simulink à partir d'un script	346

Chapitre 7: Prise en main de Stateflow

I. Introduction à Stateflow	349
A. Modélisation d'une machine à état avec Stateflow	349
B. Construction du diagramme d'état.....	349
1. Ouverture du modèle.....	349
2. Insertion d'un « chart »	350
C. Création d'un diagramme d'état élémentaire	351
1. Création des états.....	352
2. Création d'une transition par défaut	353
3. Création des transitions	353
4. Création des actions dans les états.....	354
5. Création des étiquettes de transitions	354
6. Définitions des variables d'entrée et de sortie du diagramme d'état	355
7. Simulation du diagramme d'états	360
D. Architecture des machines à états.....	360

1. La hiérarchie des états	360
2. Les priorités de test des transitions.....	361
3. Etats parallèles.....	361
E. Ajout de niveaux hiérarchique et d'états parallèles dans un diagramme d'état	362
F. Récapitulatif et complément des commandes utiles de Stateflow	368

Chapitre 8: Prise en main de Multibody

I. Introduction à Multibody	370
A. Analyse d'un modèle Multibody	370
B. Paramétrage de la gravité.....	372
II. Intégration d'un modèle Multibody dans un modèle multi-physique	373
A. Connexions du modèle.....	375
B. Interfaçage entre Simscape et Multibody	376
1. Interfaçage entre Simscape et Multibody pour la translation.....	376
2. Interfaçage entre Simscape et Multibody pour la rotation.....	377
3. Ajout de ports sur une liaison.....	378
4. Modélisation d'un effort extérieur variable	382
C. Résultat de la simulation.....	385
III. Importation d'un modèle SolidWorks dans Multibody.....	385
A. Les principes	385
B. Installation de « Multibody Link »	385
C. Conversion d'un fichier assemblage de Solidworks en fichier xml.....	387

Chapitre 9: L'identification d'un modèle

I. La modélisation black-box, l'identification.....	391
A. Présentation de la méthode.....	391
B. Mise en œuvre de la méthode en utilisant la toolbox Identification	392
1. Analyse des données utilisées pour l'identification	392
2. Ouverture et présentation de la toolbox « System Identification »	393
3. Importation des données.....	394
C. Utilisation de la méthode en utilisant les lignes de commandes.....	400

Chapitre 10: Le contrôle commande avec MATLAB/Simulink

I. Introduction.....	402
II. Réglage automatique d'un PID	402
A. Modélisation	402
B. Ouverture du modèle.....	403
1. Analyse de la réponse temporelle.....	406
2. Importation dans Simulink.....	410
III. Réglage manuel d'un PID avec l'outil « Control System Designer »	412
A. Ouverture du modèle.....	412
B. Réglage du PID.....	413

1. Lancement de Control System Designer	413
2. Diagrammes de Bode et de Black de la FTBO	416
3. Visualisation des réponses temporelles.....	418
4. Réglage du PID.....	423
5. Définition et visualisation des critères de performance.....	423
6. Réglage du PID à l'aide des curseurs	426
7. Exportation du réglage dans le modèle Simulink	428
IV. Conception et réglage d'un correcteur de forme quelconque	430
A. Ouverture du modèle.....	430
B. Conception du correcteur	431
1. Diagrammes fréquentiels de la FTBO.....	433
2. Visualisation des réponses temporelle et fréquentielle de la FTBF	436
3. Synthèse du correcteur.....	441
4. Visualisation de l'influence du gain de la FTBO.....	442
5. Ajout d'un intégrateur.....	444
6. Ajout d'un correcteur à avance de phase (Lead).....	446
7. Ajout d'un filtre rejecteur (Notch).....	449
8. Réglage d'un filtre rejecteur	450
9. Exportation de la fonction de transfert du correcteur vers le modèle Simulink	455

Chapitre 11: Ingénierie numérique en langage MATLAB

I. Introduction.....	457
II. Les fonctions.....	457
A. Organisation et structure.....	457
B. Portée des variables.....	459
C. Création et définition d'une fonction.....	459
D. Utilisation d'une fonction par un script	463
E. Création d'une fonction prenant en argument une autre fonction	468
F. Appel d'une fonction à plusieurs arguments.....	473
III. Dérivation numérique.....	475
A. Aspects théoriques	475
1. Différence finie progressive	476
2. Différence finie rétrograde	476
3. Différence finie centrée	477
B. Codage en langage MATLAB	478
1. Différence finie progressive	478
2. Différence finie rétrograde	479
3. Différence finie centrée	481
4. Calcul exact de la fonction dérivée en utilisant le calcul symbolique	482
C. Exploitation de l'algorithme	483
1. Comparaison des différentes méthodes de dérivation numérique	483
2. Quantification et visualisation de l'erreur induite par les différentes méthodes.....	487
3. Détermination de la loi d'évolution de l'erreur en fonction du pas.....	489
4. Robustesse du processus.....	494

D. Application de la dérivation numérique	497
1. Dérivation du signal issu d'un codeur incrémental	497
2. Mise en évidence des problèmes rencontrés.....	498
3. Dérivation numérique à pas variable.....	499
IV. Résolution de l'équation $f(x)=0$	503
A. Dichotomie.....	503
B. Méthode de Newton	507
V. Intégration numérique	511
A. Principe.....	511
B. Codage en langage MATLAB.....	513
1. Méthode des rectangles.....	513
2. Méthode des trapèzes	514
3. Calcul exact de l'intégrale d'une fonction en utilisant le calcul symbolique.....	515
C. Exploitation de l'algorithme	517
1. Comparaison des différentes méthodes d'intégration numérique	517
2. Quantification et visualisation de l'erreur induite par les différentes méthodes.....	520
3. Détermination de la loi d'évolution de l'erreur en fonction du pas.....	522
4. Robustesse du processus.....	526
D. Applications	529
1. Analyse d'une loi en trapèze de vitesse.....	529
VI. Filtrage d'un signal	533
A. Intérêt du filtrage d'un signal	533
B. Le filtre à moyenne glissante	535
C. Le filtre passe-bas du premier ordre	538
D. Utilisation de Simulink pour le filtrage d'un signal.....	542
1. Utilisation d'une fonction de transfert de la bibliothèque Continuous de Simulink	543
2. Utilisation du bloc Analog Filter Design de la DSP Toolbox de Simulink	545
3. Utilisation du bloc Moving Average de la DSP System Toolbox	547
VII. Résolution numérique des équations différentielles.....	550
A. Résolution d'une équation différentielle d'ordre 1 – Méthode d'Euler.....	550
1. Aspects théoriques	550
2. Codage en langage MATLAB.....	552
B. Résolution d'une équation différentielle d'ordre n – Méthode d'Euler.....	555
1. Aspects théoriques	555
2. Codage en langage MATLAB.....	558
3. Robustesse du processus. Résolution d'une équation différentielle d'ordre 3	565
4. Robustesse du processus. Application à la résolution de l'équation de Van der Pol	567
C. Applications	570
1. Modélisation d'un oscillateur mécanique	570
2. Les différentes méthodes de résolution d'une équation différentielle avec MATLAB	570
3. Résolution avec my_ode, ode 23 et en calcul symbolique.....	571
4. Résolution de l'équation différentielle avec Simulink	573
5. Résolution de l'équation différentielle avec Simscape	575
VIII. Résolution de systèmes linéaires – Méthode de Gauss	577
A. Présentation de la méthode.....	577

B. Exemple	578
1. Recherche du pivot.....	578
2. Echange de lignes.....	578
3. Transvection - Triangularisation	579
4. Résolution.....	579
C. Codage en langage MATLAB	580
1. Recherche du pivot.....	580
2. Echange de lignes.....	583
3. Transvection.....	584
4. Triangularisation.....	586
5. Résolution.....	587
6. Pivot de Gauss.....	589
D. Robustesse	591

Chapitre 12: MATLAB/Simulink Online

I. Présentation	593
II. Installation de l'environnement de travail à distance.....	594
III. Utilisation de MATLAB/Simulink online	595
A. Création d'un script élémentaire avec MATLAB online	596
B. Création d'un modèle élémentaire avec Simulink online.....	597
C. Exploitation d'un modèle multi-physique avec MATLAB/Simulink on line	599
D. Partager des dossiers contenant ses modèles avec d'autres utilisateurs	602
1. Partage d'un dossier à l'aide de la version de MATLAB installée sur son ordinateur	602
2. Partage d'un dossier en utilisant MATLAB online	604
3. Partage d'un dossier en utilisant MATLAB Drive online	605

Chapitre 13: MATLAB Mobile

I. Présentation	607
II. Les fonctionnalités de MATLAB Mobile.....	608
1. Taper des lignes de commandes	608
2. Configurer le clavier MATLAB Keyboard.....	609
3. Accès à MATLAB Drive.....	609
4. Exécuter un script depuis MATLAB Mobile	609
5. Acquisition des données des capteurs du smartphone avec MATLAB Mobile	610
6. Exploitation des données des capteurs du smartphone avec MATLAB Mobile	612

ANNEXES

Paramétrage des scopes.....	615
Utilisation du signal builder.....	622
Table des figures	628

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 maîtrise 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. L'outil de modélisation privilégié est le schéma bloc qui décrit les équations mises en place dans la phase de modélisation.

- *La modélisation « multi-phérique » :*
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 « gray box » :*
Cette approche vient compléter le modèle « white box » ou « multi-phérique » 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.

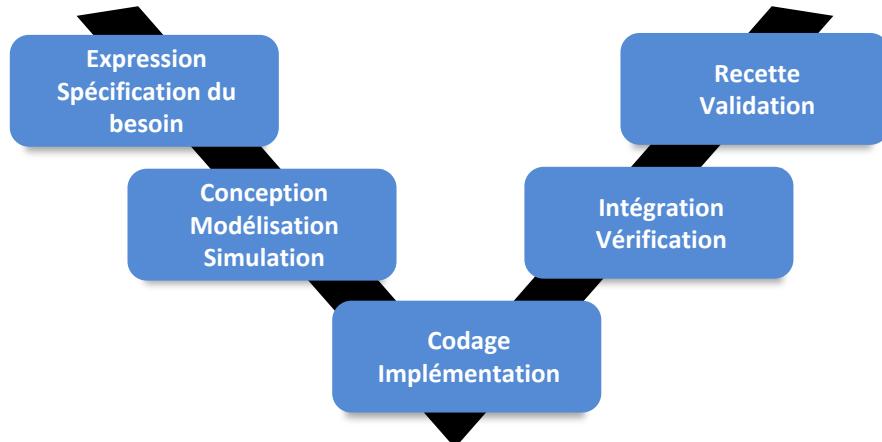


Figure 1 : cycle en V simplifié

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.

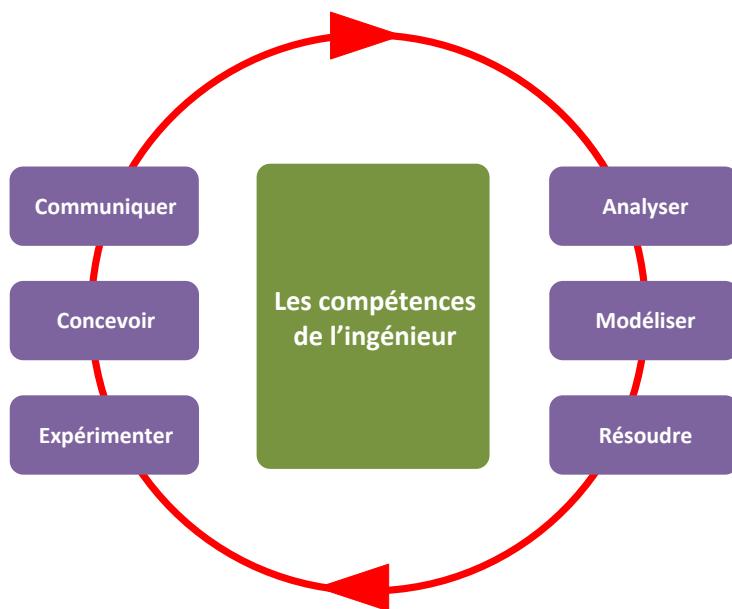


Figure 2 : les compétences de l'ingénieur

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

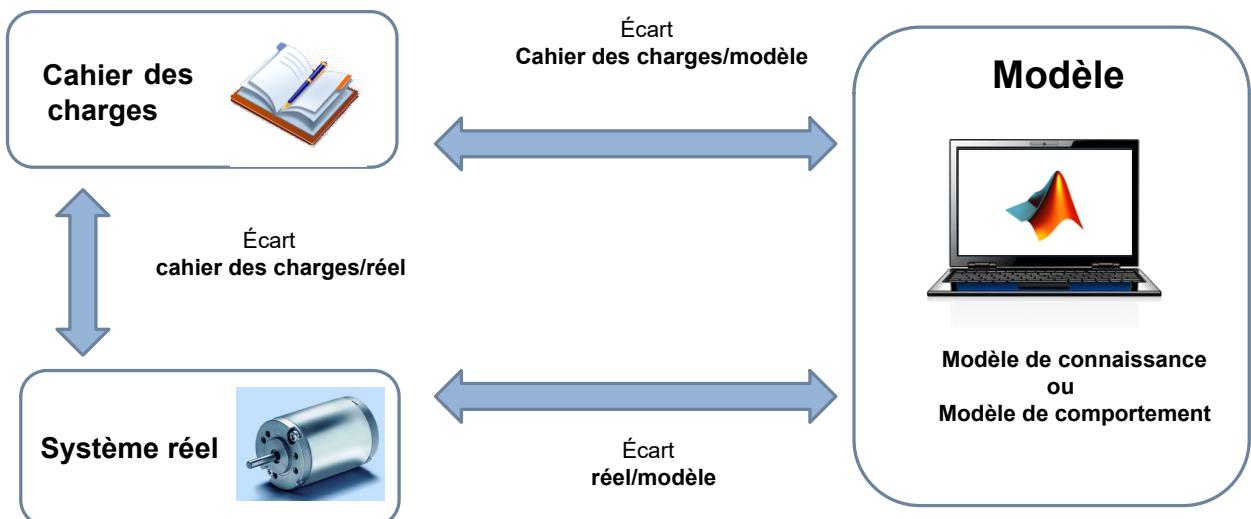


Figure 3 : triptyque Cahier des charges/Système réel/Modèle

(*) 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 modifiable 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.

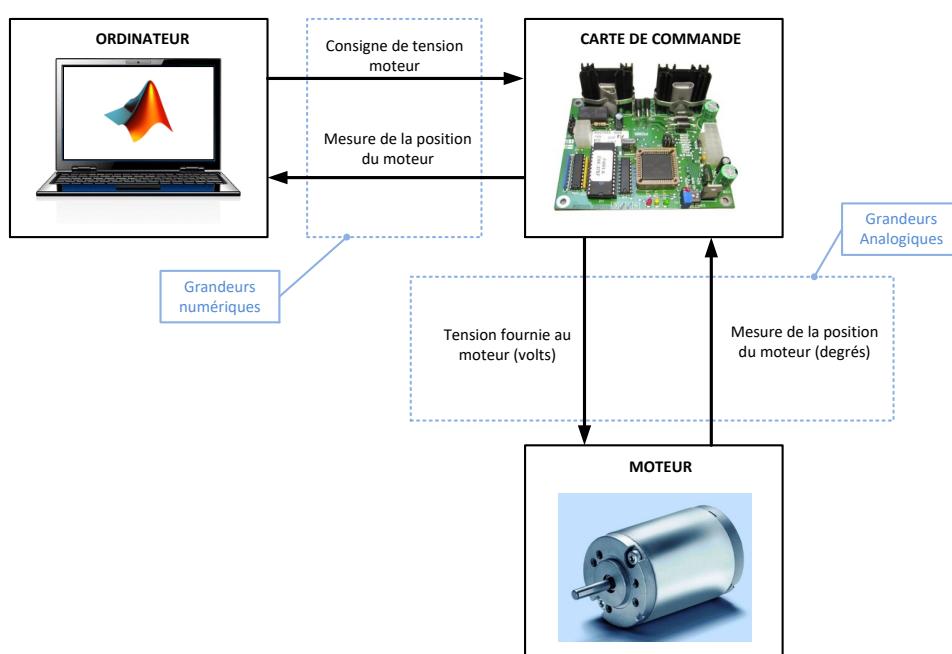


Figure 4 : architecture matérielle nécessaire à la mise en place de la démarche

B. La phase d'Expression et de Spécification du Besoin

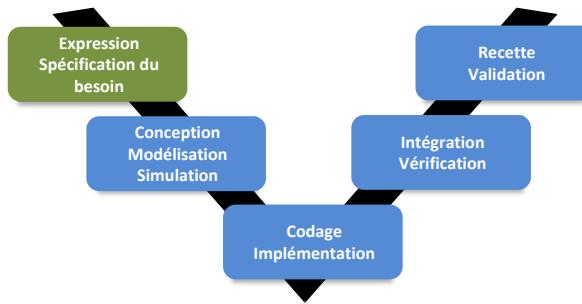


Figure 5 : la phase d'expression et de spécification du besoin

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.

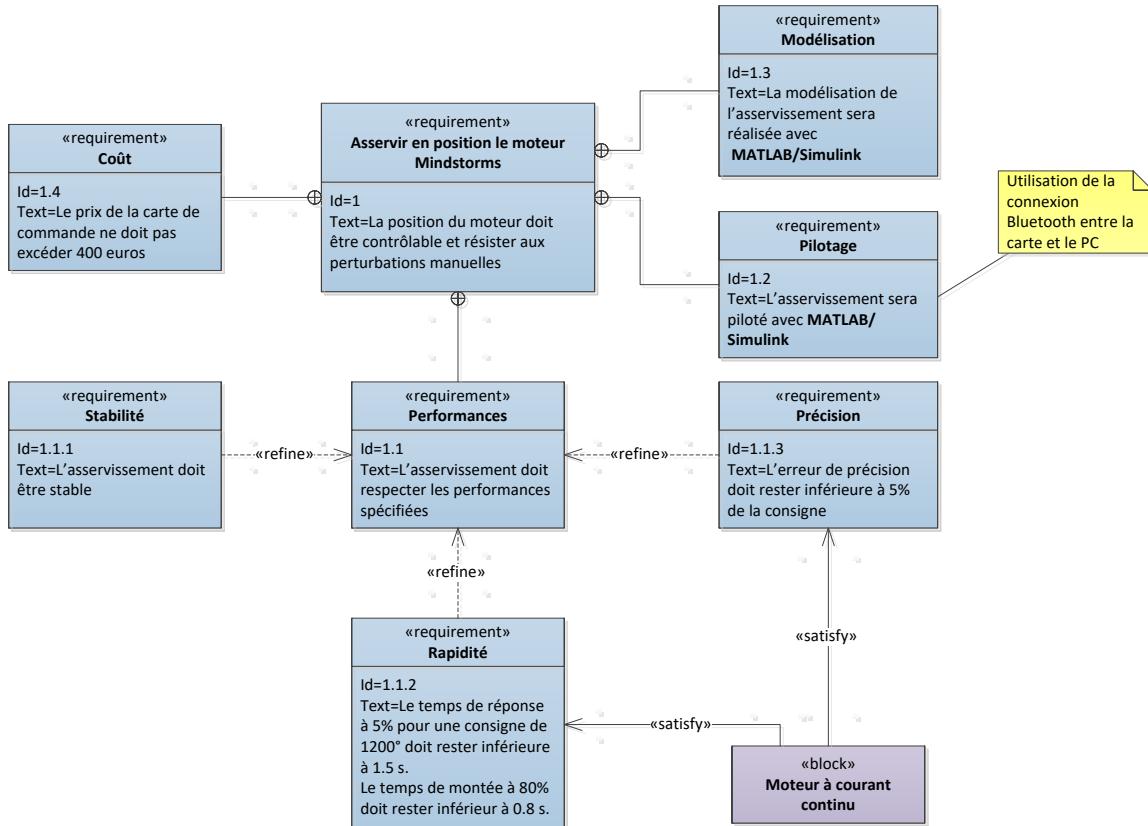


Figure 6 : diagramme des exigences de l'asservissement de position

C. La phase de Conception-Modélisation-Simulation

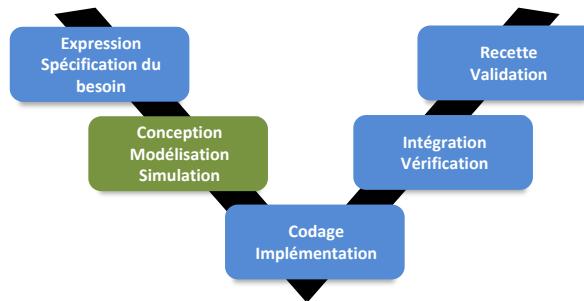


Figure 7 : la phase de Conception, Modélisation, Simulation

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 importants 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 maîtrise 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 mises en œuvre pour optimiser cette phase.

- La modélisation « white box » :
- La modélisation « multi-physique » :
- La modélisation « gray 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 maîtrise 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 permettra 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.

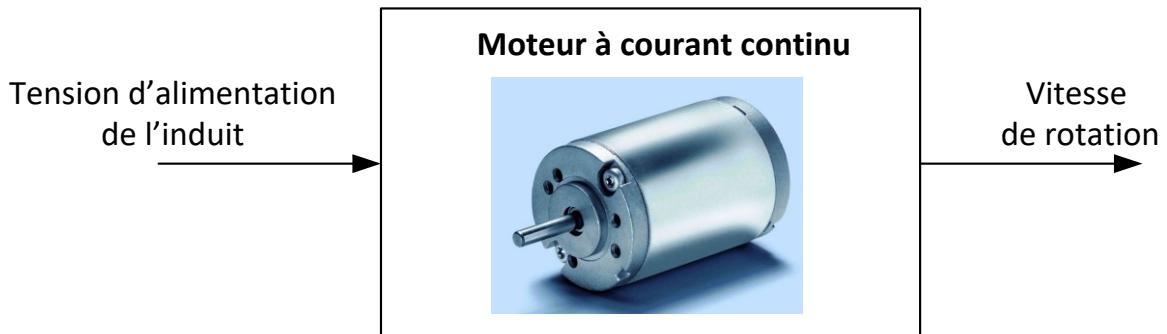


Figure 8 : relation entrée/sortie pour le moteur à courant continu

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 :

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

$u(t)$: tension de commande du moteur (V)
 $e(t)$: force contre électromotrice du moteur (V)
 $i(t)$: intensité dans le moteur (A)
 $C_m(t)$: couple exercé par le moteur (N.m)
 $C_r(t)$: couple résistant ramené sur l'arbre moteur (N.m)
 $\omega_m(t)$: vitesse de rotation du moteur (rad/s)
 R : résistance de l'induit (Ω)
 L : inductance de l'induit (H)
 J : inertie équivalente ramenée sur l'arbre moteur ($\text{kg} \cdot \text{m}^2$)
 f : paramètre de frottement visqueux (N.m.s)
 K_t : constante de couple (N.m/A)
 K_e : coefficient de force contre électromotrice (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 K_e et K_t . 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.

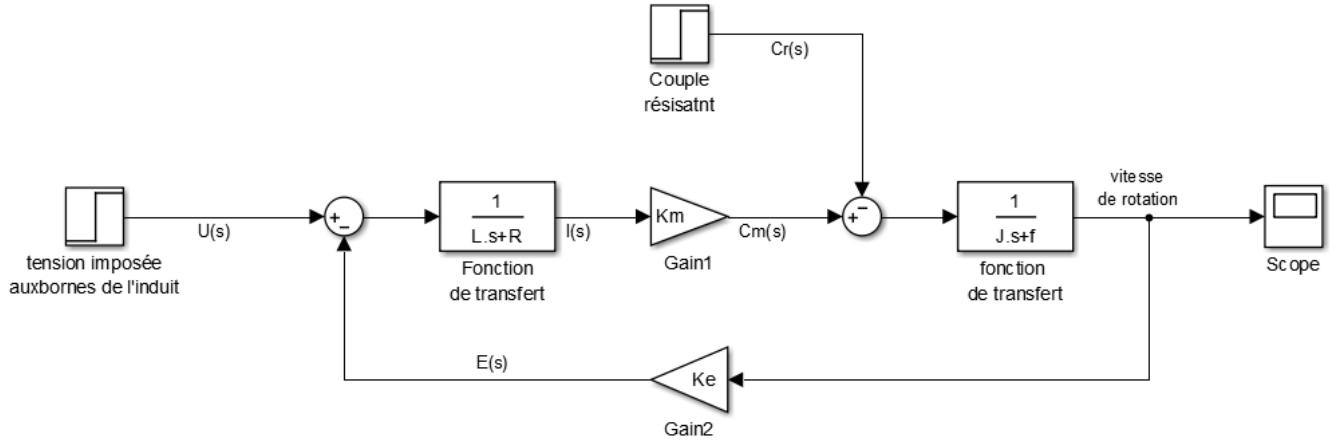


Figure 10 : modélisation sous forme de schéma bloc du moteur à courant continu

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

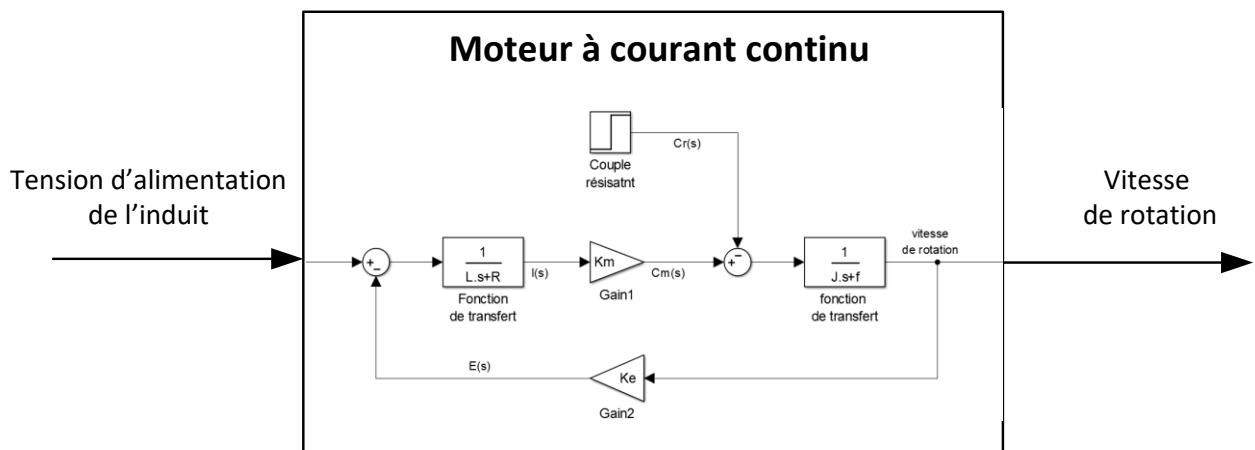


Figure 11 : modélisation « white box » du moteur à courant continu

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.

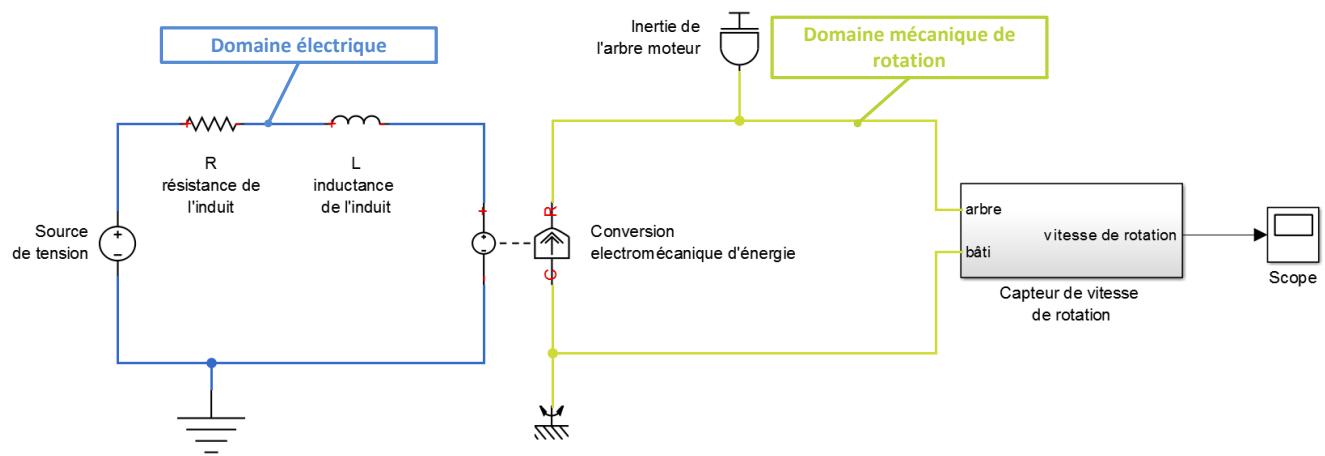


Figure 12 : modélisation par assemblage de composants du moteur à courant continu

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

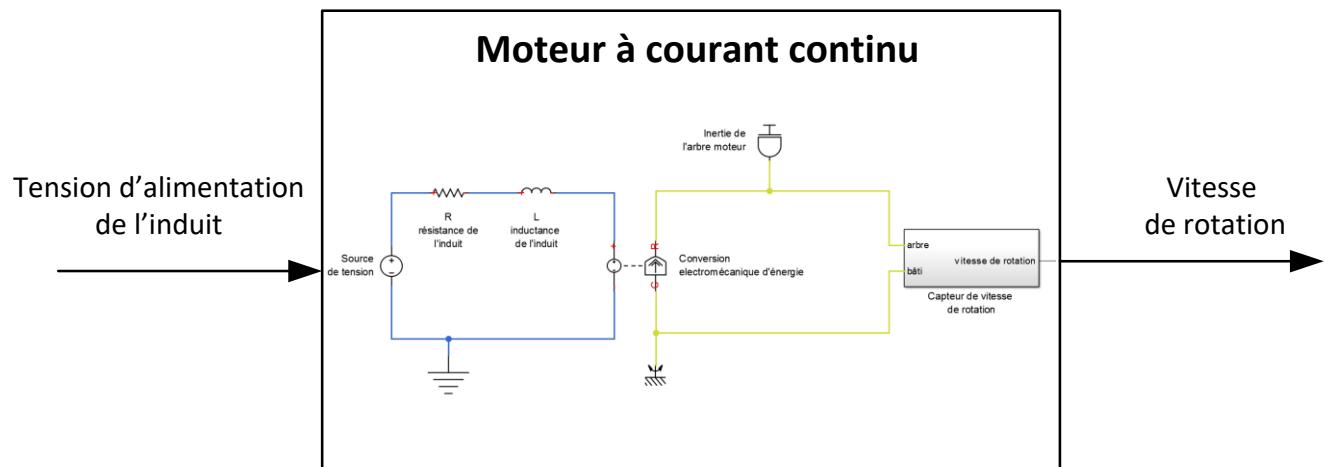


Figure 13 : modélisation multi-physique acausal du moteur à courant continu

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-physics », 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.

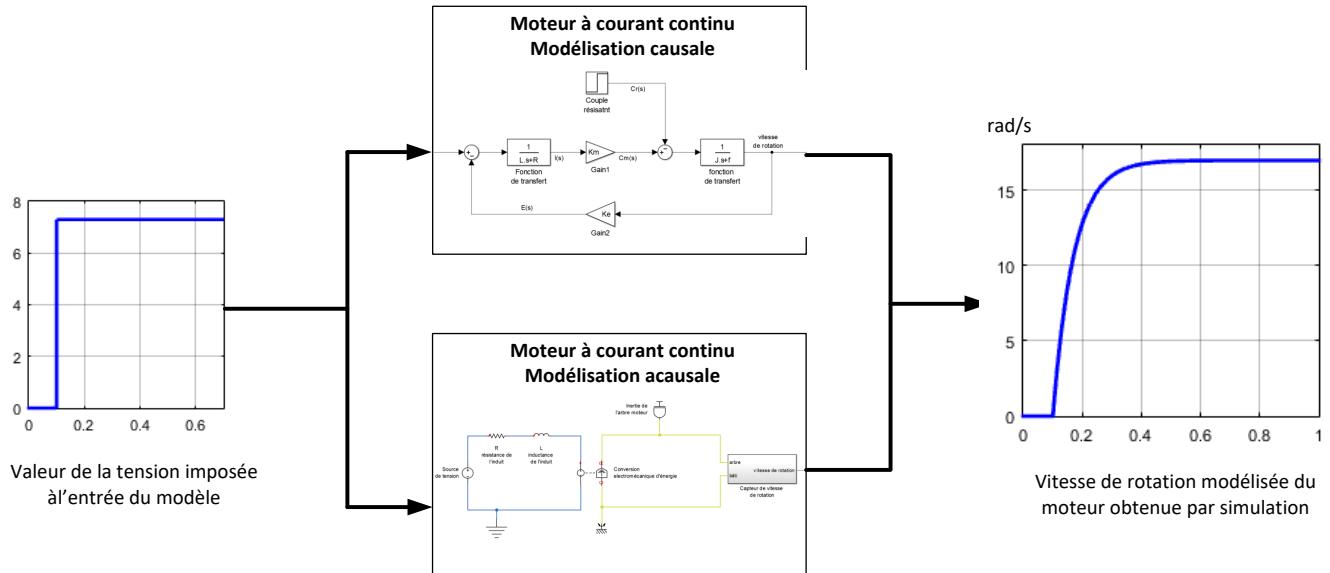


Figure 14 : résolution des modèles et visualisation des résultats

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.

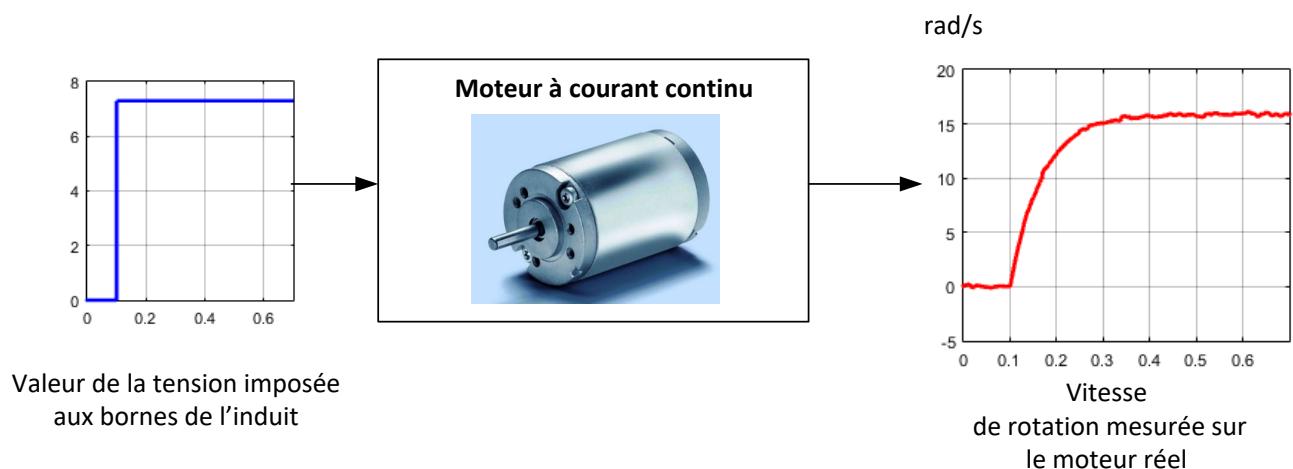


Figure 15 : mesure de l'évolution de la vitesse de rotation du moteur en fonction du temps

4. La comparaison des performances simulées et mesurées

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

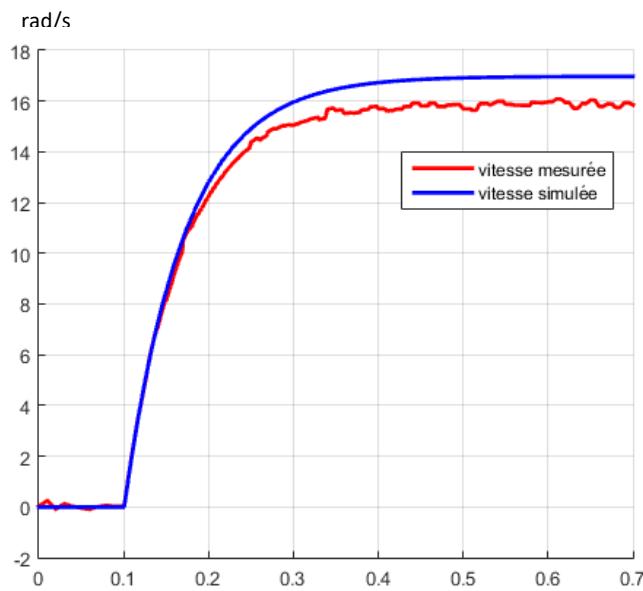


Figure 16 : évaluation des écarts entre les performances simulée et les performances mesurées

Dans le cas général et à ce stade, la mise en évidence d'écart 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 « gray box » va permettre de compléter le processus de modélisation en proposant une méthode de détermination des paramètres inconnus.

5. La modélisation « gray box »

Le principe de cette modélisation est d'estimer les paramètres inconnus du modèle par comparaison entre 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).

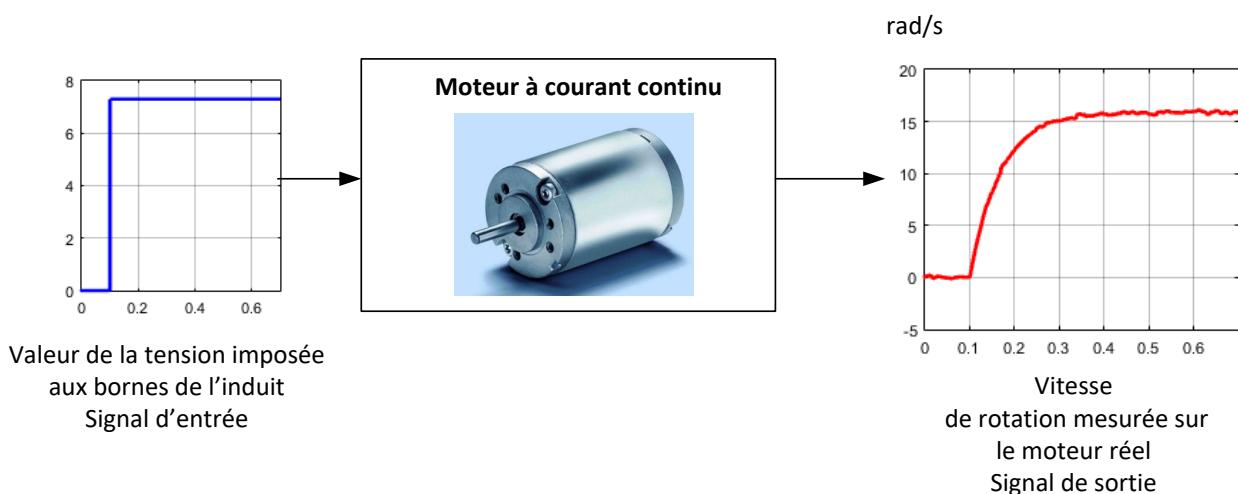


Figure 17 : essai réalisé en vue de l'estimation des paramètres inconnus

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.

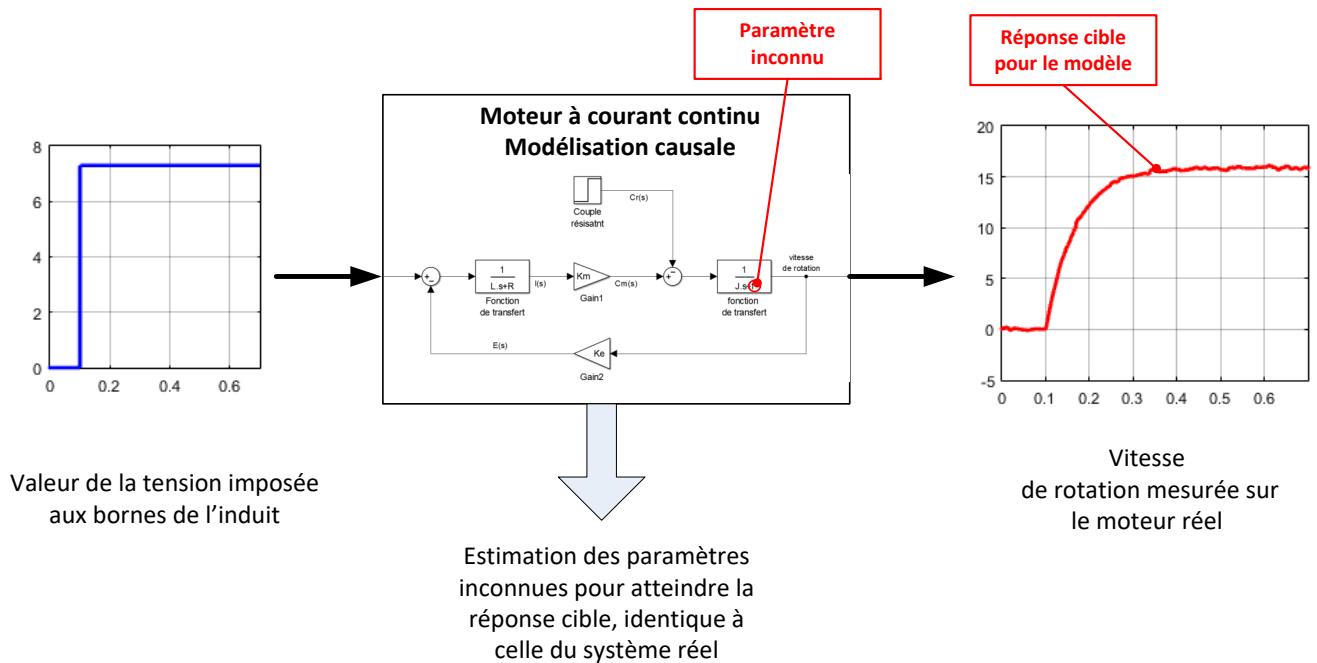


Figure 18 : illustration du principe de l'estimation de paramètre

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.

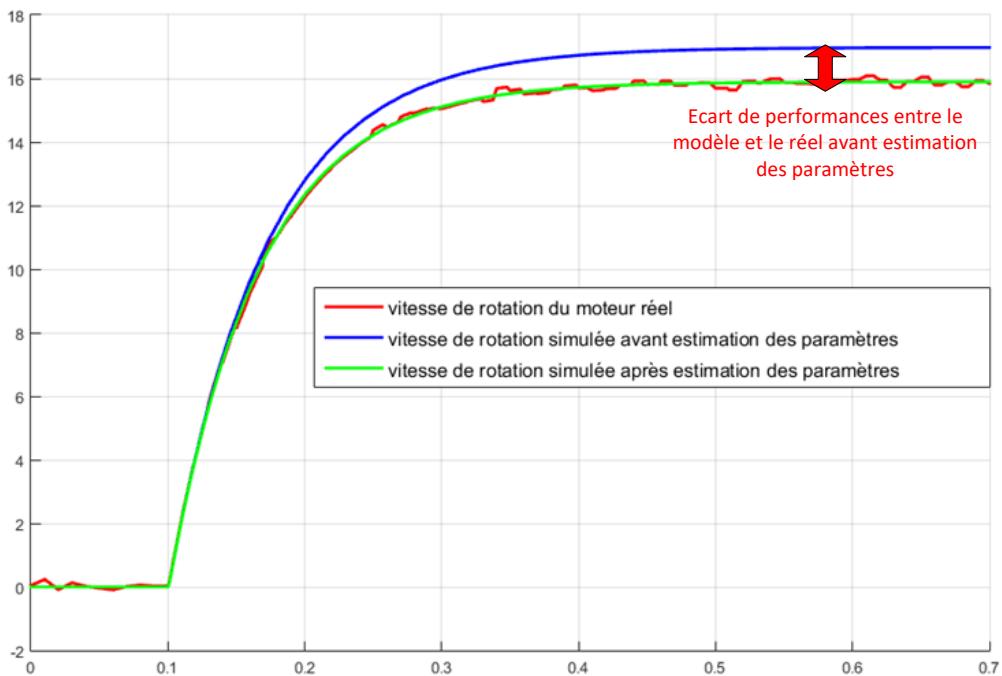


Figure 19 : les réponses du modèle avant et après estimation des paramètres inconnus

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.

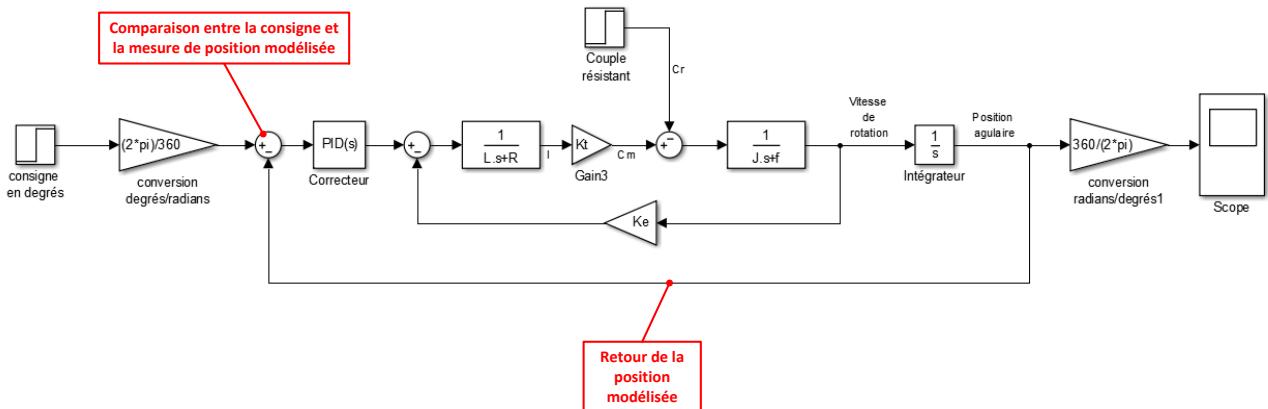


Figure 20 : modélisation de l'asservissement de position du moteur

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

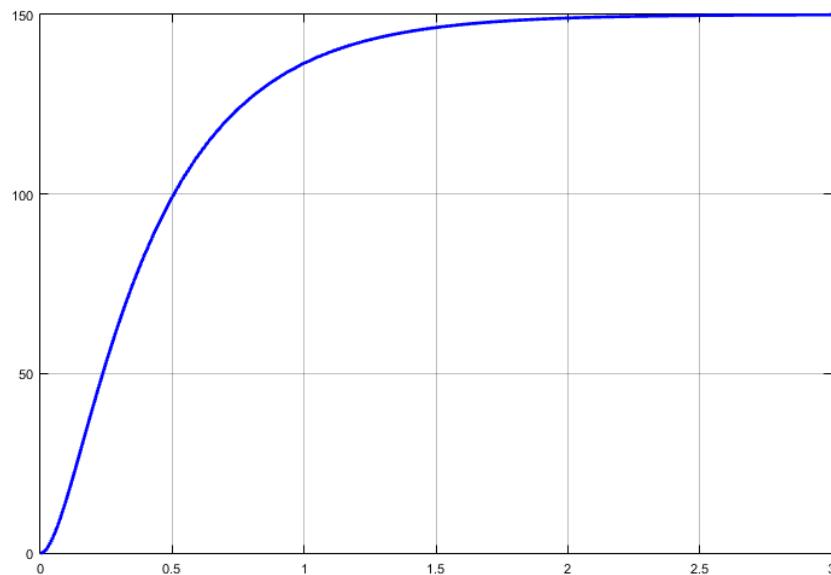


Figure 21 : réponse de l'asservissement de position non corrigé

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.

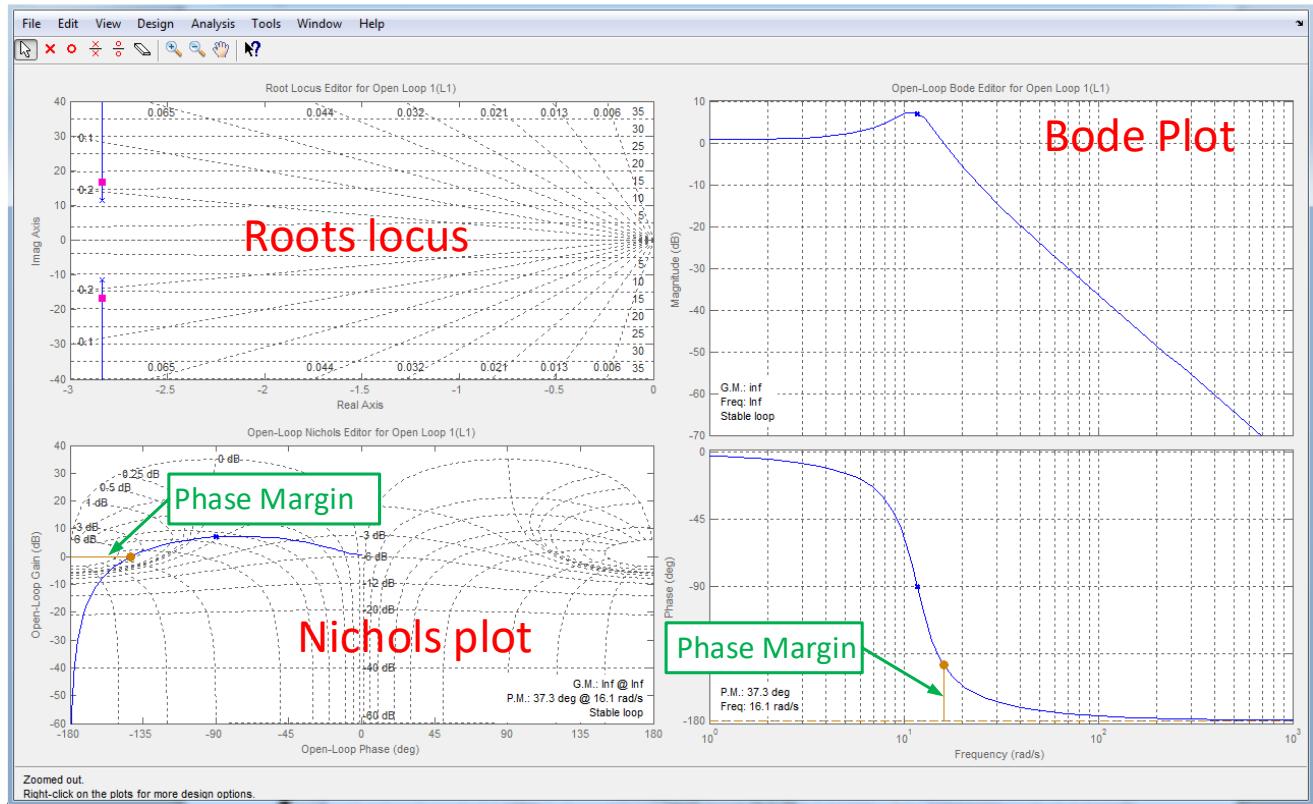


Figure 22 : illustration de l'utilisation d'un outil de contrôle pour régler un correcteur

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.

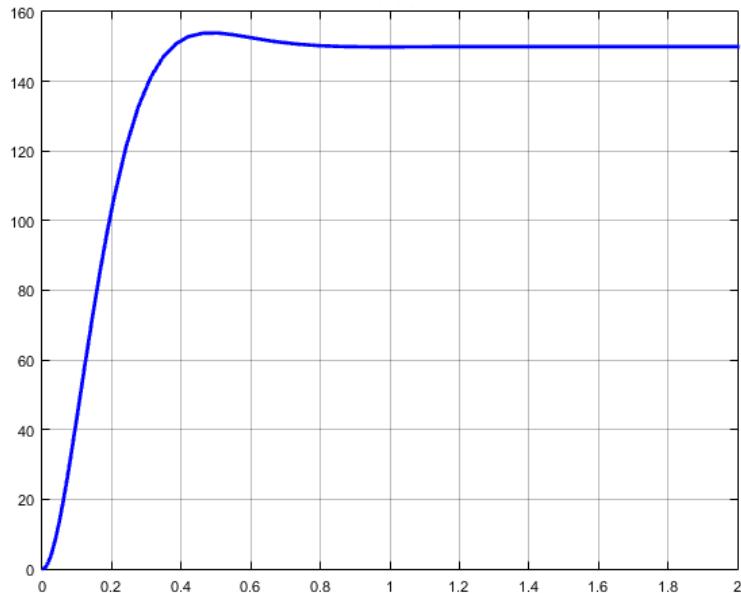


Figure 23 : réponse de l'asservissement de position corrigée

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 Implémention** 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.

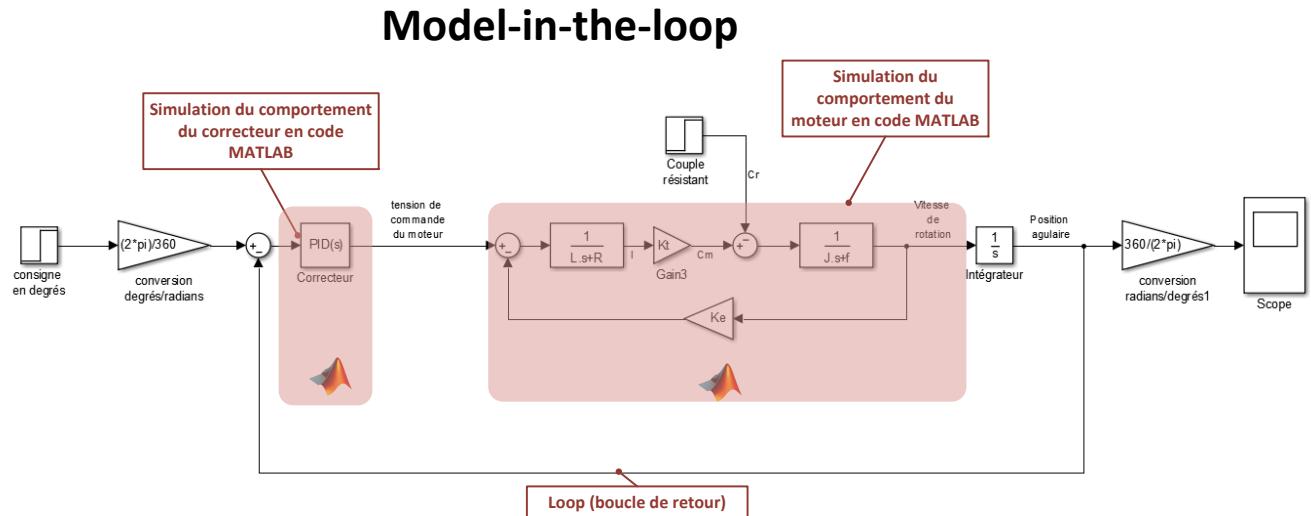


Figure 24 : le Model-in-the-loop (MIL)

D. La phase de Codage Implémentation

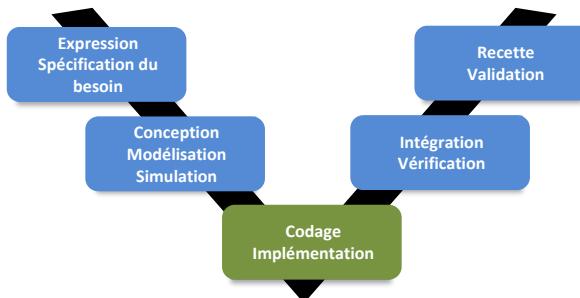


Figure 25 : la phase de Codage Implémentation

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.

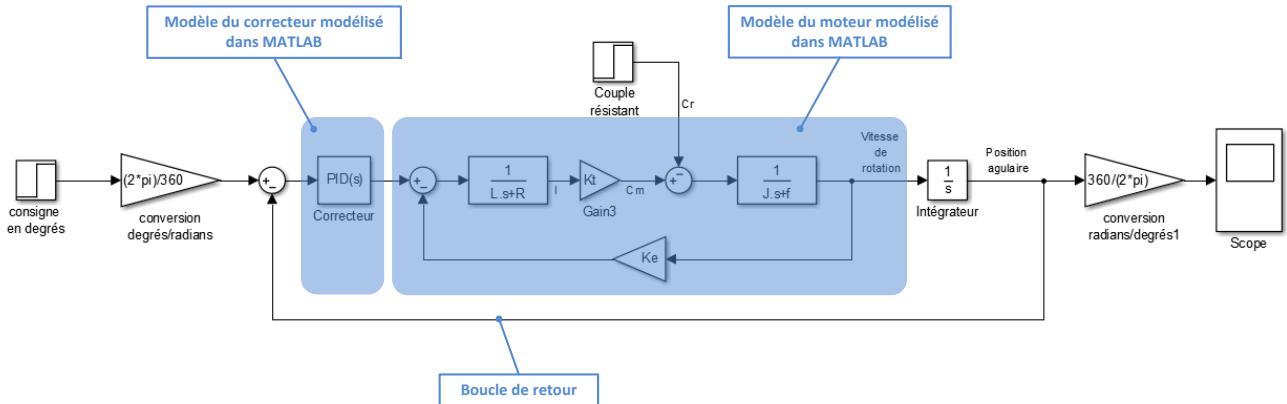


Figure 26 : structure du modèle à l'issu de la phase de Conception Modélisation Simulation

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 termes 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).

Software-in-the-loop

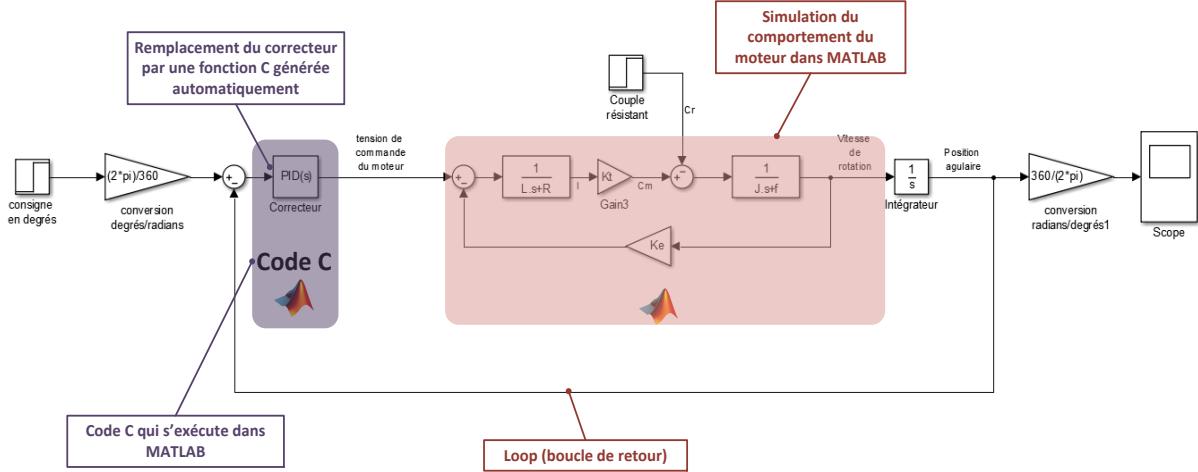


Figure 27 : le Software-in-the-loop(SIL)

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 exclure 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.

Processor-in-the-loop

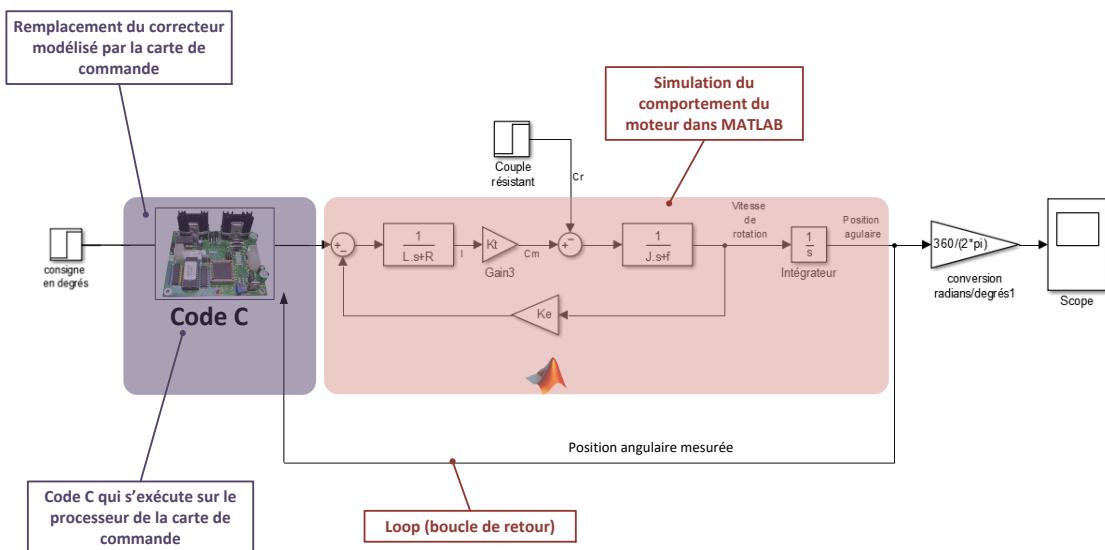


Figure 28 : le Processor-in-the-loop

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.

E. La phase d'Intégration Vérification

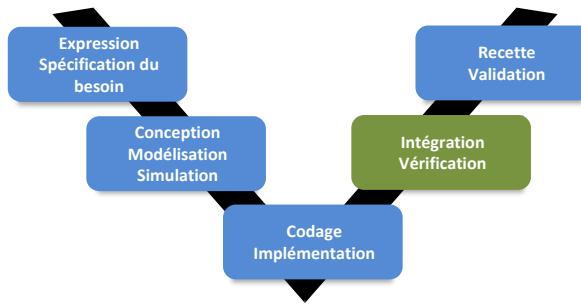


Figure 29 : la phase d'Intégration Vérification

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.

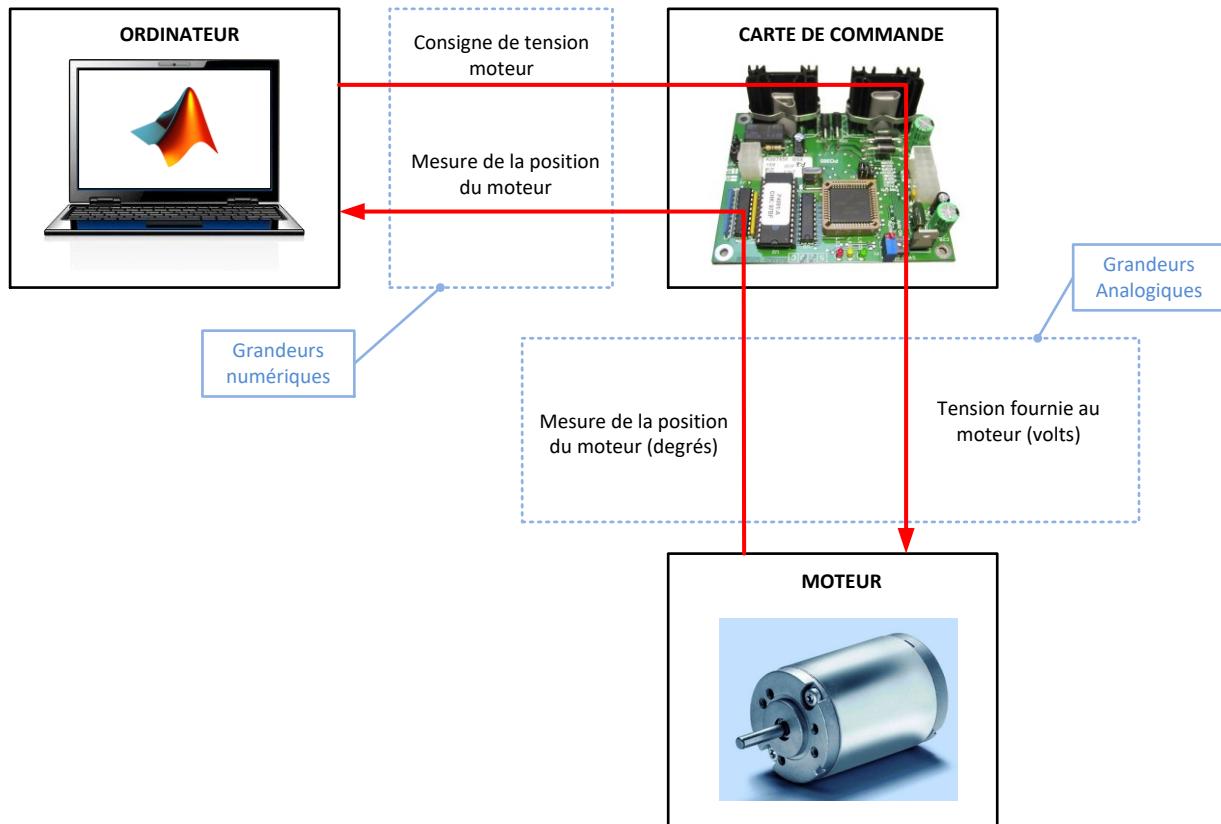


Figure 30 : architecture matérielle du hardware-in-the-loop en mode externe

Hardware-in-the-loop

Mode externe

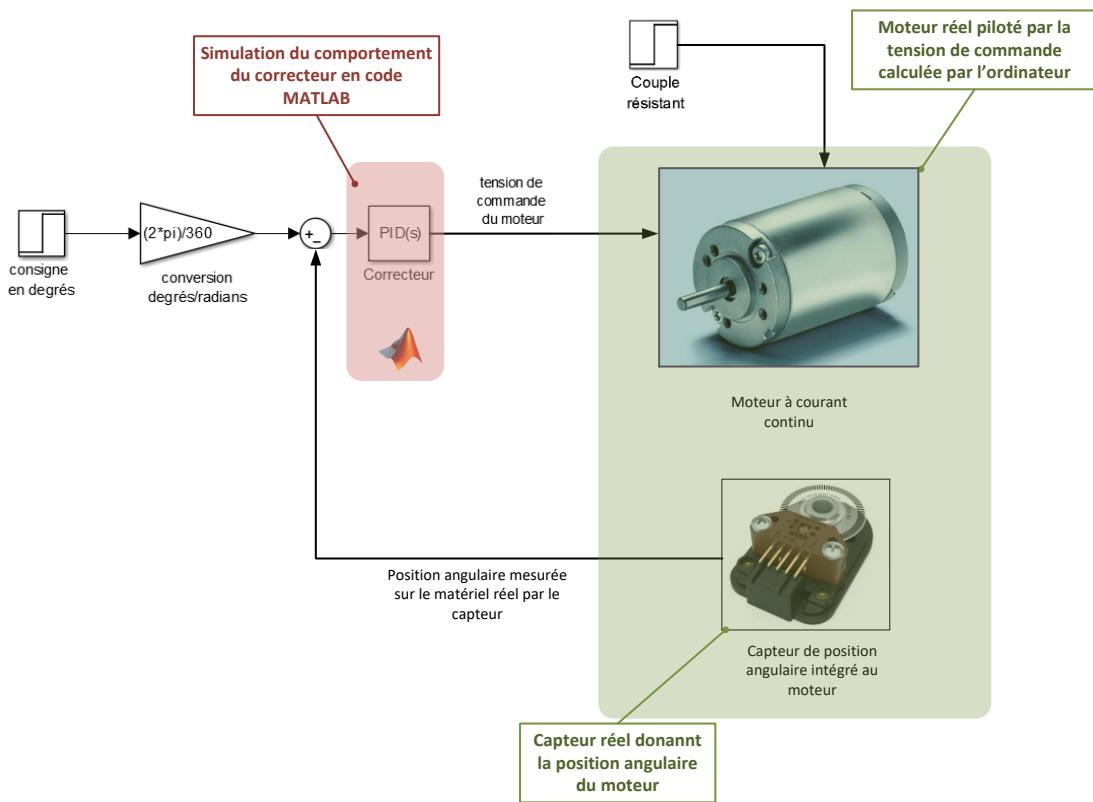


Figure 31 : le hardware-in-the-loop en mode externe: intégration du matériel dans la boucle

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.

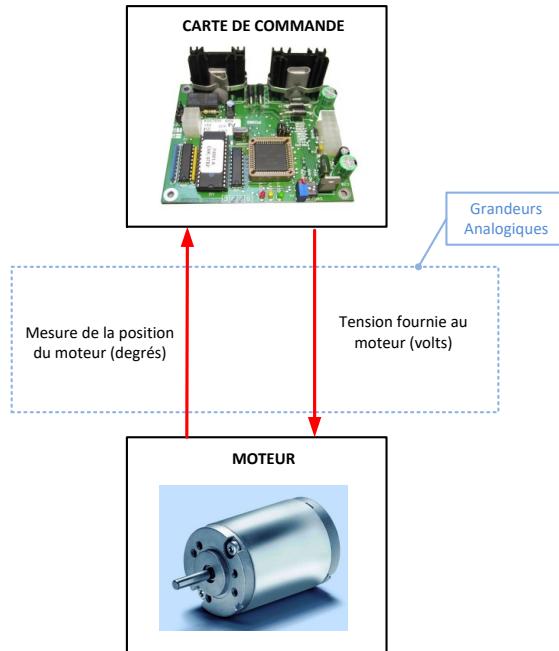


Figure 32 : architecture matérielle du Hardware-In-the-Loop en mode embarqué

Hardware-in-the-loop

Mode embarqué

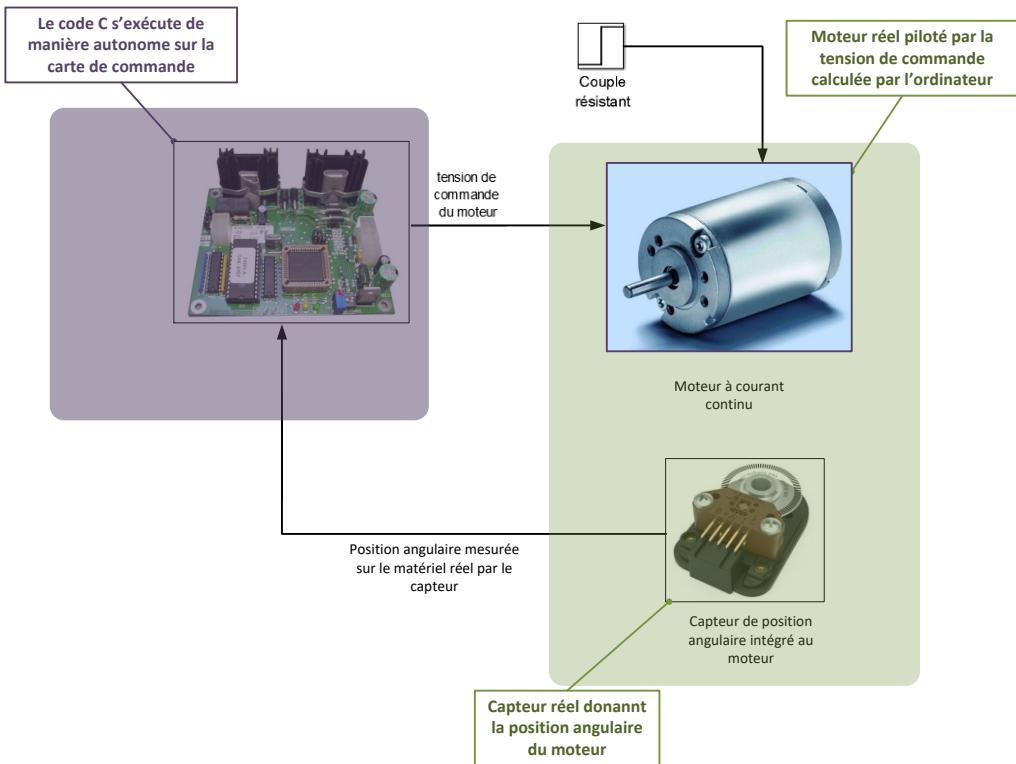


Figure 33 : le Hardware-In-the-Loop mode embarqué: fonctionnement autonome du matériel

L'une ou l'autre de ces configurations permettent de vérifier que le correcteur mis au point permet d'atteindre les performances spécifiées 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**.

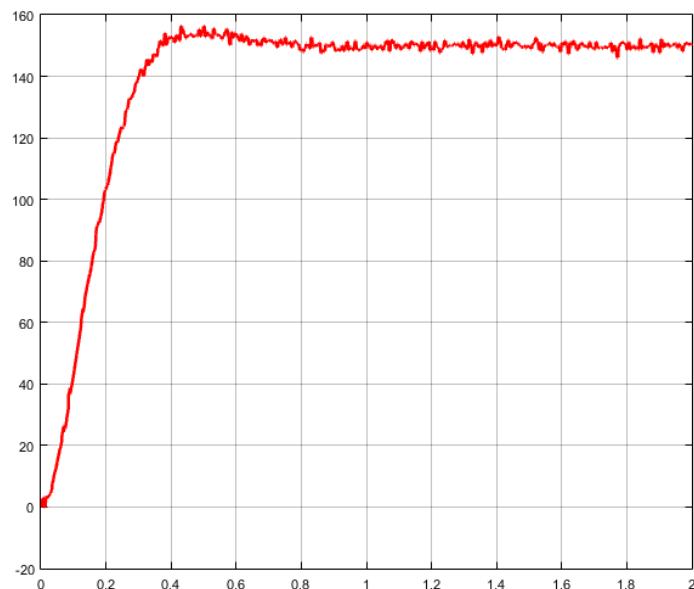


Figure 34 : visualisation de la vitesse de sortie du moteur à l'issu de la phase Hardware-in-the-loop

F. La phase de Validation Recette

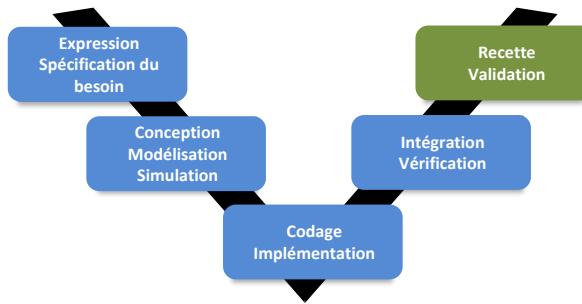


Figure 35 : la phase de Validation Recette

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-phérique. 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 maîtrise 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 systèmes, **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.

Editor - C:\Users\ivan\Documents\MATLAB\My Scripts\Introduction\My_first_script.m

```

1 % Ce script permet de tracer sur une même fenêtre graphique les courbes
2 % f(t) = sin (a.t) avec un paramètre a variant de 1 à 4 et la variable
3 % theta variant de 0 à 2*pi
4
5
6 % création du vecteur theta
7 theta = [0:0.01:2*pi];
8
9 % création d'une nouvelle fenêtre graphique
10 figure;
11
12 % la fonction hold all permet de garder tous les tracés sur la même fenêtre
13 % chaque courbe sera tracée avec une couleur différente
14 hold all;
15
16 % pour a prenant les valeurs entre 1 et 4 avec un pas de 1
17 for a = 1:4
18     % tracé de la fonction f(theta) = sin(a*theta) pour theta
19     % compris entre 0 et 2*pi
20     % LineWidth permet de spécifier une épaisseur de ligne
21     plot(theta,sin(a*theta), 'LineWidth', 2);
22 end
23
24 grid on;
25 % spécification du titre du graphique et des grandeurs physiques des axes
26 title('Fonction Sinus')
27 % MATLAB intègre un interpréteur TeX
28 xlabel('angle \theta en radians');
29 ylabel('f(\theta)=sin(a.\theta)');
30
31 % Spécification des limites des axes
32 axis([-1 7 -1.5 1.5]);

```

Figure 36 : exemple de script écrit en langage MATLAB

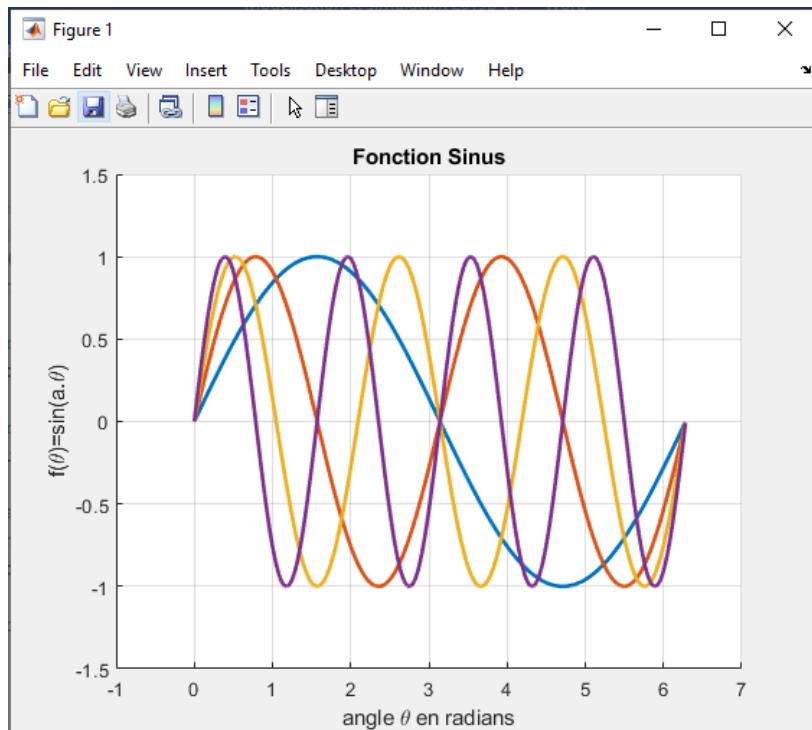


Figure 37 : le résultat obtenu après exécution du script

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 :

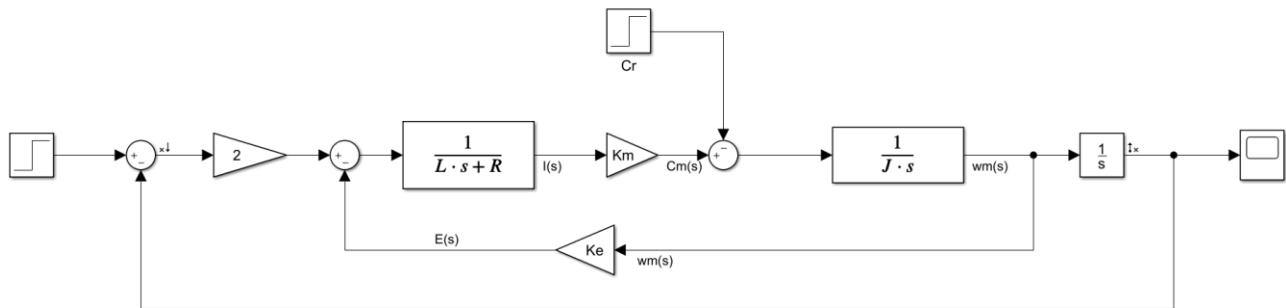


Figure 38 : exemple de modélisation Simulink

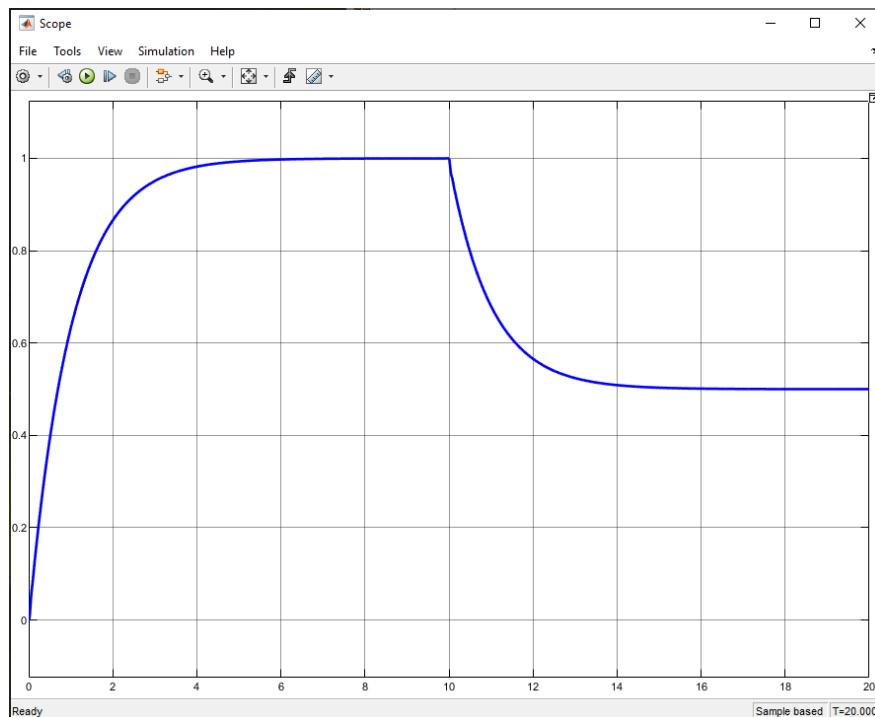


Figure 39 : la visualisation dans un scope des résultats obtenus

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.

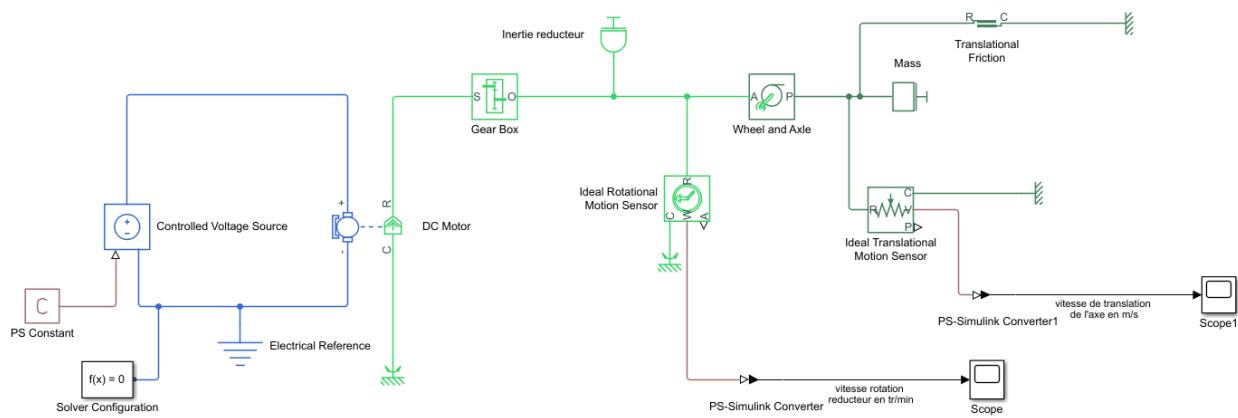


Figure 40 : exemple de modélisation Simscape

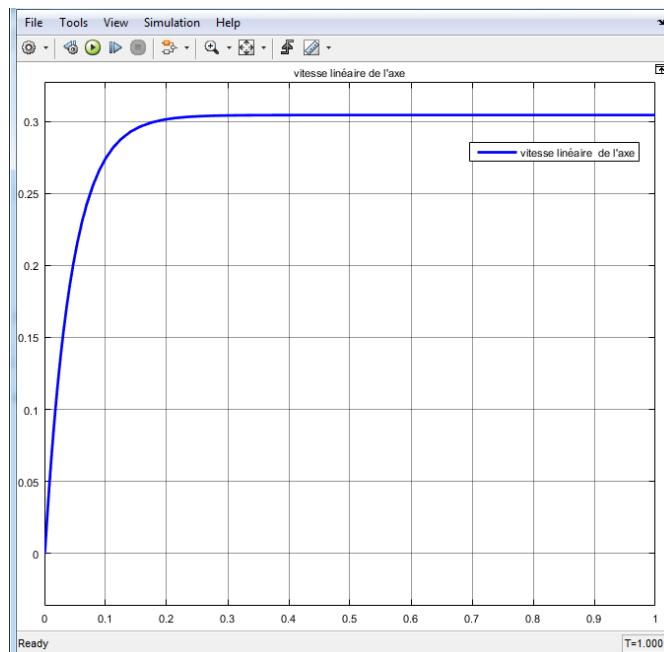


Figure 41 : la visualisation dans un scope des résultats obtenus

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

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

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

Simscape et ses bibliothèques

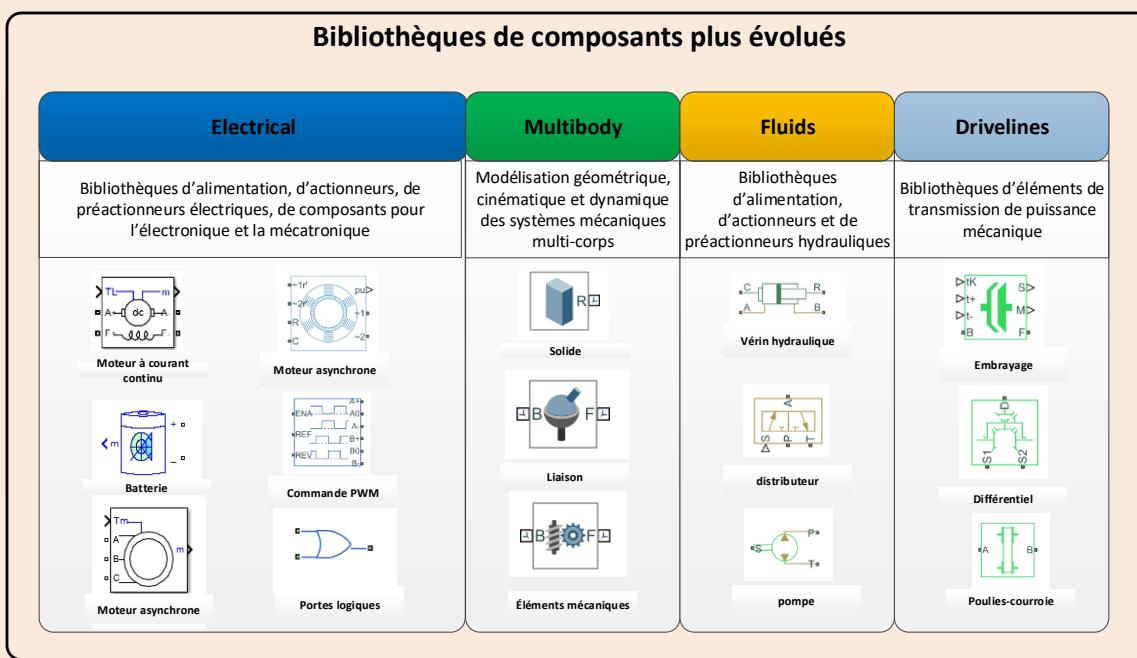
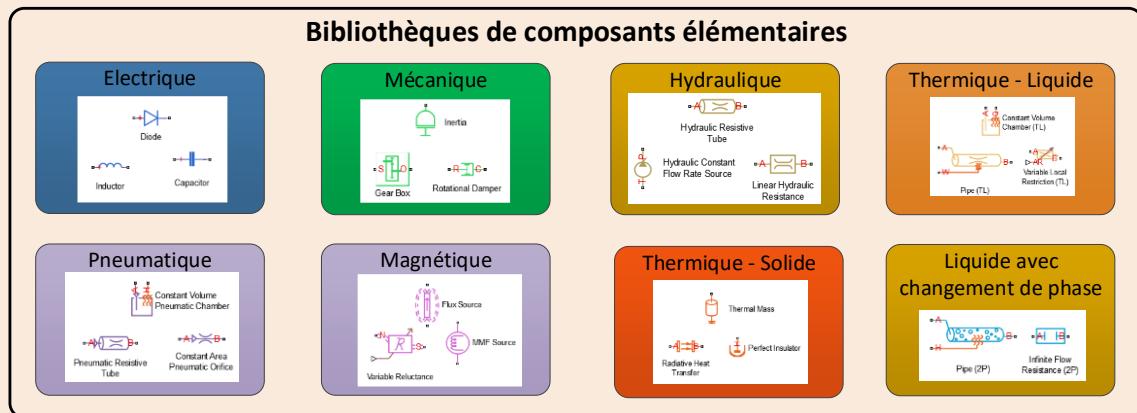


Figure 42 : Simscape et ses bibliothèques

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 modélisée avec **Stateflow**.

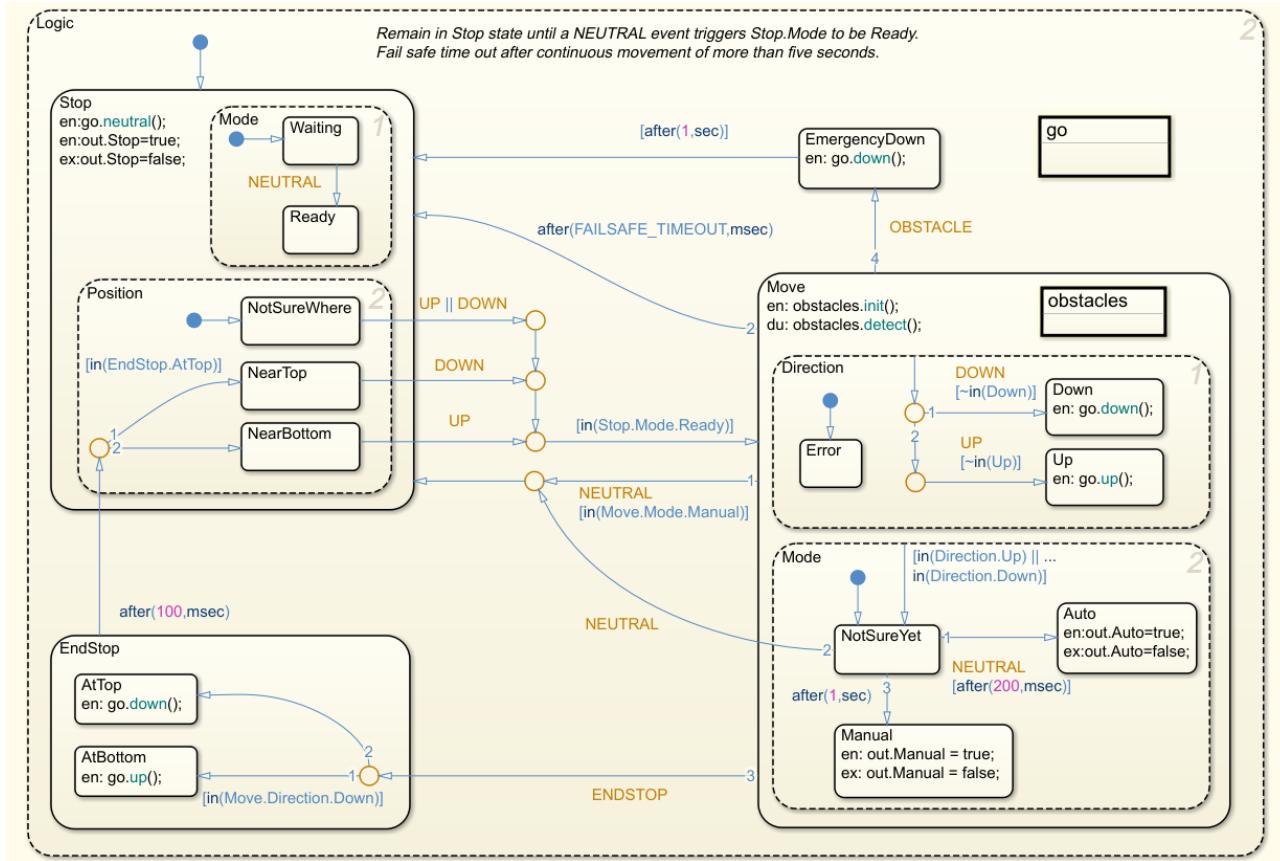


Figure 43 : modélisation du comportement d'une boîte de vitesse automatique avec Stateflow

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 **Multibody**, 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 aisément pour l'utilisateur d'approfondir l'exploitation de ces outils en fonction des besoins de la démarche de modélisation menée.

MATLAB, Simulink

Simscape

Electrical

Multibody

Fluids

Drivelines

Stateflow

Stateflow

Modélisation du comportement combinatoire et séquentiel d'un système

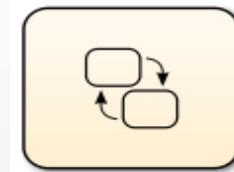
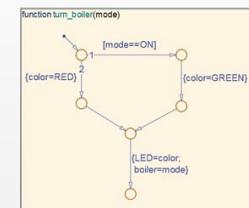


Diagramme d'états



Flux logiques



Table de vérité

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

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.

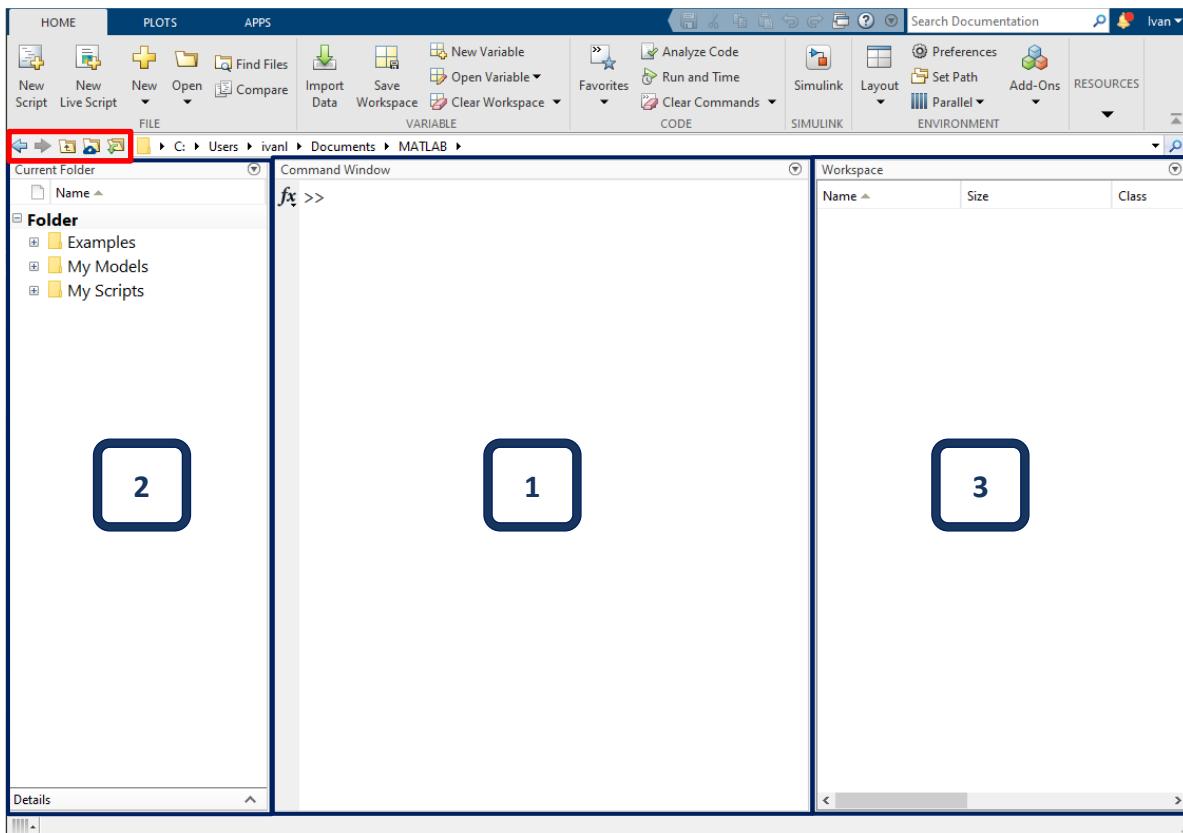


Figure 45 : la fenêtre de l'environnement MATLAB

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.

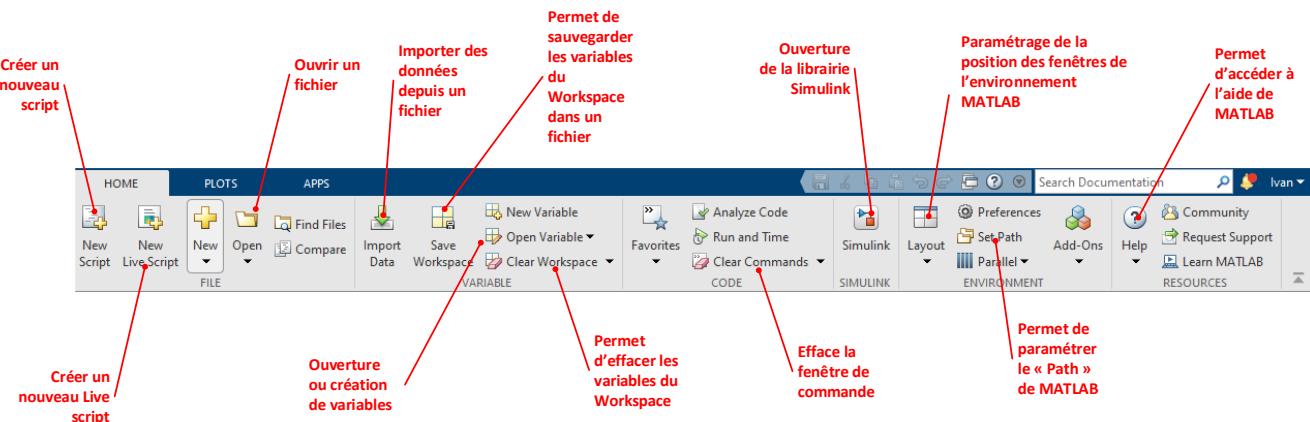


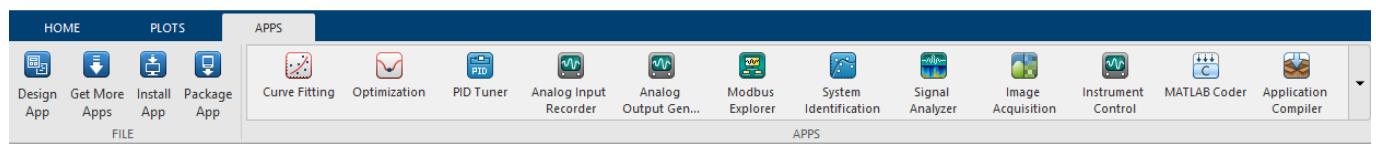
Figure 46 : la barre de commande de l'environnement MATLAB

La barre de commande possède également deux onglets complémentaires :

- L'onglet « PLOT » qui permet de tracer des courbes à partir de variables stockées dans le Workspace sans avoir à utiliser du code.



- L'onglet « APPS » qui permet d'accéder à des applications en fonction des besoins de l'utilisateur (interpolation d'une série de données, réglage d'un correcteur...)



C. La fenêtre de l'environnement Simulink.

Pour lancer Simulink cliquer sur l'icône



afin d'ouvrir la Simulink Start Page (Figure 47) qui

permet d'ouvrir un fichier Simulink.

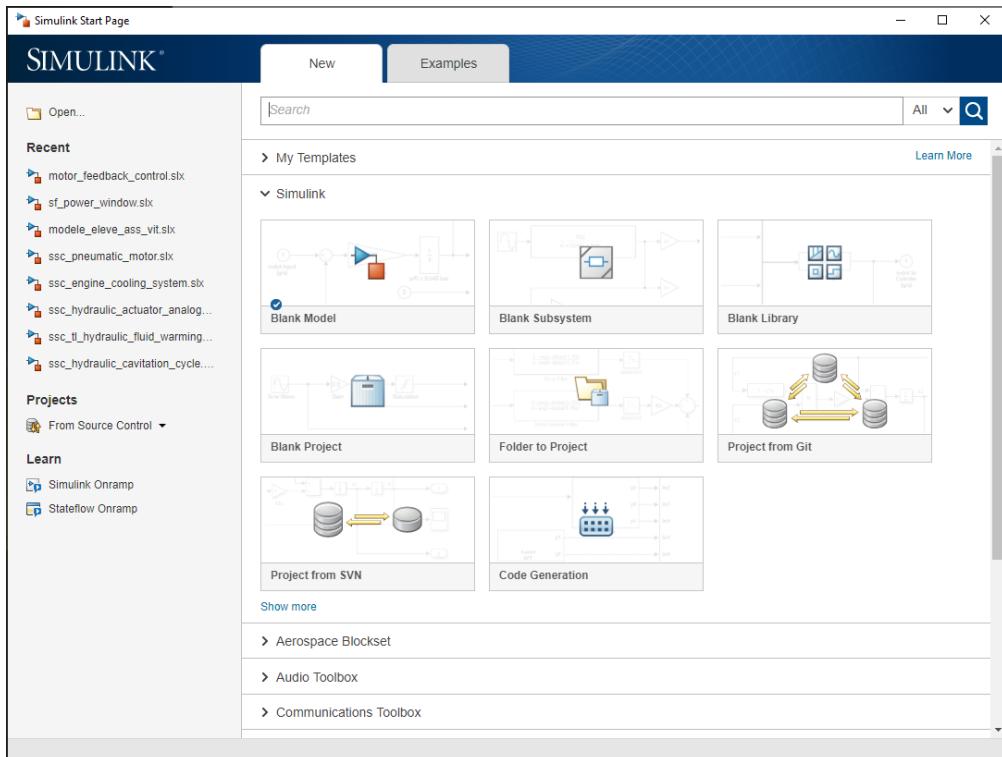


Figure 47 : Simulink Start Page

A partir de cette page, il est possible d'ouvrir un nouveau modèle. Afin d'accélérer le processus de conception Simulink propose différents modèles de fichiers en fonction des besoins spécifiques de l'utilisateur et des outils de modélisation qui seront utilisés. Ces modèles proposent donc des environnements de modélisation adaptés en termes de choix de solveur, de mise en place par défaut des blocs les plus utiles, de choix du format d'exportation des variables de la modélisation... Dans un premier temps nous allons découvrir l'environnement de modélisation et nous partirons d'un modèle vierge « Blank Model » afin de paramétrier manuellement les fonctionnalités qui nous intéressent au fur et à mesure. Plus tard avec de l'expérience, il sera plus efficace de partir d'un modèle adapté à l'objectif de la modélisation.

A partir de la Simulink Start Page sélectionner un « Blank Model » (modèle vierge).

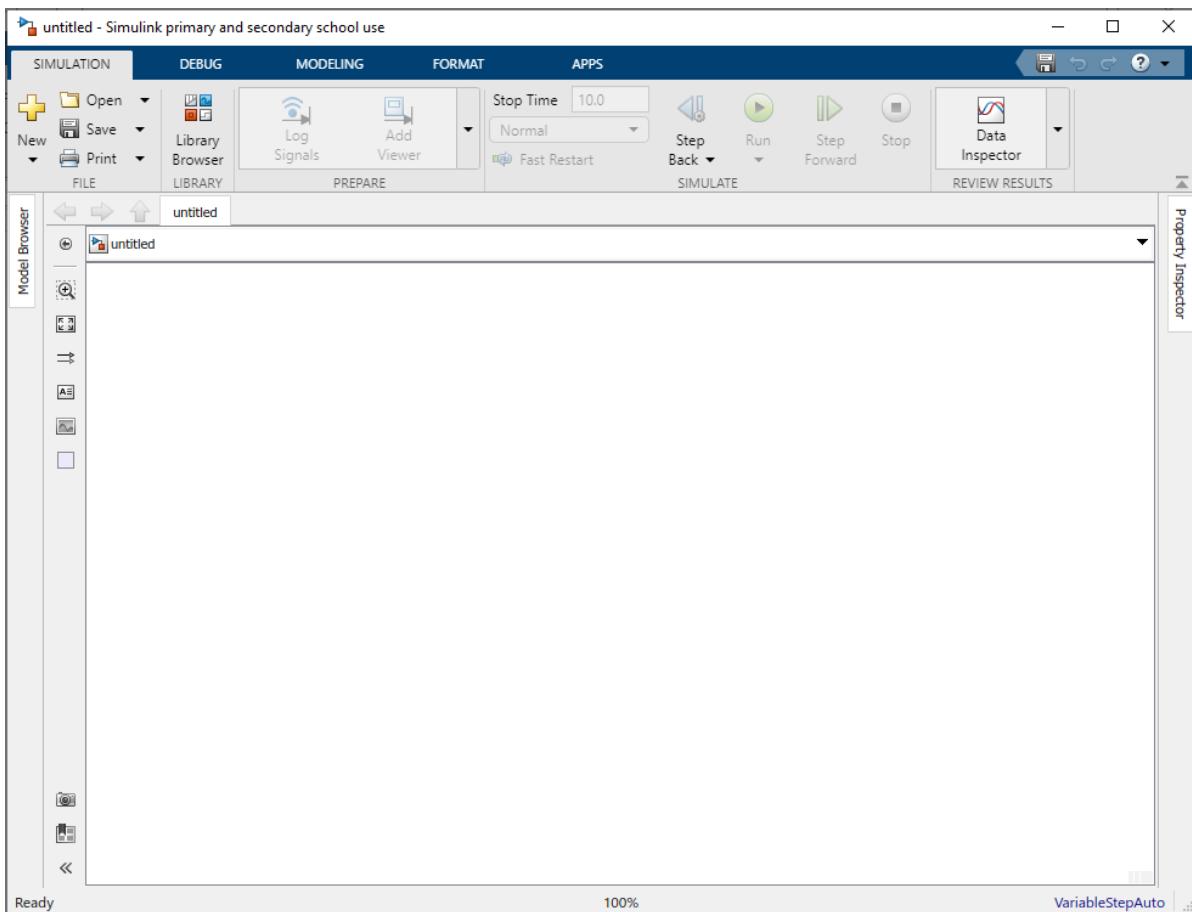


Figure 48 : Blank Model Simulink

Ouvrir la librairie Simulink en cliquant sur



La librairie de Simulink s'ouvre (Figure 49) et propose toute une palette de composants pour la modélisation causale ou acausale.

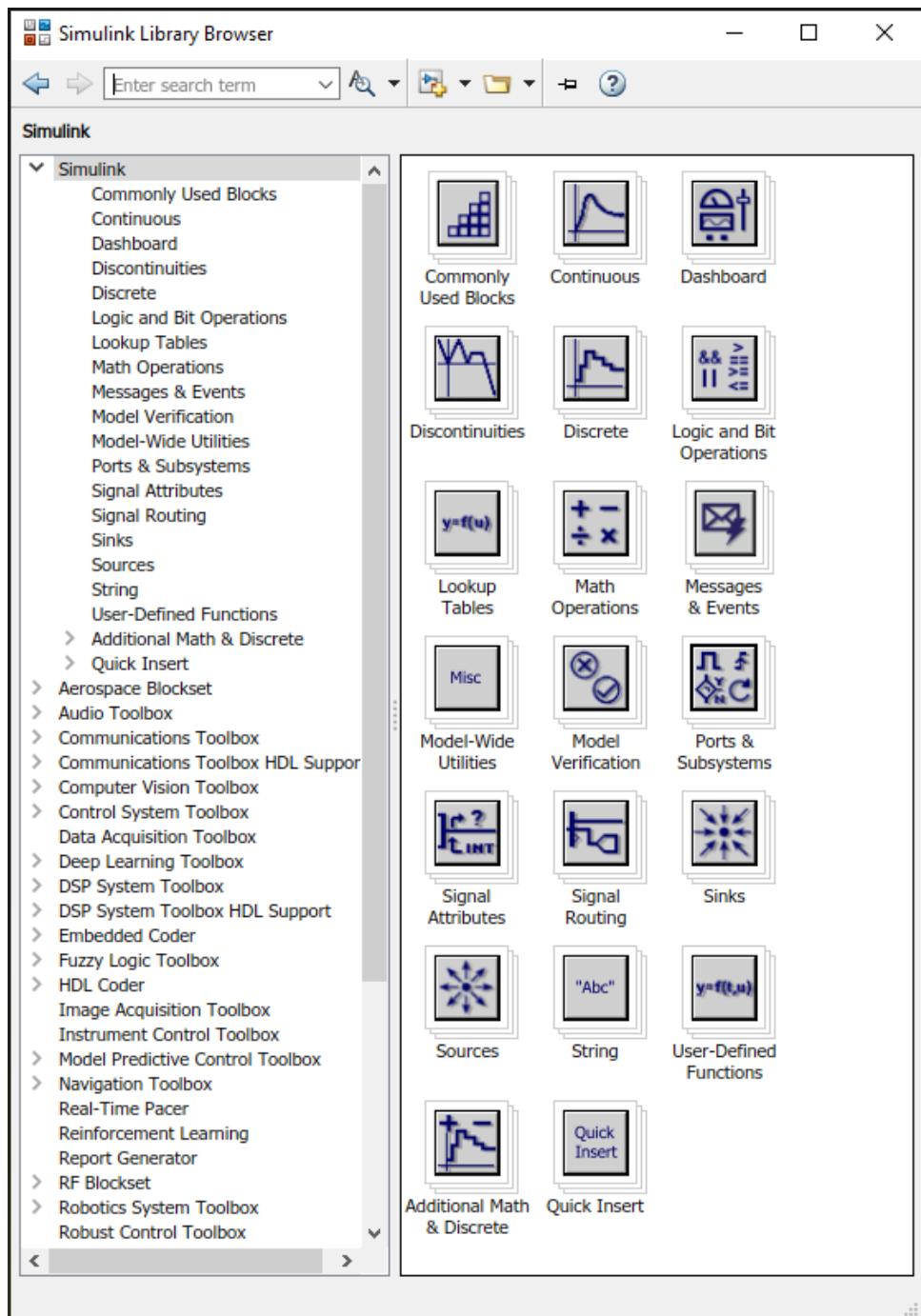


Figure 49 : la bibliothèque de Simulink

La Figure 50 décrit les fonctionnalités de la fenêtre de l'environnement Simulink.

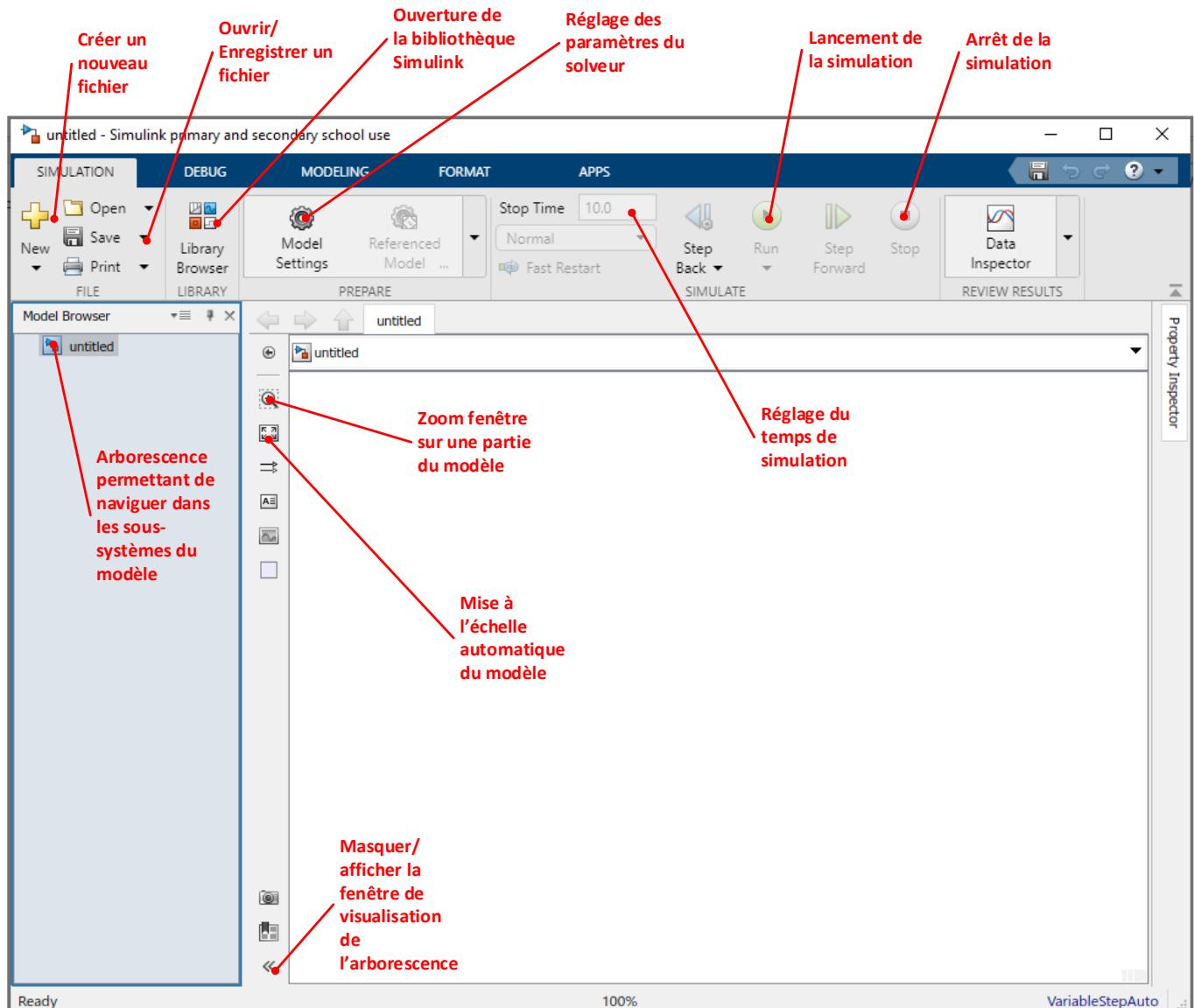


Figure 50 : la fenêtre de l'environnement Simulink

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 :

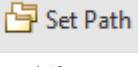
- Ne pas utiliser d'espace
- Ne pas utiliser de caractères accentués

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 ou deux fonctions ne doivent pas porter le même nom. Il donc important de maîtriser 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 51).

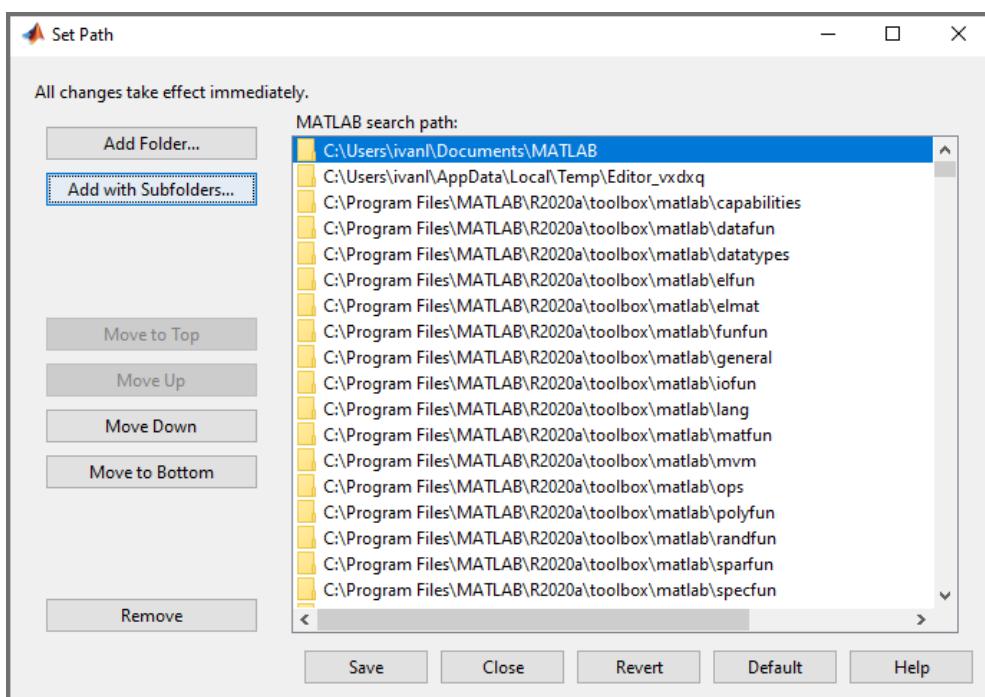


Figure 51 : le path de MATLAB

Sélectionner **Add with Subfolders**, puis choisir les dossiers que vous voulez ajouter dans le « path » de MATLAB (les sous-dossiers seront automatiquement ajoutés) et tous les dossiers et sous-dossiers ajoutés par cette procédure seront dans le « Path » pour toute les sessions futures.

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 modifier 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 52). 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 utilisés dans l'ouvrage, ajouter le dossier contenant tous les modèles du livre dans le « Path » de MATLAB.

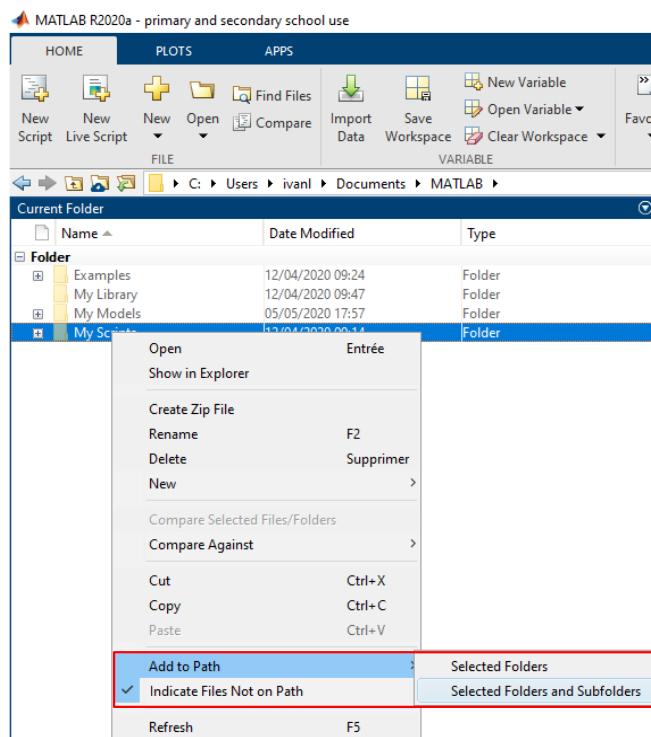


Figure 52 : ajout de dossier au « path » pour la session courante

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

Chapitre 3 : Stratégie de conception d'un modèle multi-physique

I. 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**.

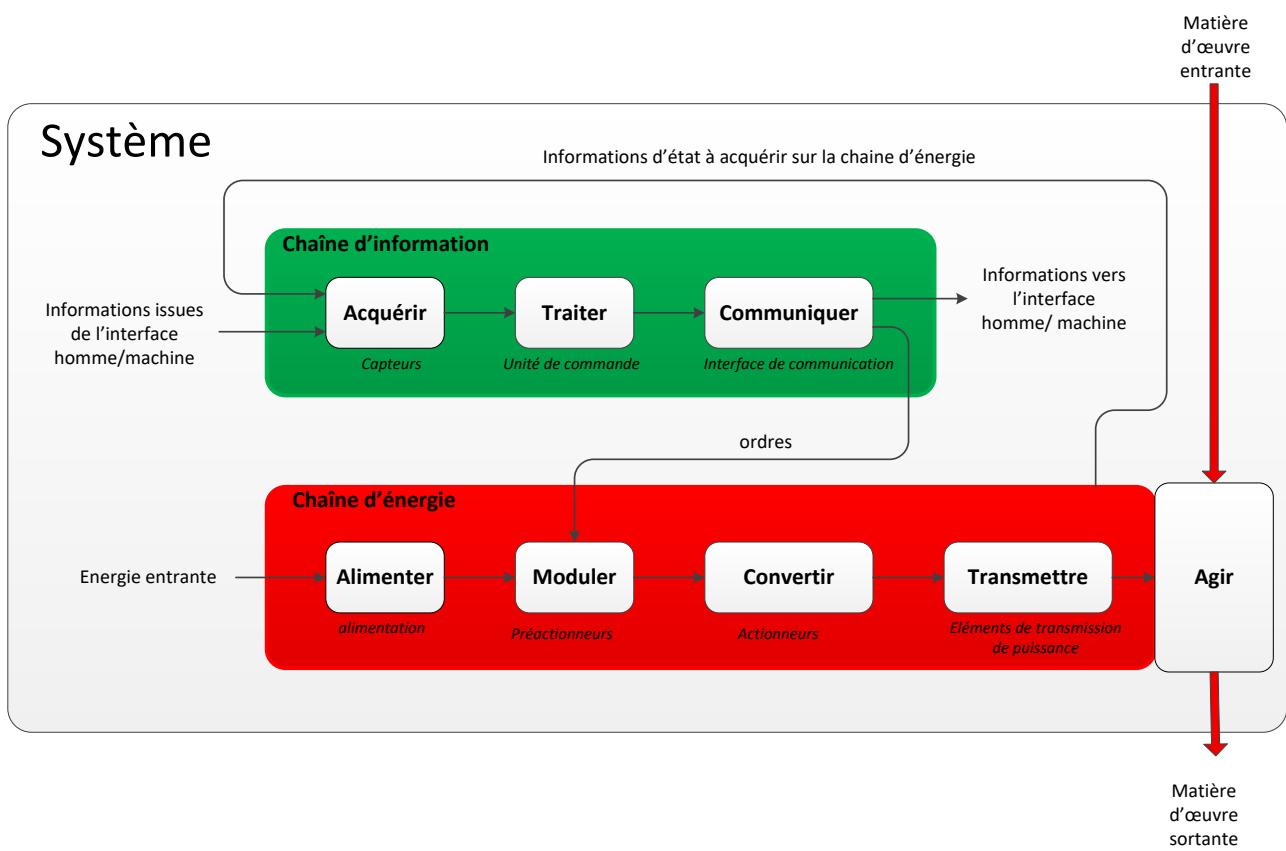


Figure 53 : le diagramme Chaîne d'énergie / Chaîne d'information

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. La toolbox **System Composer** permet de construire un diagramme de blocs internes et d'y inclure les modèles.

Le diagramme de la Figure 54 illustre cette démarche.

MATLAB-Simulink

Stateflow

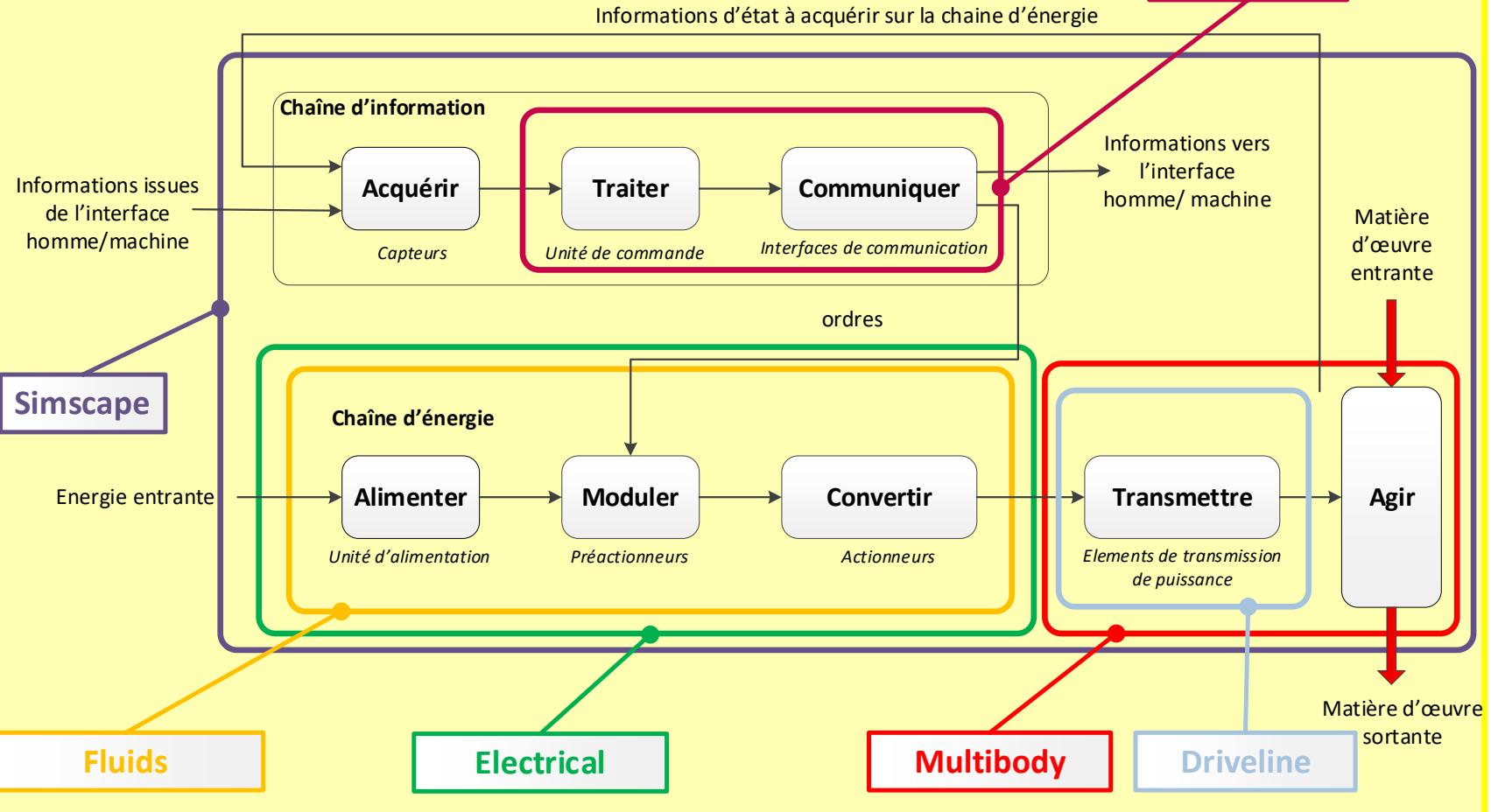


Figure 54 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB

II. 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 (Figure 55).

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 entraîne 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.

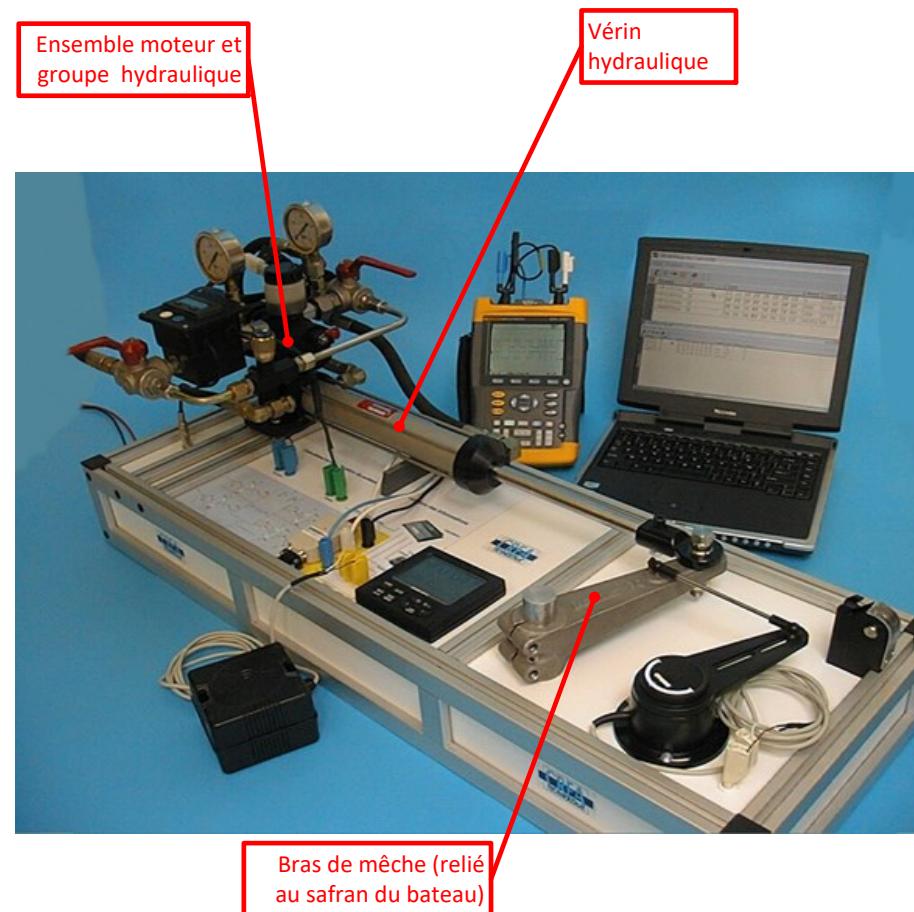


Figure 55 : photo du pilote automatique de bateau

A. Diagramme présentant la chaîne d'énergie et d'information du pilote hydraulique de bateau

La Figure 56 présente le diagramme chaîne d'énergie/chaîne d'information du pilote de bateau et les outils choisis pour modéliser les différentes parties du système.

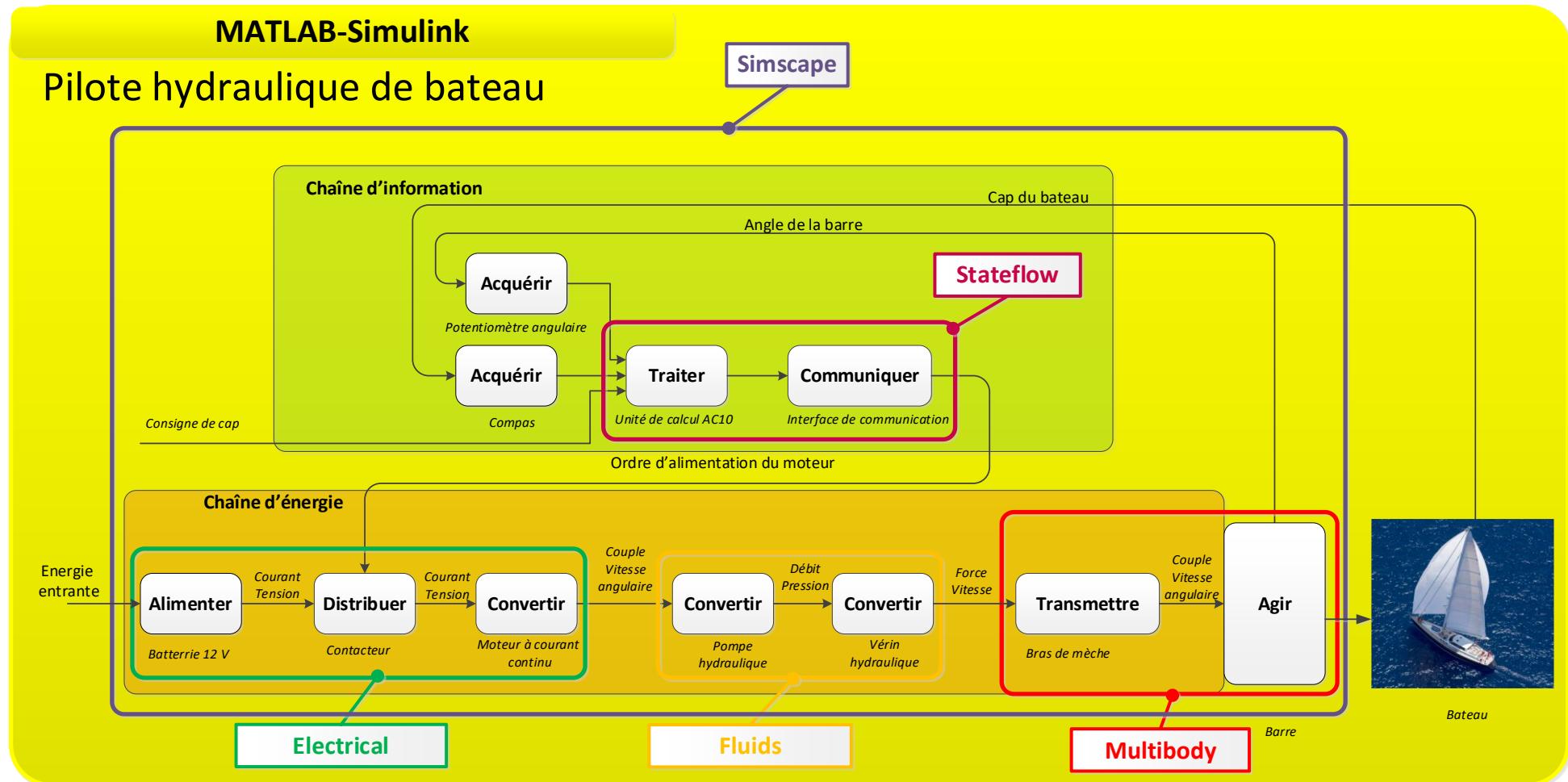


Figure 56 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB

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

Le modèle multi-phérique 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-phérique.

En double cliquant sur un sous-système, il est possible d'accéder à la modélisation. Le modèle multi-phérique devient également un outil d'aide à la description qui permet d'explorer de manière dynamique les composants qui réalisent les fonctions.

La Figure 57 présente une copie d'écran du modèle multi-phérique réalisé avec MATLAB – Simulink.

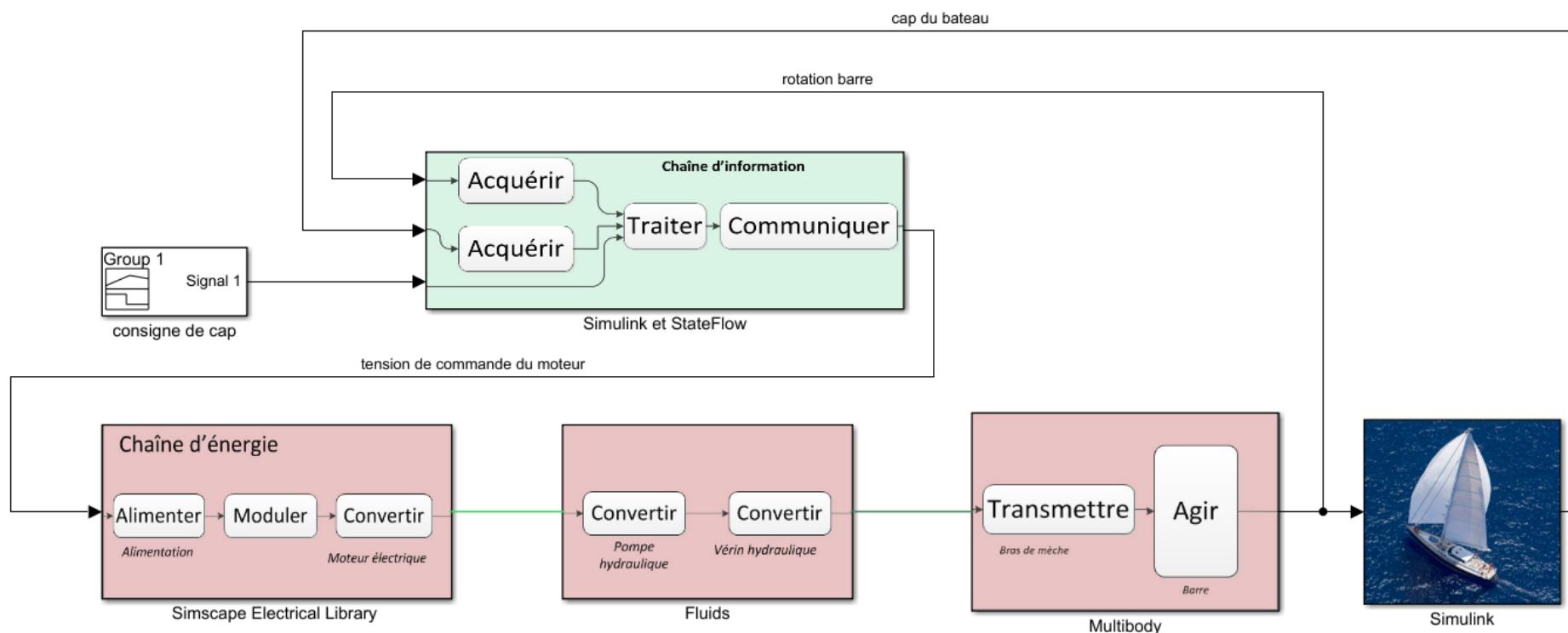


Figure 57 : copie d'écran du modèle multi-phérique du pilote automatique de bateau réalisé avec MATLAB

C. Chargement et simulation du modèle

Afin de pouvoir utiliser l'ensemble des fichiers de simulation utilisés dans l'ouvrage, le dossier contenant tous les modèles du livre doit impérativement être inclus dans le « Path » de MATLAB (procédure décrite page 55).

D. Visualisation des résultats issus du modèle multi-phérique

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.

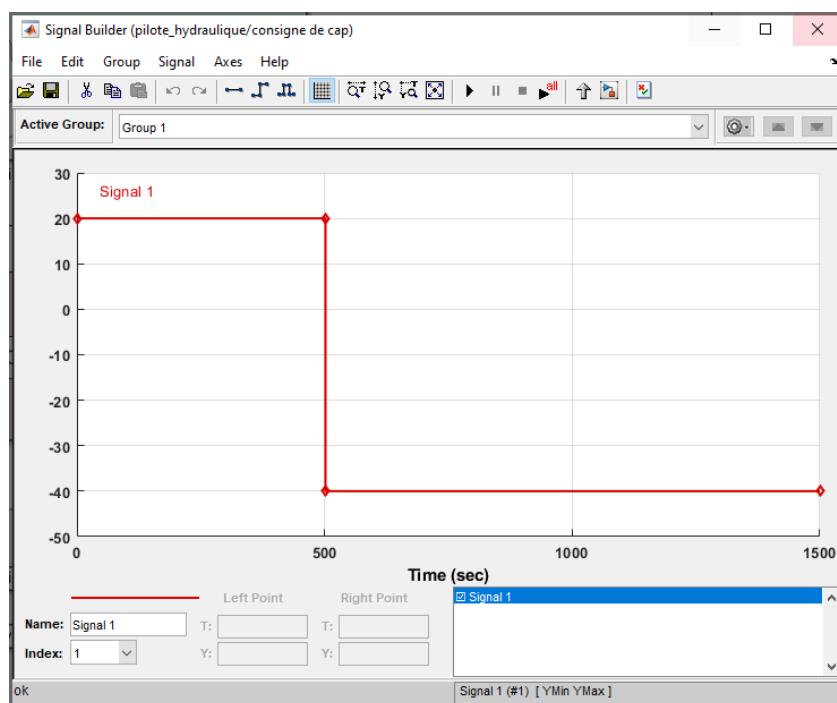


Figure 58 : visualisation de la consigne de cap



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 **Multibody** (Figure 59).

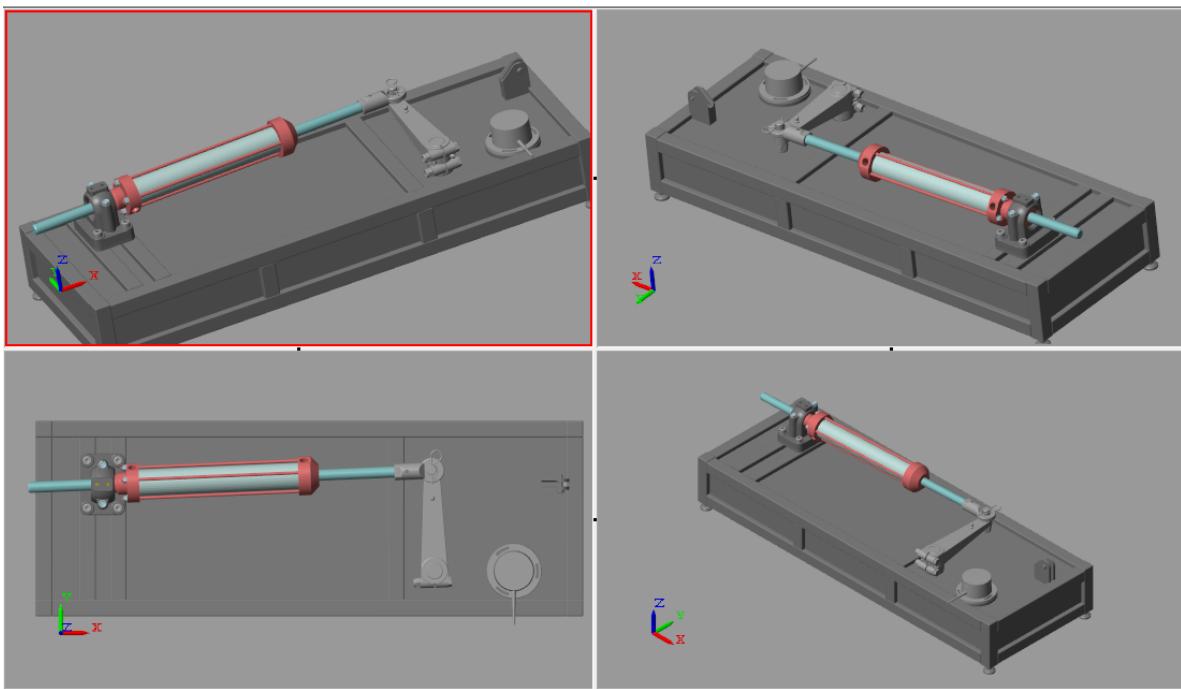


Figure 59 : fenêtre de visualisation de Multibody (Mechanics Explorer)

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



Figure 60 : barre de visualisation de Multibody

Cliquez sur l'icône dans la partie supérieure de la fenêtre **Mechanics Explorers** (Figure 60) pour revenir à un affichage avec une seule fenêtre (Figure 61).

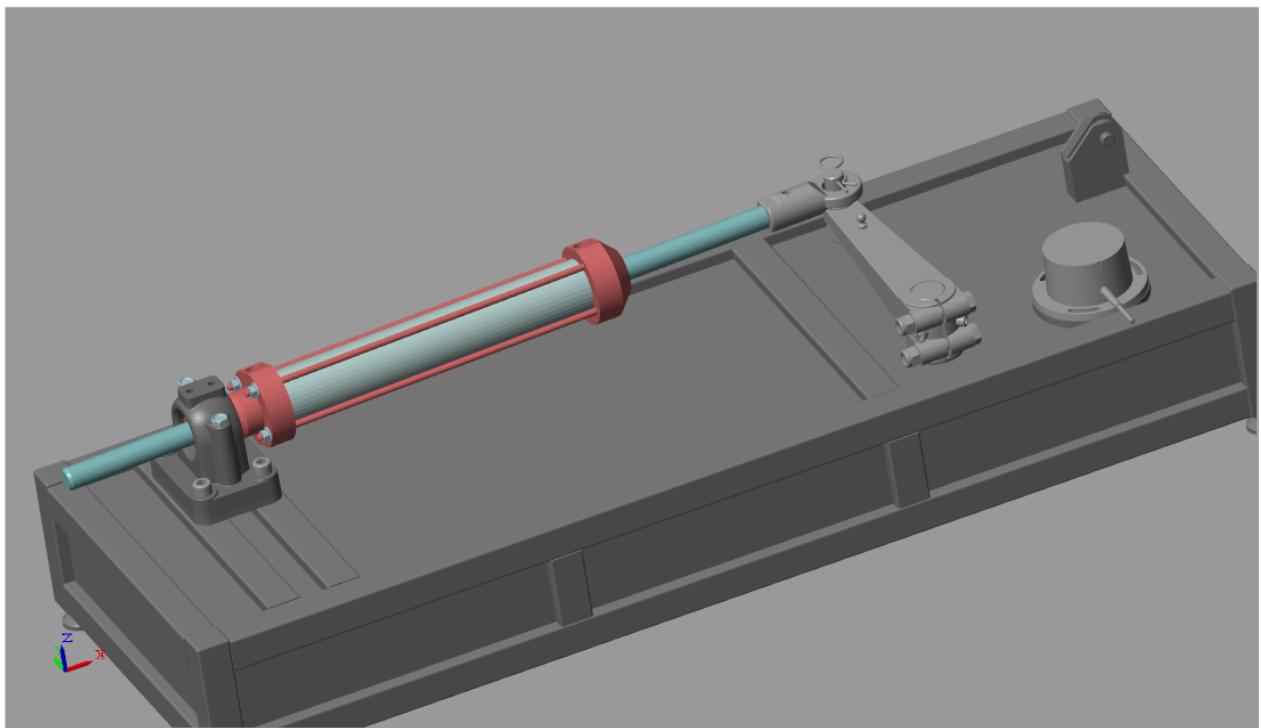


Figure 61 : fenêtre de visualisation de Multibody (Mechanics Explorer)

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 62).

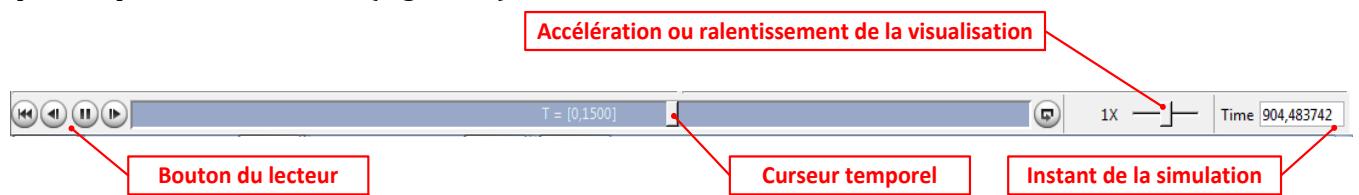


Figure 62 : barre temporelle de Multibody

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.

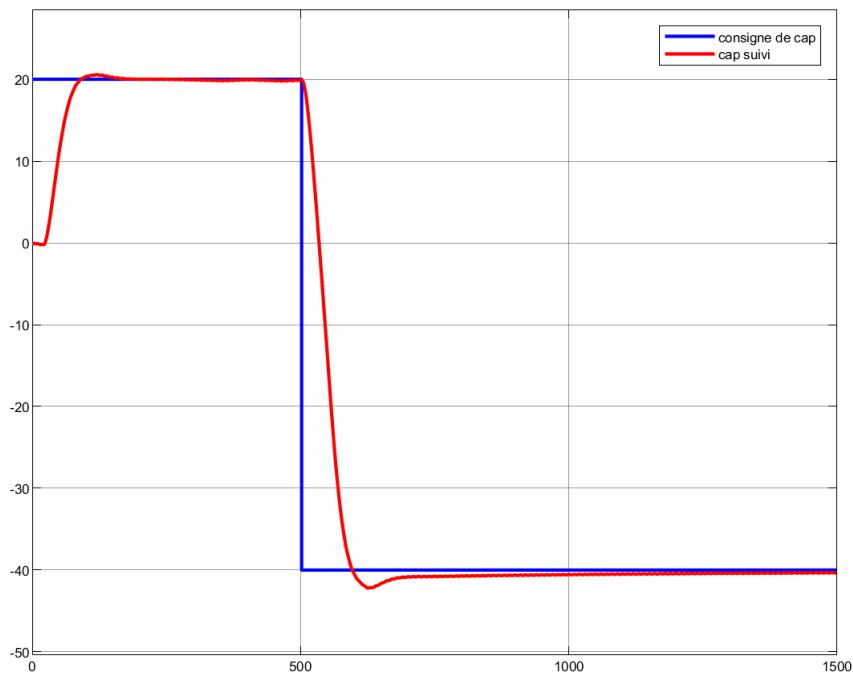


Figure 63 : visualisation de la consigne de cap et du cap effectif suivi par le bateau

Le courant, la tension et le couple du moteur :

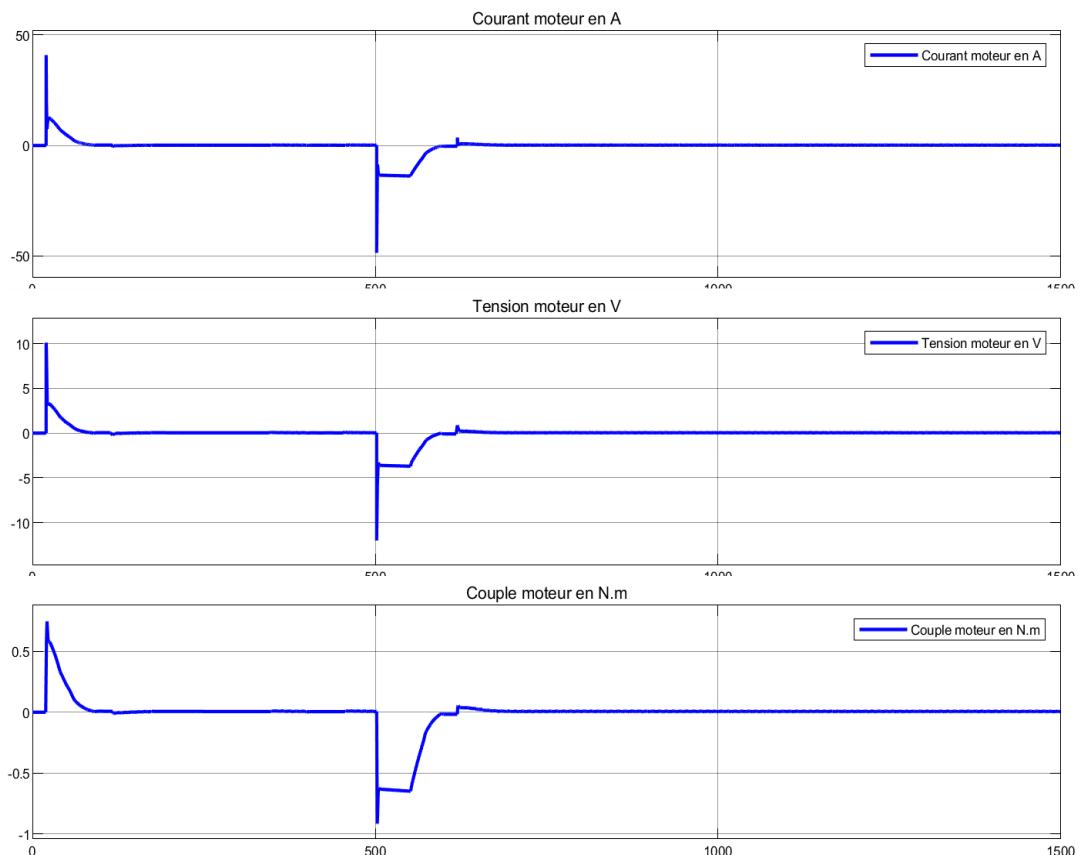


Figure 64 : visualisation des grandeurs relatives au moteur

L'angle de rotation de la barre :

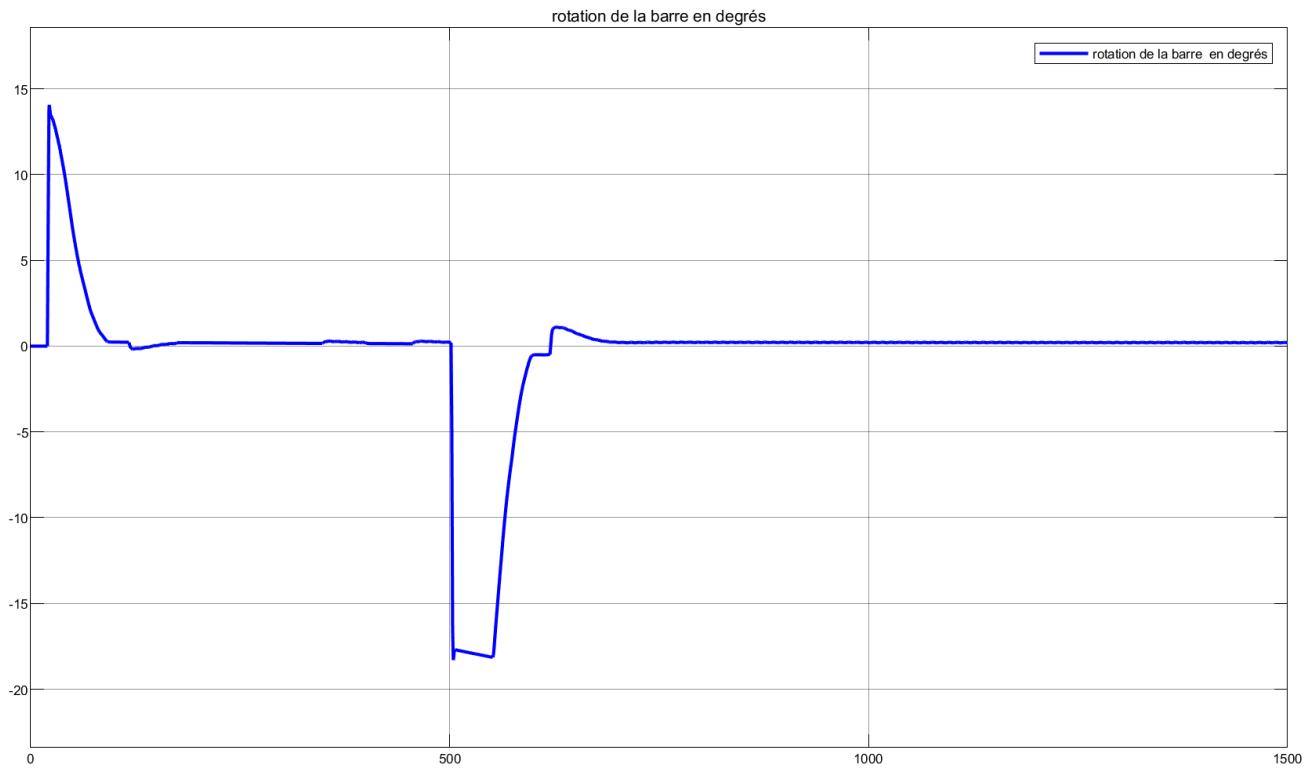


Figure 65 : visualisation de l'angle de rotation de la barre

Les pressions dans les chambres avant et arrière du vérin :

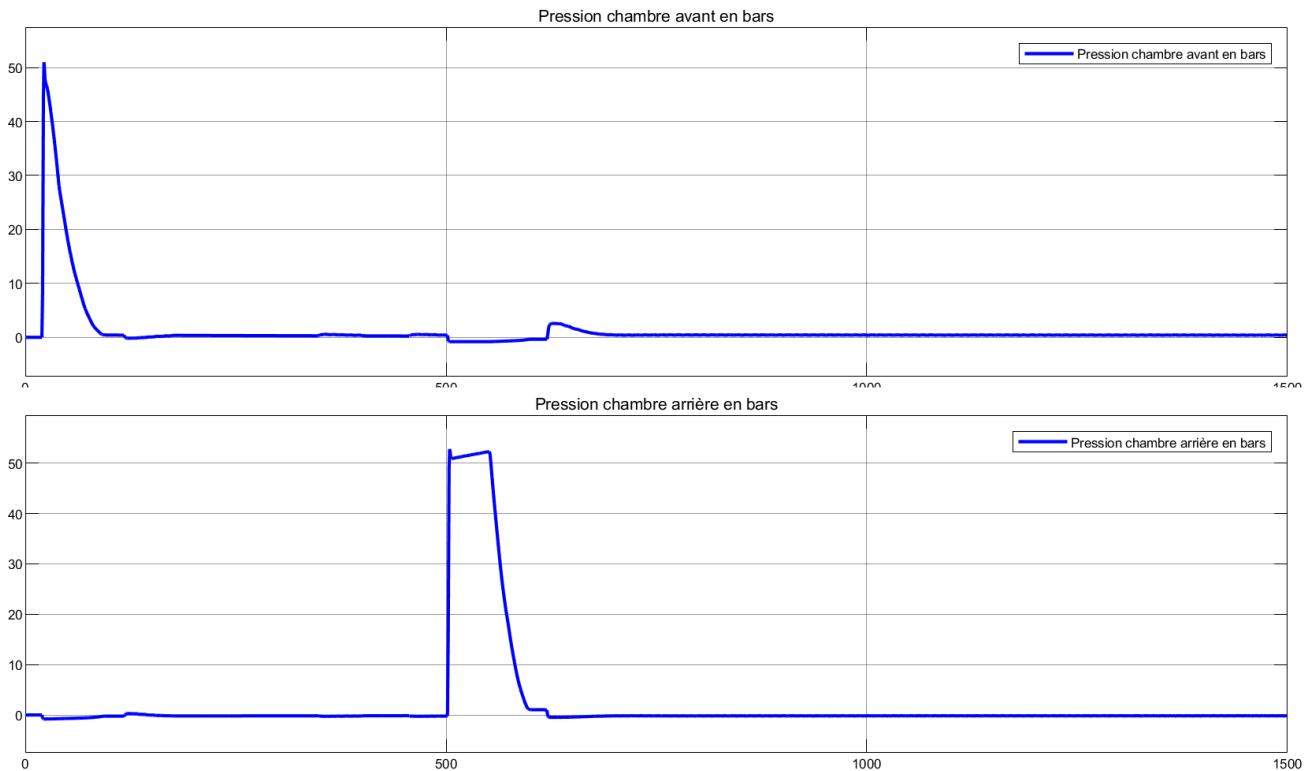


Figure 66 : visualisation des pressions dans les chambres avant et arrière du vérin

Les débits dans les chambres avant et arrière du vérin :

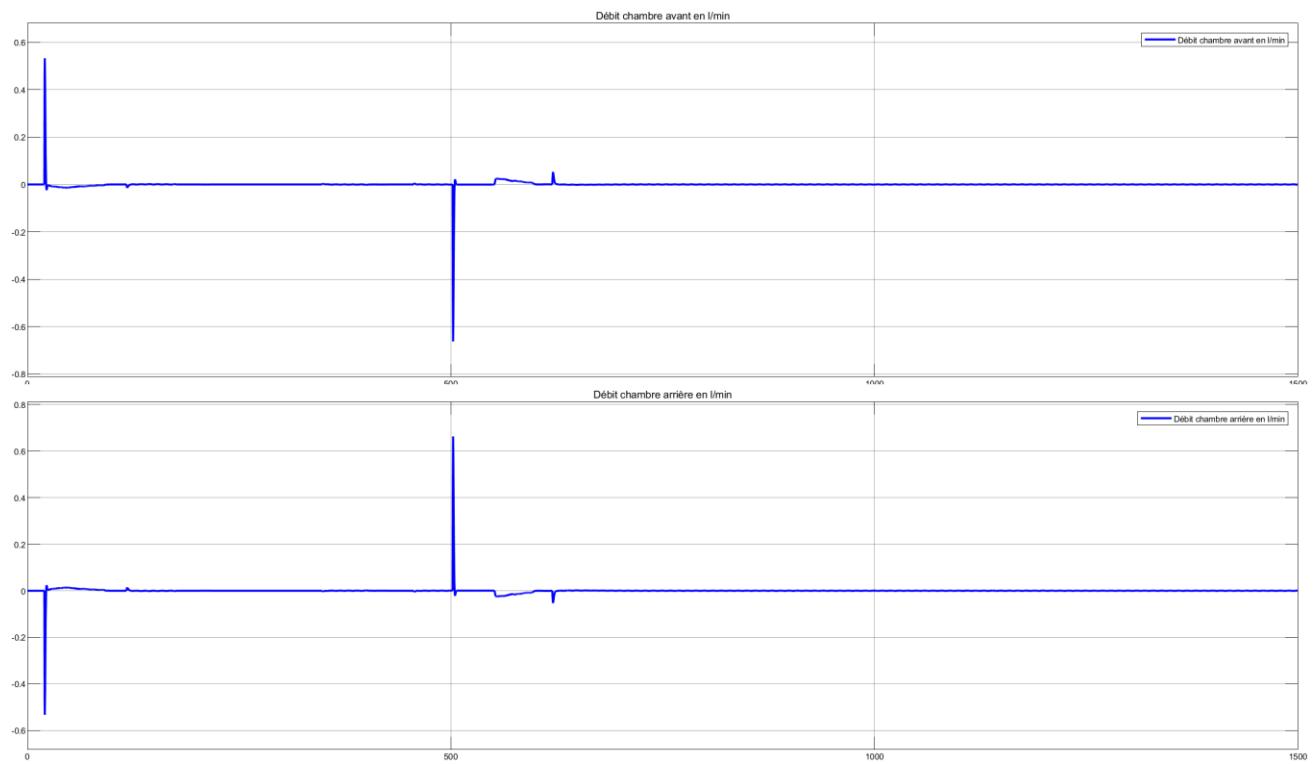


Figure 67 : visualisation des débits dans les chambres avant et arrière du vérin

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** (Figure 68). 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.

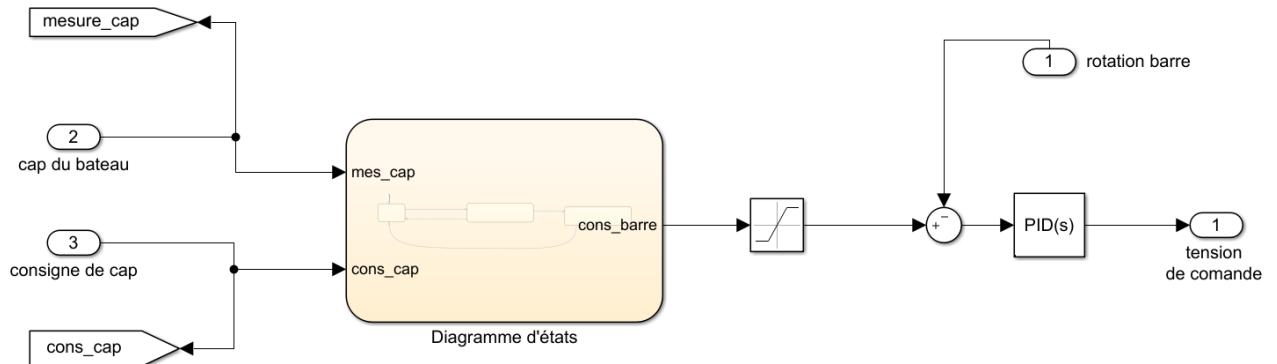


Figure 68 : modèle MATLAB – Simulink de la chaîne d'information du pilote automatique

Diagramme d'état de la boucle de cap :

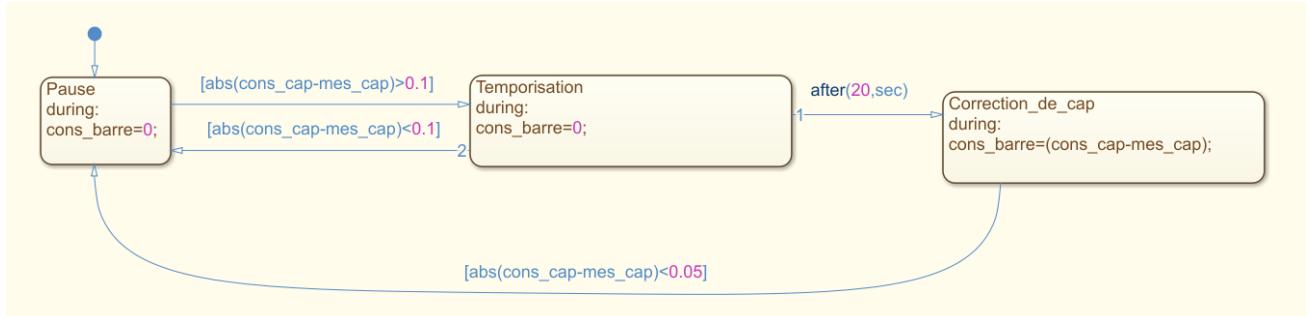


Figure 69 : modélisation de la commande de cap avec des diagrammes d'états.

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.

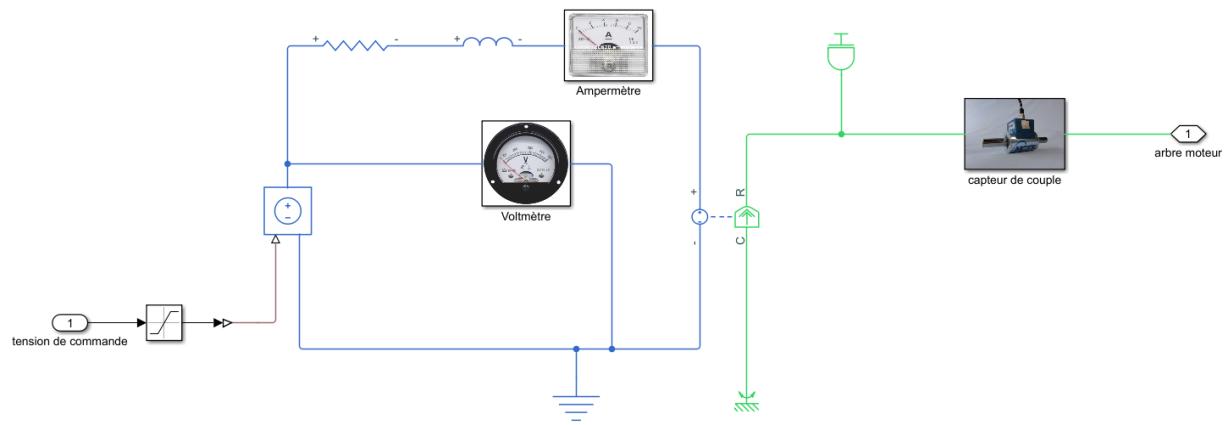


Figure 70 : modèle Simscape de la partie électrique de la chaîne d'énergie du pilote automatique

3. Exploration de la chaîne d'énergie : Simscape Fluids

La partie hydraulique de la chaîne d'énergie est réalisée à l'aide de composants extraits de la bibliothèque **Fluids**. 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.

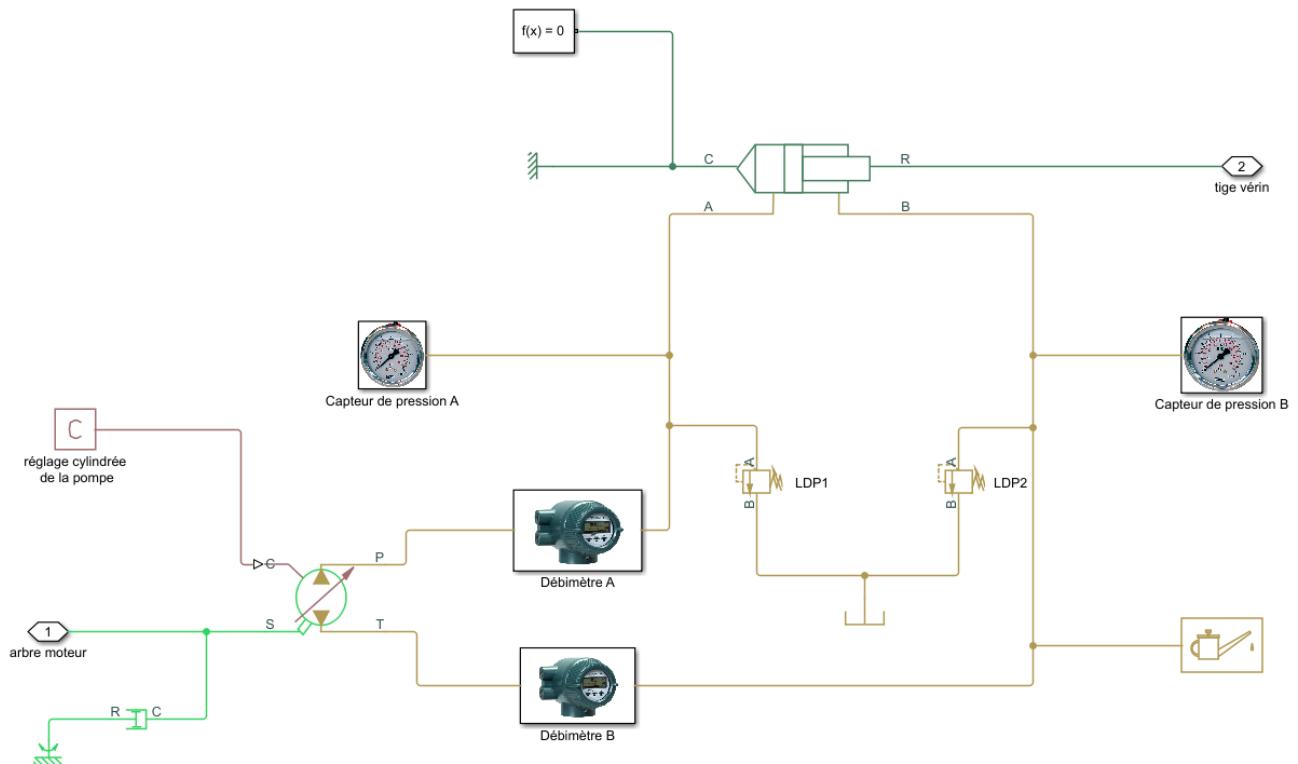


Figure 71 : modèle réalisé avec Simscape Fluids de la partie hydraulique de la chaîne d'énergie du pilote automatique

4. Exploration de la chaîne d'énergie : Simscape Multibody

La partie mécanique de la chaîne d'énergie est réalisée à l'aide de **Multibody**. 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 **Multibody** peut se construire directement avec **MATLAB** ou être importé à partir d'un logiciel de CAO comme **Solidworks**.

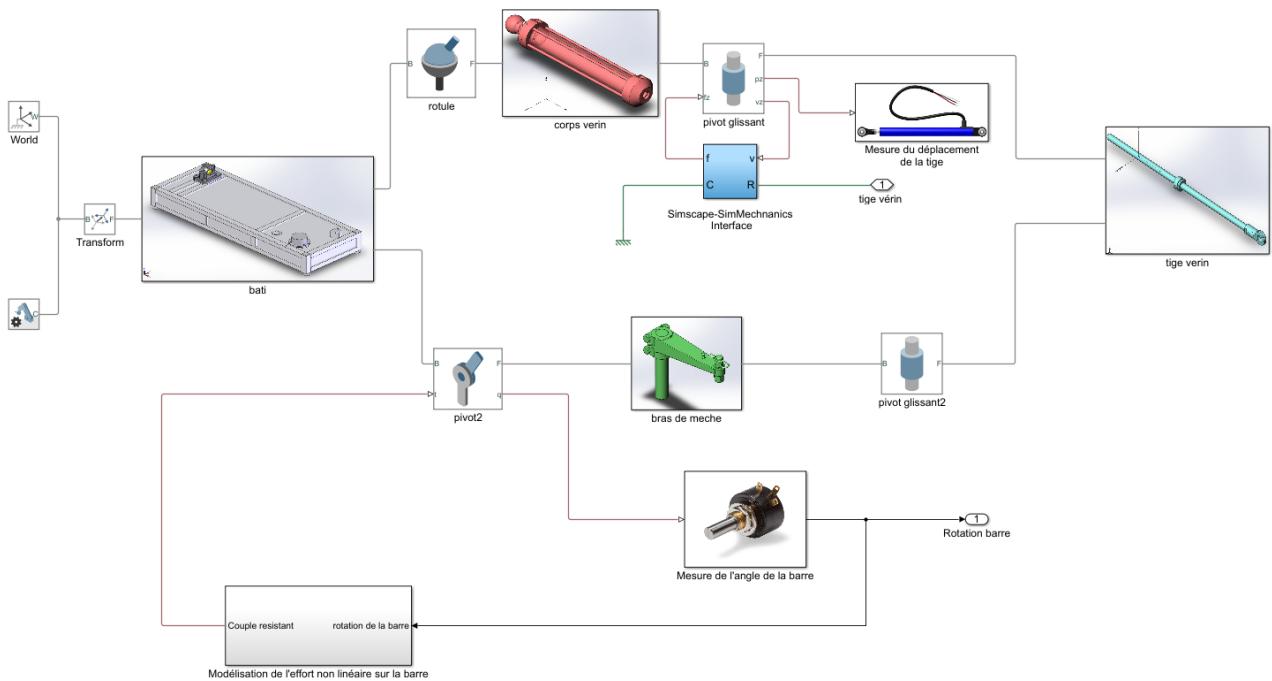


Figure 72 : Modélisation à l'aide de Simscape Multibody de la structure du pilote automatique

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**.

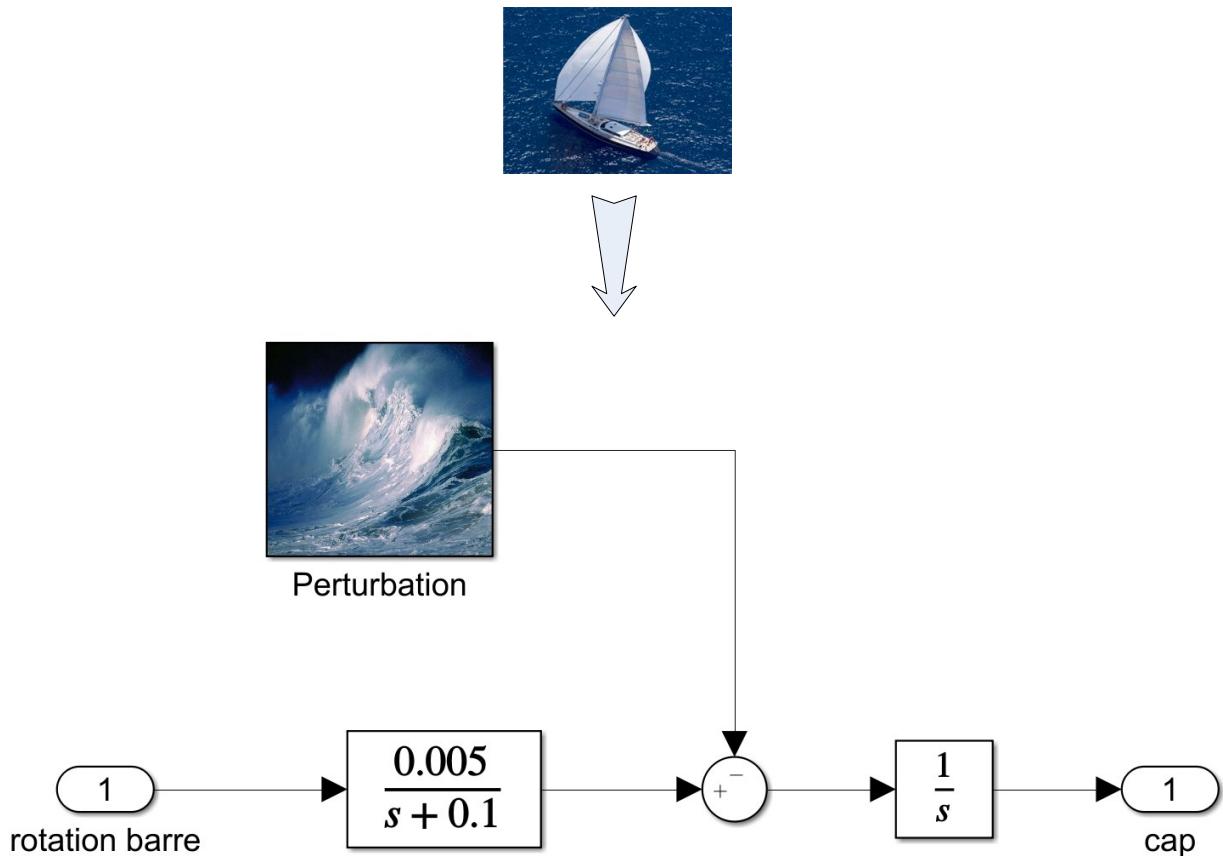


Figure 73 : Modélisation Simulink du comportement dynamique du bateau

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 183).

Ouvrir le fichier « **pilote_hydraulique_dashboard.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.

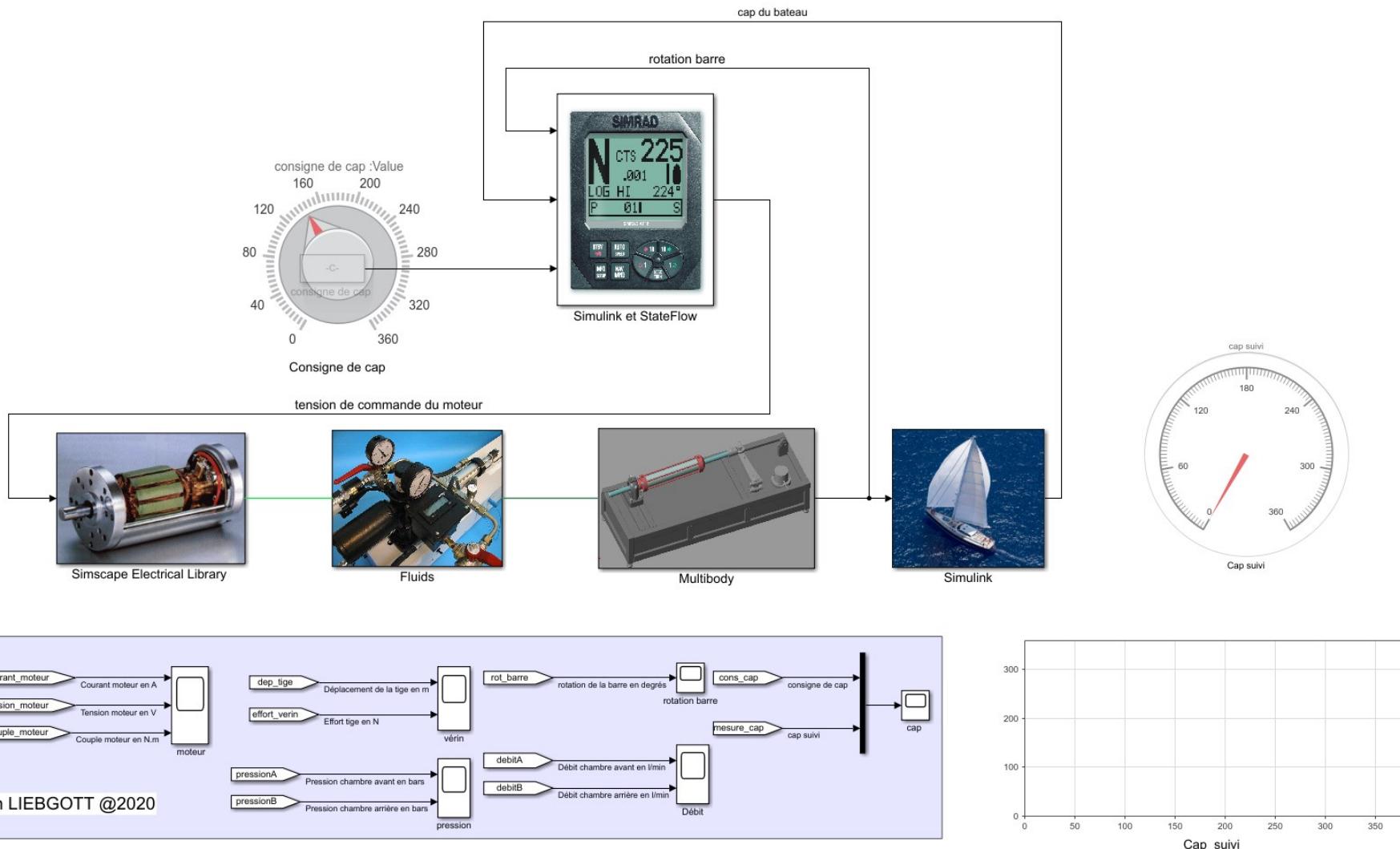


Figure 74 : modèle du pilote hydraulique avec pilotage interactif

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.

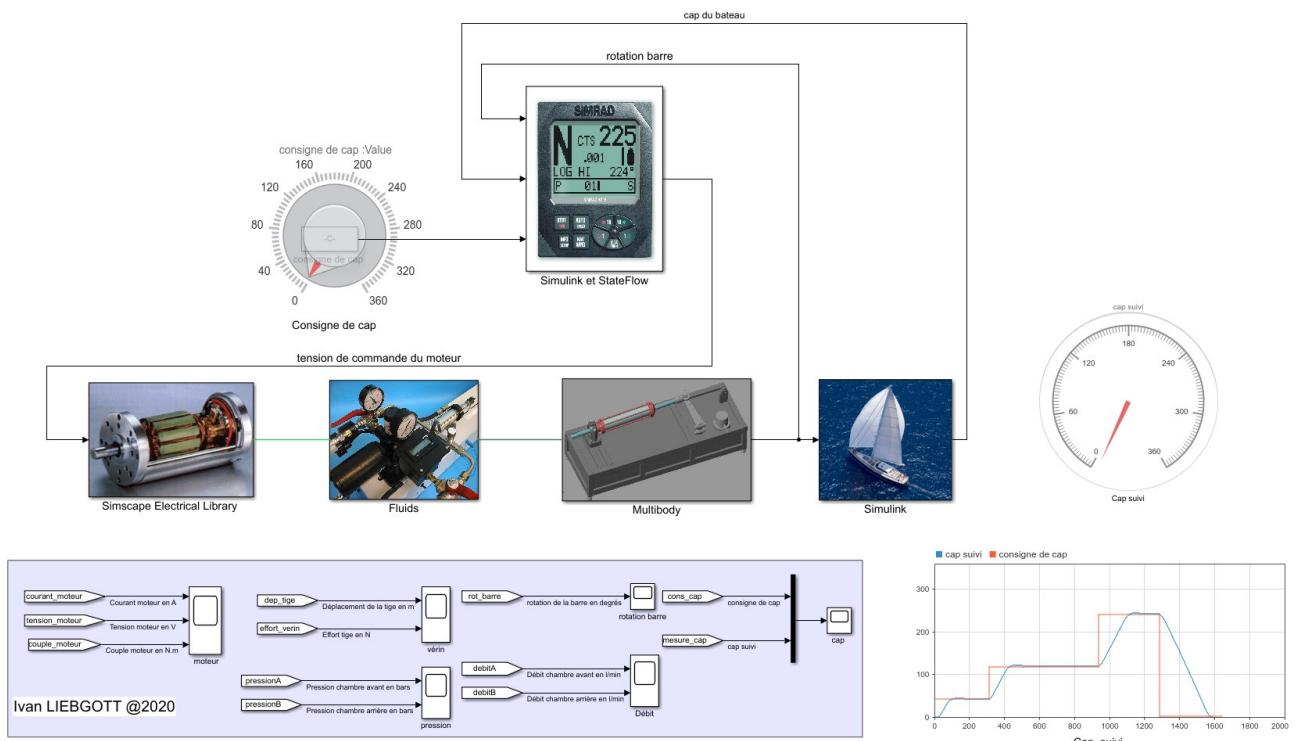


Figure 75 : utilisation du pilotage interactif d'un modèle

III. 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.

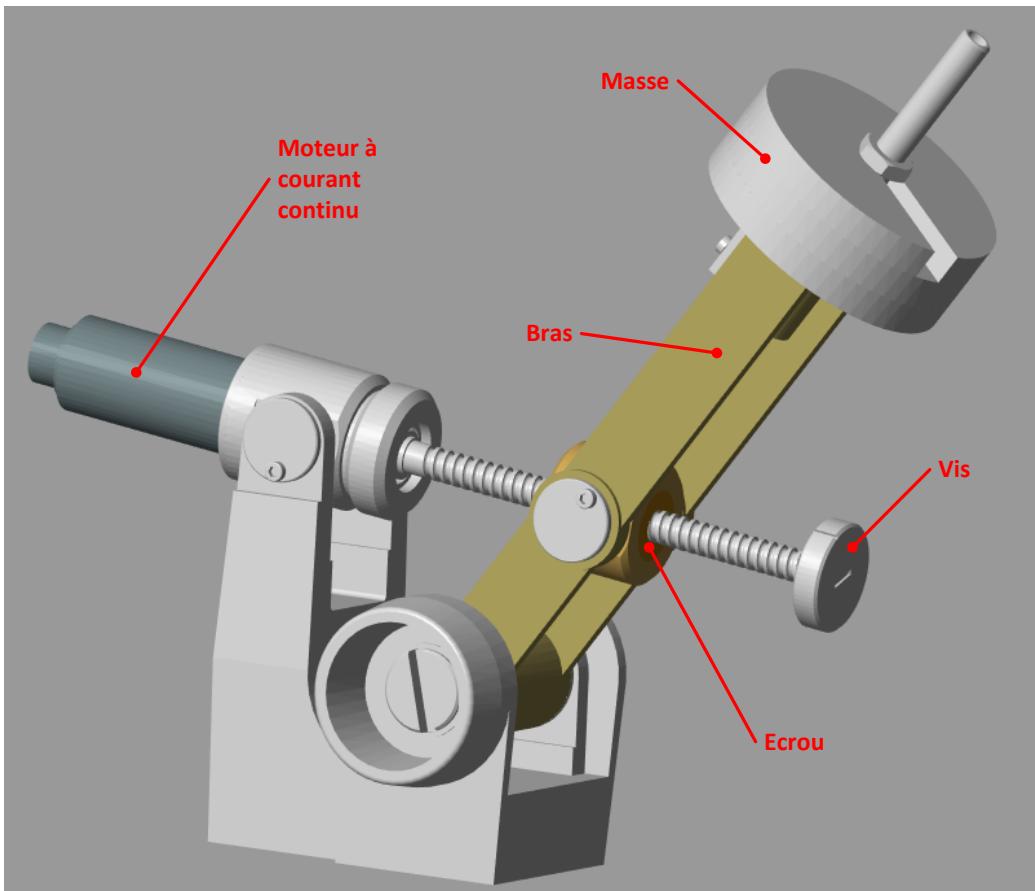


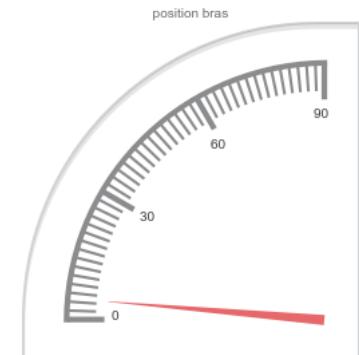
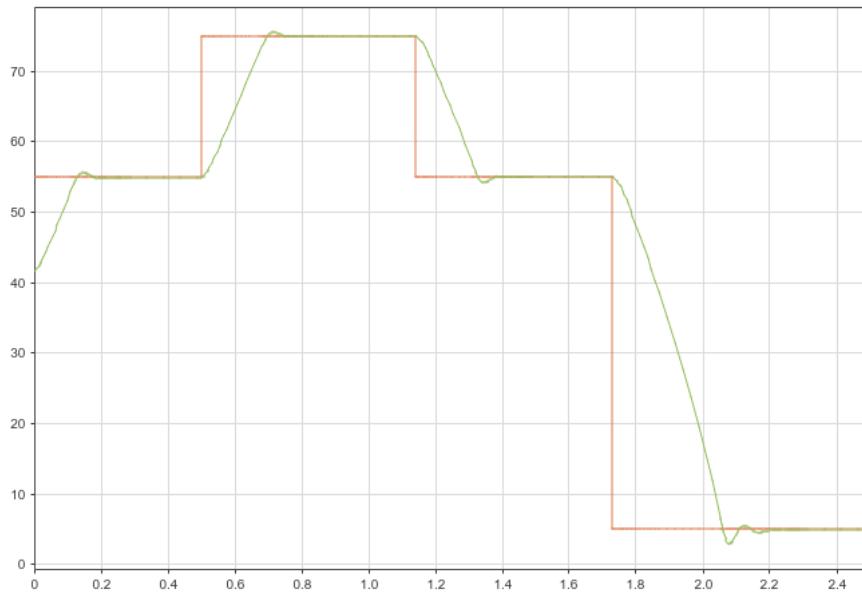
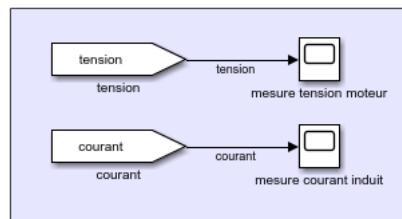
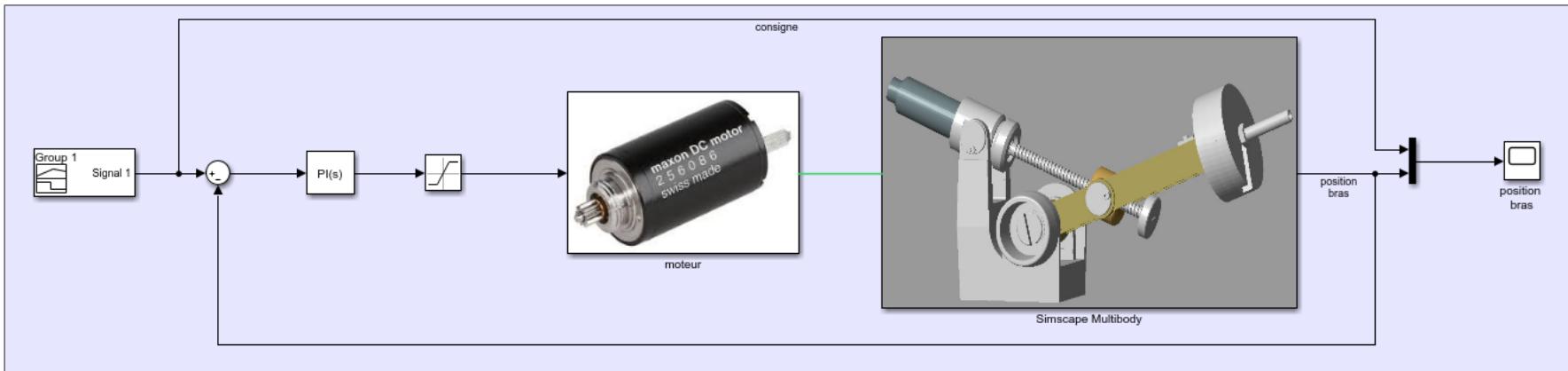
Figure 76 : présentation du robot Maxpid

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 **Mechanics Explorer** permet de visualiser les mouvements du bras du robot.



@ Ivan LIEBGOTT 2020

Figure 77 : modélisation multi-physique du robot MAXPID

Visualisation de la position du bras et de la consigne :

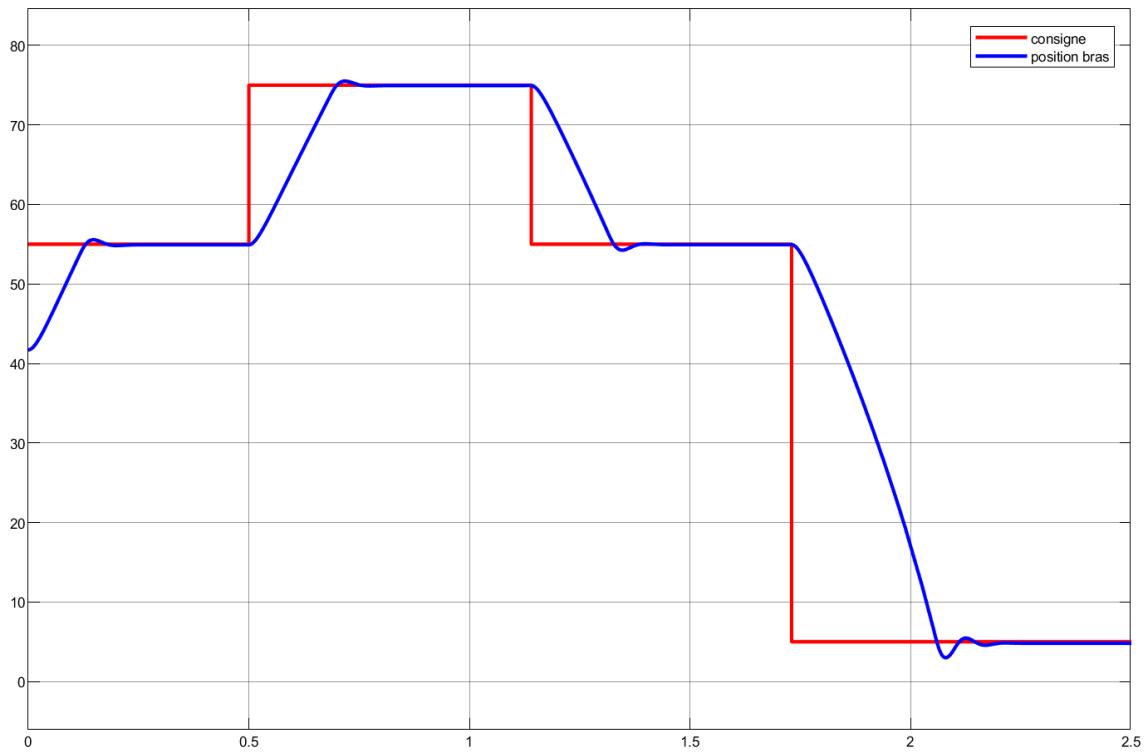


Figure 78 : position du bras du robot MAXPID

Visualisation de la tension de commande :

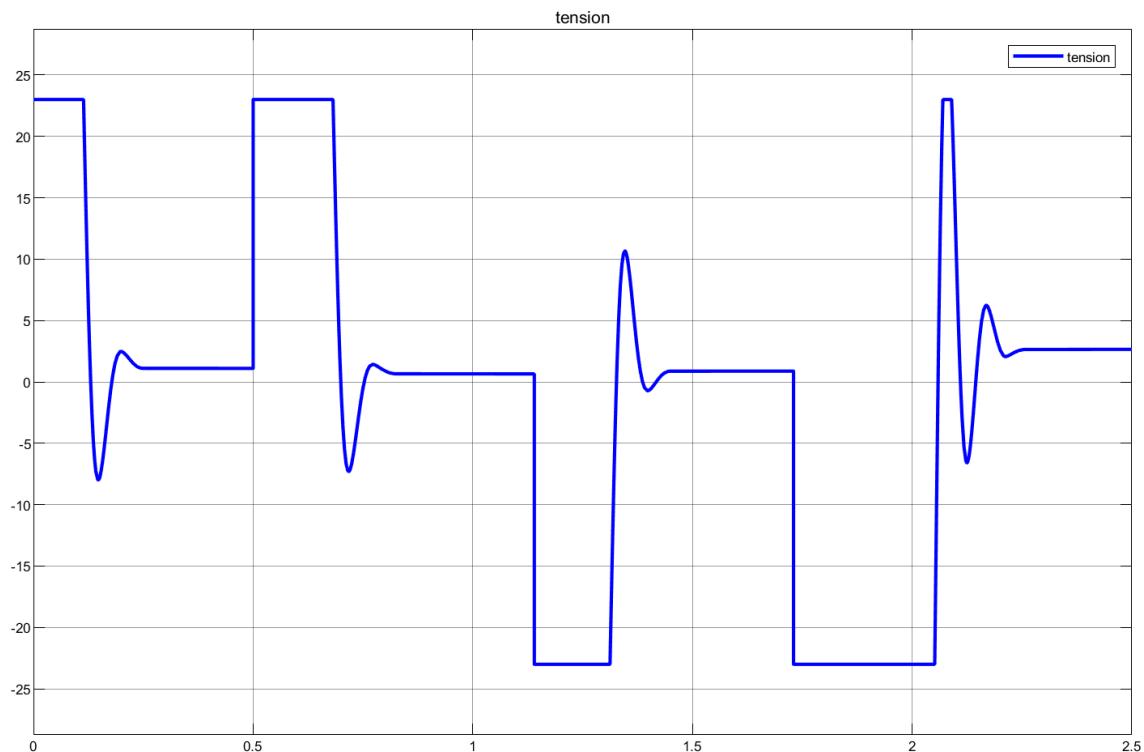


Figure 79 : visualisation de la tension de commande du moteur

Visualisation du courant dans l'induit du moteur :

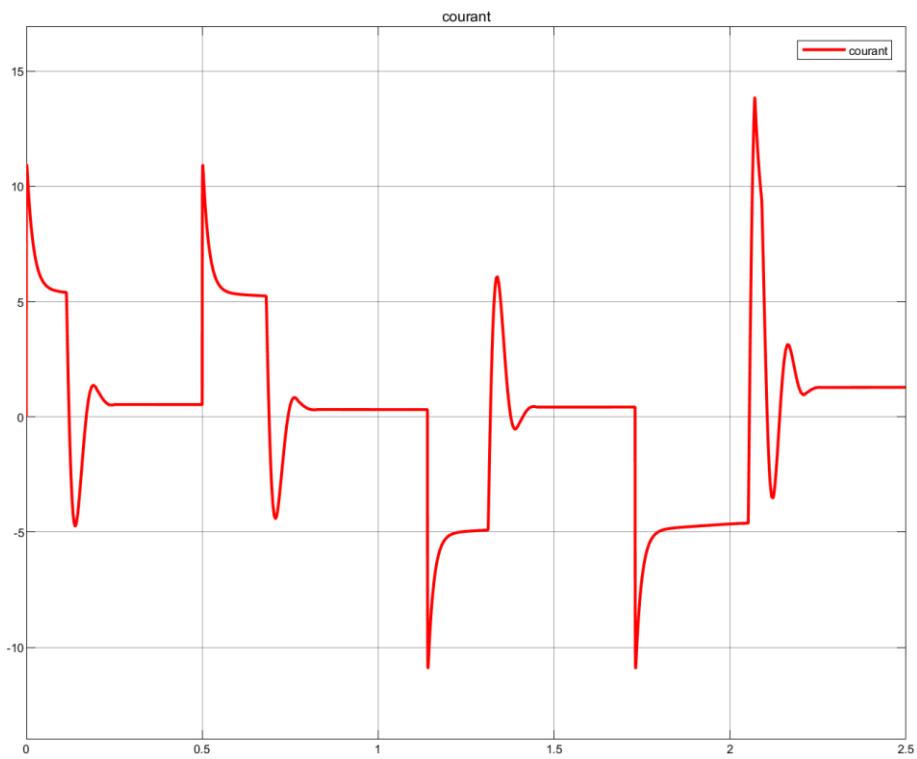


Figure 80 : visualisation du courant dans l'induit du moteur

Il est possible de modifier les paramètres de masse directement dans **Multibody** 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 81 : l'axe linéaire Control'X

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

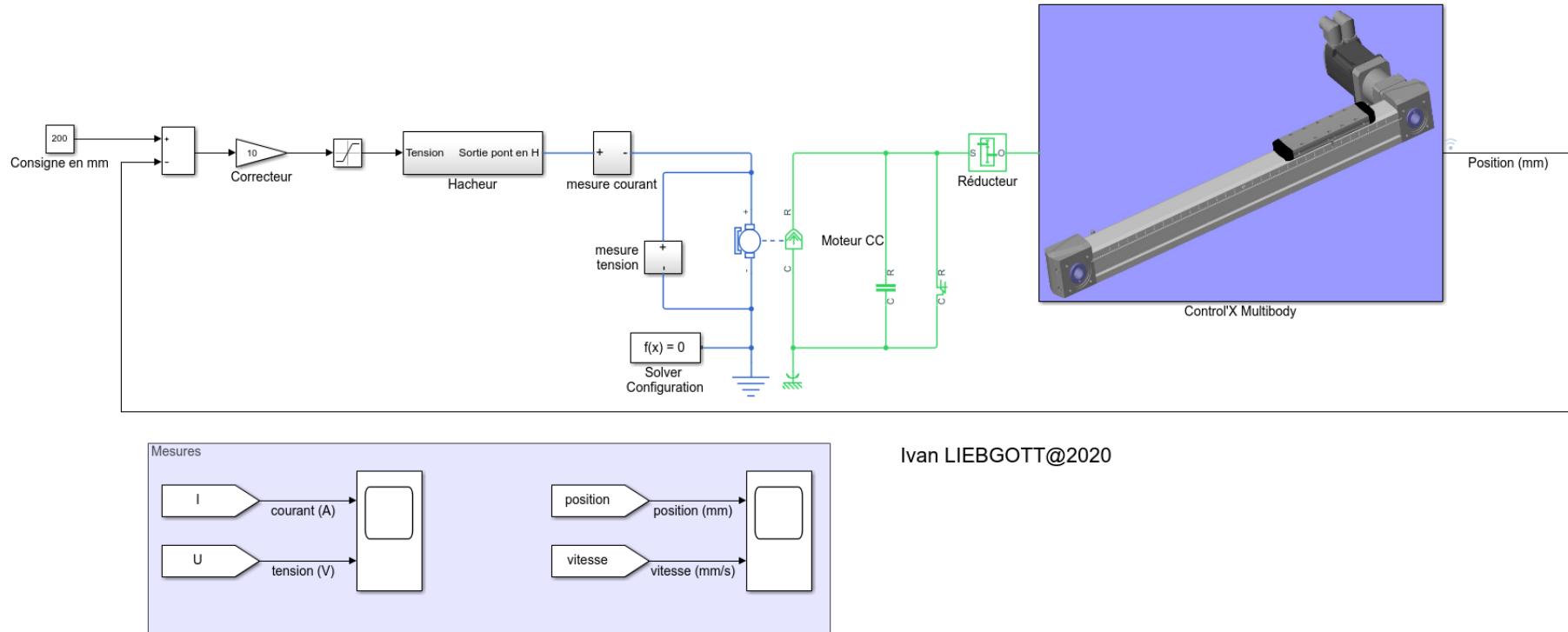


Figure 82 : modélisation multi-physique du système Control'X

Lancer la simulation et observer les résultats obtenus.

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

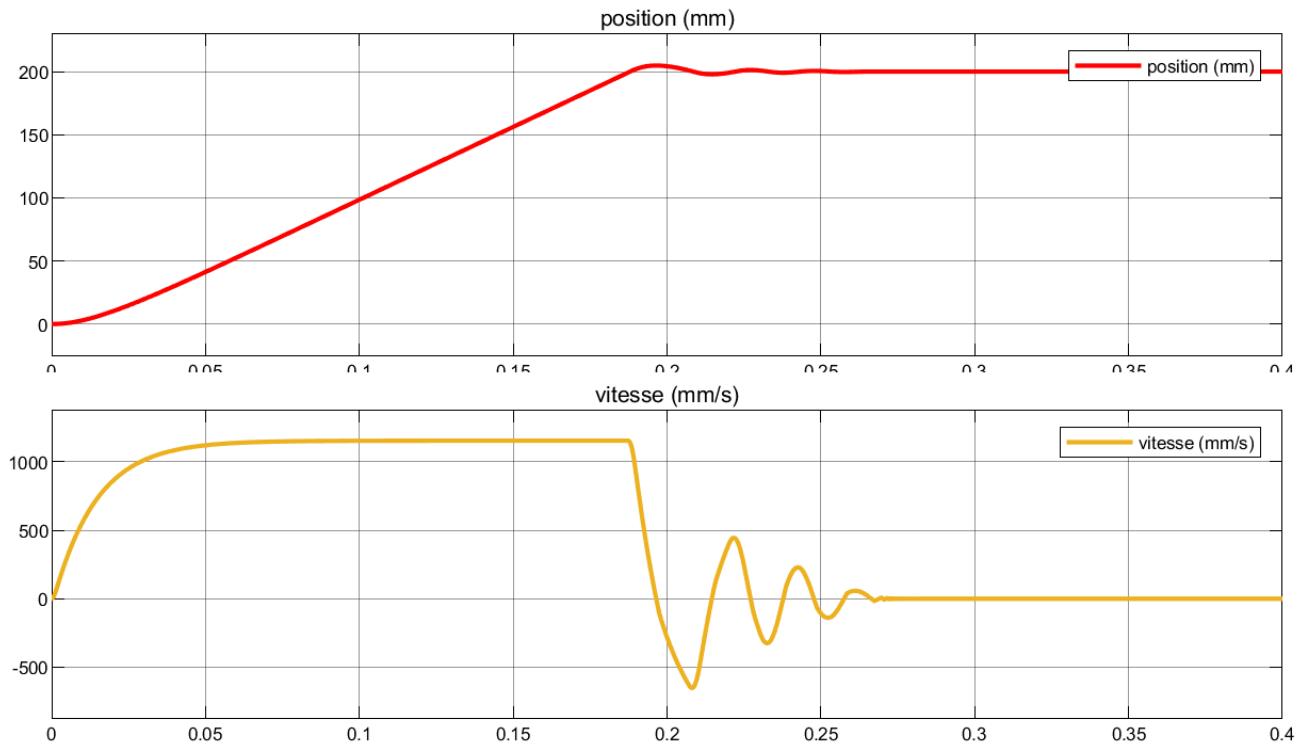


Figure 83 : visualisation de la vitesse et de la position de l'axe linéaire

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

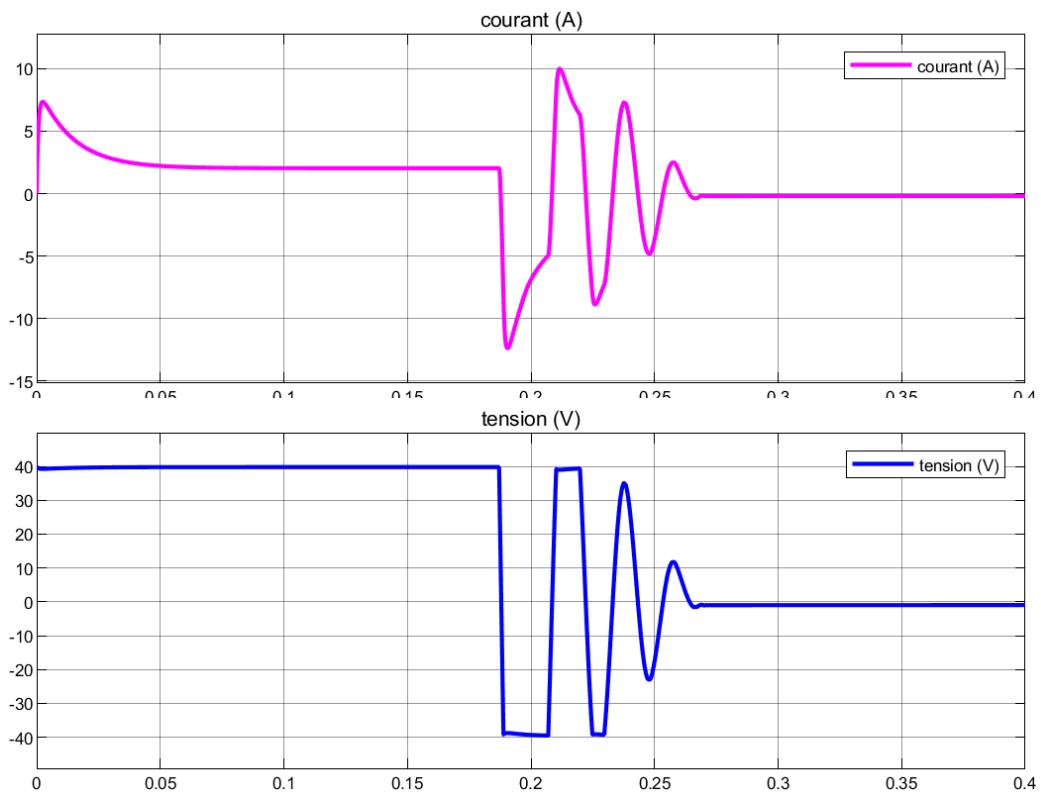


Figure 84 : visualisation de la tension de commande du moteur et du courant dans l'induit

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.
Lancer la simulation et observer les résultats obtenus.

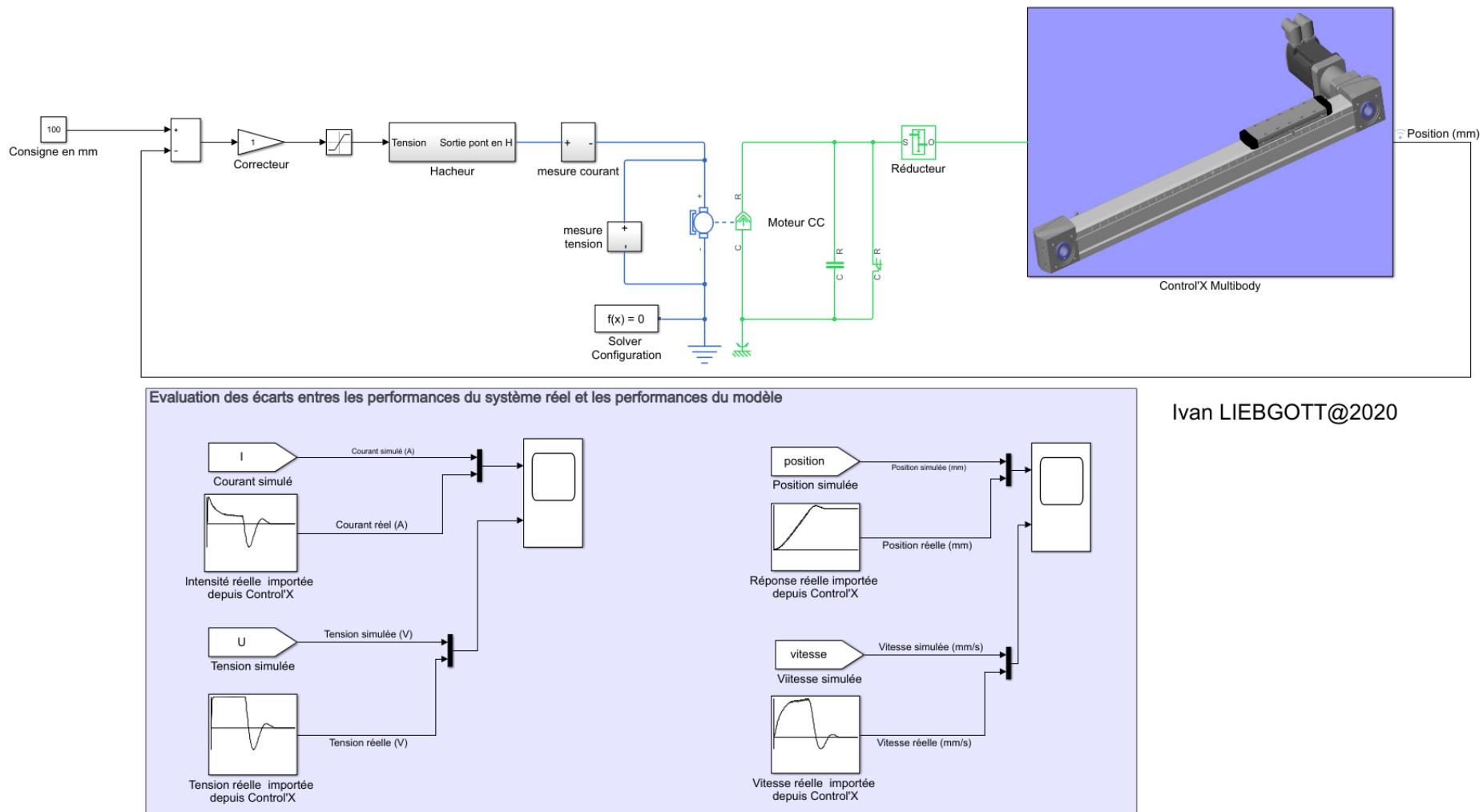


Figure 85 : évaluation des écarts entre les performances du modèle et celles de l'axe réel

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

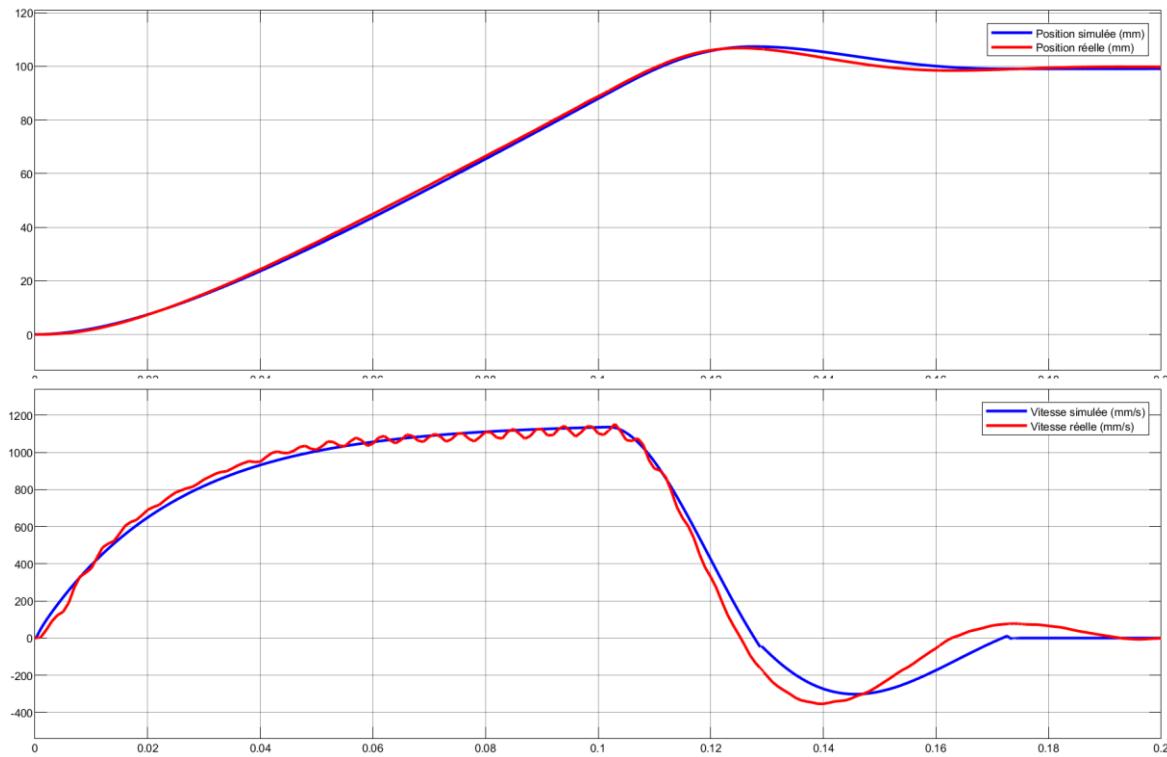


Figure 86 : évaluation des écarts réel/modèle pour la vitesse et la position de l'axe

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

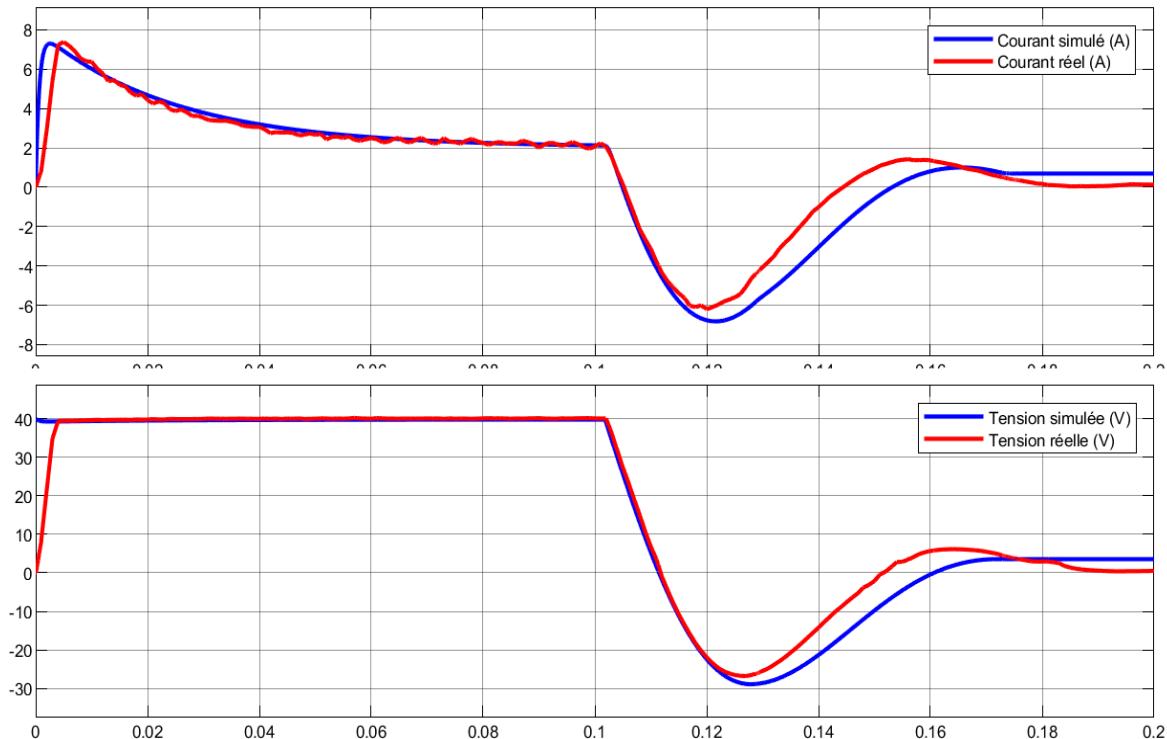


Figure 87 : évaluation des écarts réel/modèle pour le courant d'induit et la tension de commande

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-phérique avec MATLAB-Simulink ?

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

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

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

- Multibody
- Fluids
- Electrical
- Drivelines

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 **Multibody**. 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 Control 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 4 : 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.

À $t=0$ 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.

$$R=20 \Omega$$

$$L=0.01 \text{ mH}$$

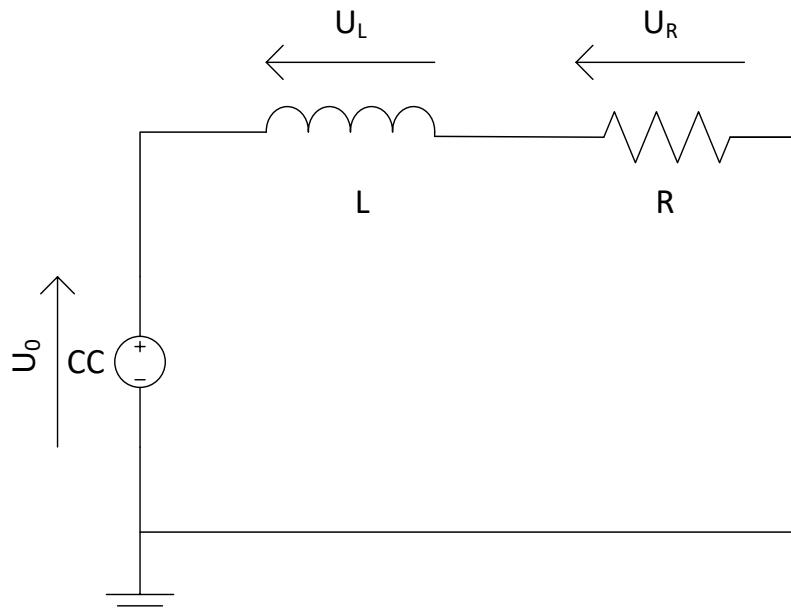


Figure 88 : circuit R-L

A. Choix des composants

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

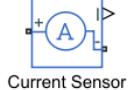
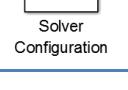
Fonction du composant	Représentation	Bibliothèque
Source de tension continue	 DC Voltage Source	Simscape/Foundation Library/Electrical/Electrical Sources
Résistance	 Resistor	Simscape/Foundation Library/Electrical/Electrical Element
Inductance	 Inductor	Simscape/Foundation Library/Electrical/Electrical Element
Référence électrique	 Electrical Reference	Simscape/Foundation Library/Electrical/Electrical Element
Capteur de courant	 Current Sensor	Simscape/Foundation Library/Electrical/Electrical Sensors
Capteur de tension	 Voltage Sensor	Simscape/Foundation Library/Electrical/Electrical Sensors
Conversion d'un signal physique en signal Simulink	 PS-Simulink Converter	Simscape/Utilities
Moniteur	 Scope	Simulink/Sinks
Solveur	 Solver Configuration	Simscape/Utilities

Figure 89 : les composants nécessaires à la modélisation du circuit R-L avec Simscape

B. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser** (bibliothèque de Simulink), 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 sur la Figure 90 ou ouvrir directement le fichier **circuit_RL_start.slx**. Vous pouvez vous reporter à la Figure 91 pour prendre connaissance des commandes utiles à l'orientation des composants.

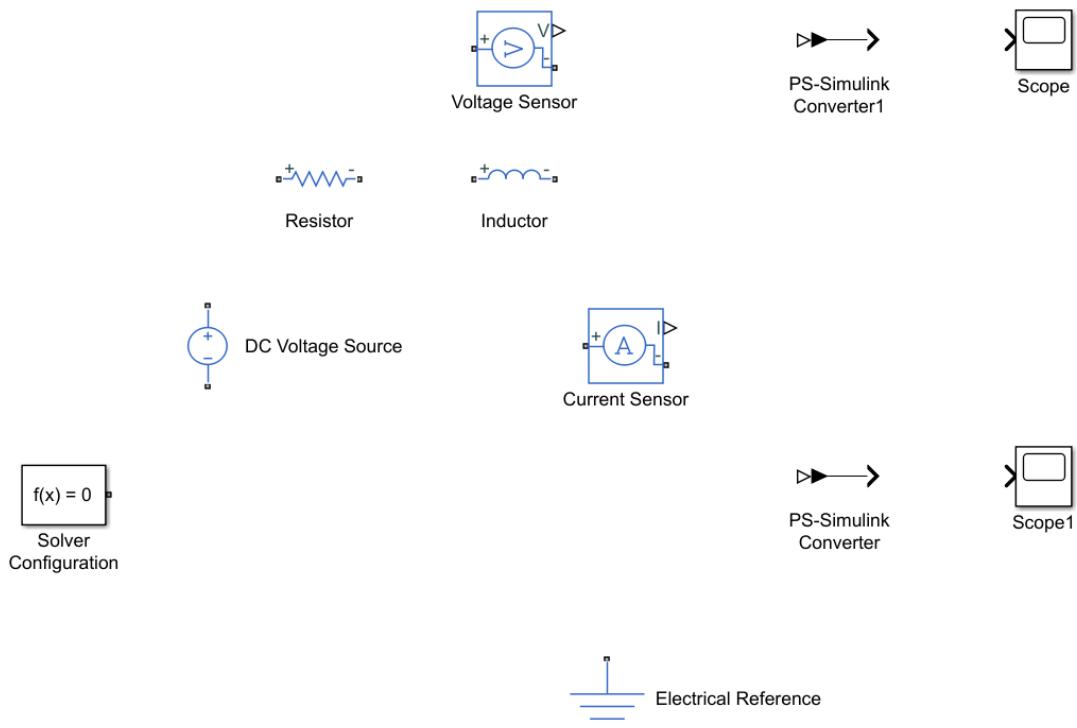


Figure 90 : composants nécessaires à la modélisation du circuit RL

Par défaut le nom des composants n'est pas visible. Pour afficher le nom de tous les composants, sélectionner tous les composant avec la souris (ou Ctrl+A). Cliquer ensuite sur un composant avec le bouton droit de la souris pour faire apparaître le menu contextuel et sélectionner **Format>Show Bloc Name/on**.

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 sélectionner Delete	Suppr

Figure 91 : commandes utiles

Relier les composants pour obtenir le modèle ci-dessous :

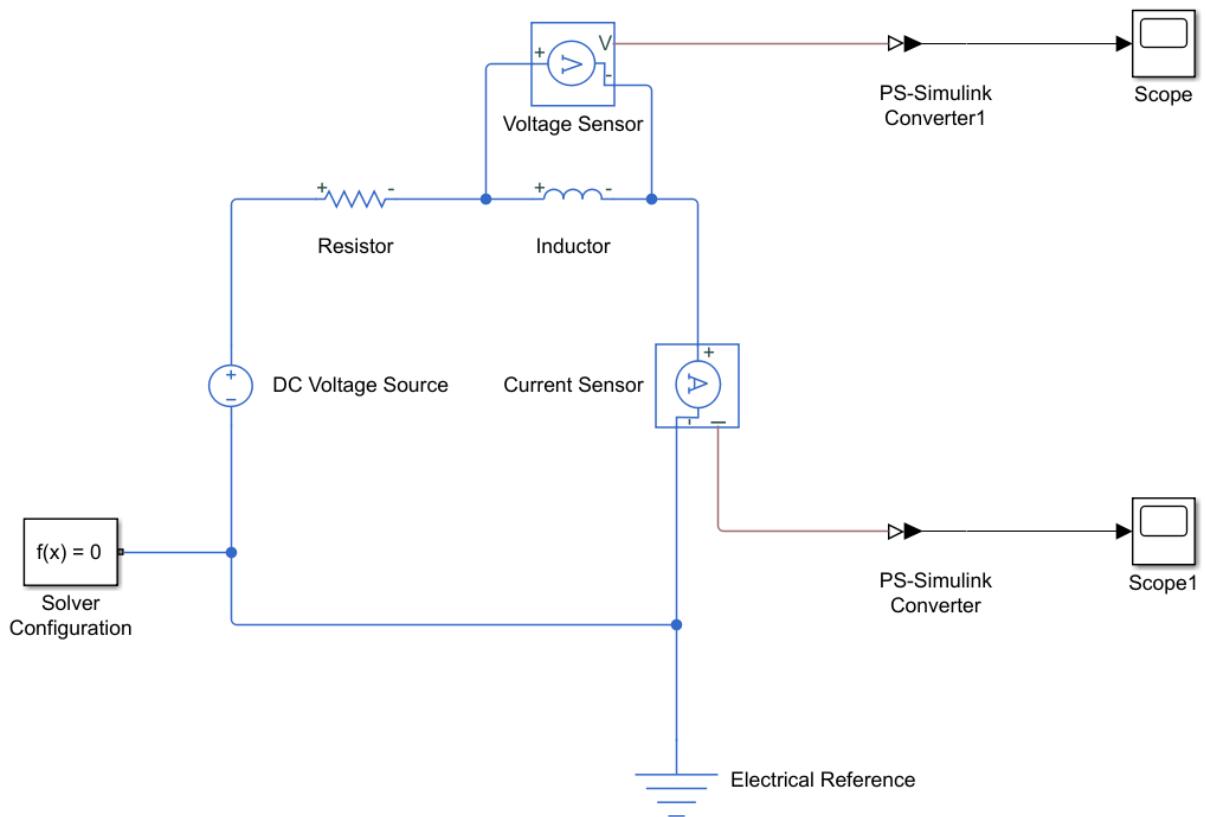


Figure 92 : Modèle Simscape du circuit R-L

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 93 : commandes utiles

C. Les différents types de ports et de connexions

Avant de paramétriser 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 94).

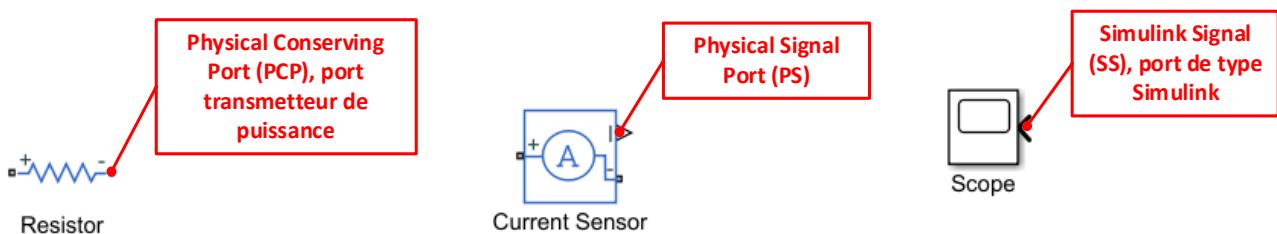


Figure 94 : les différents types de ports de Simscape

- 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 95).

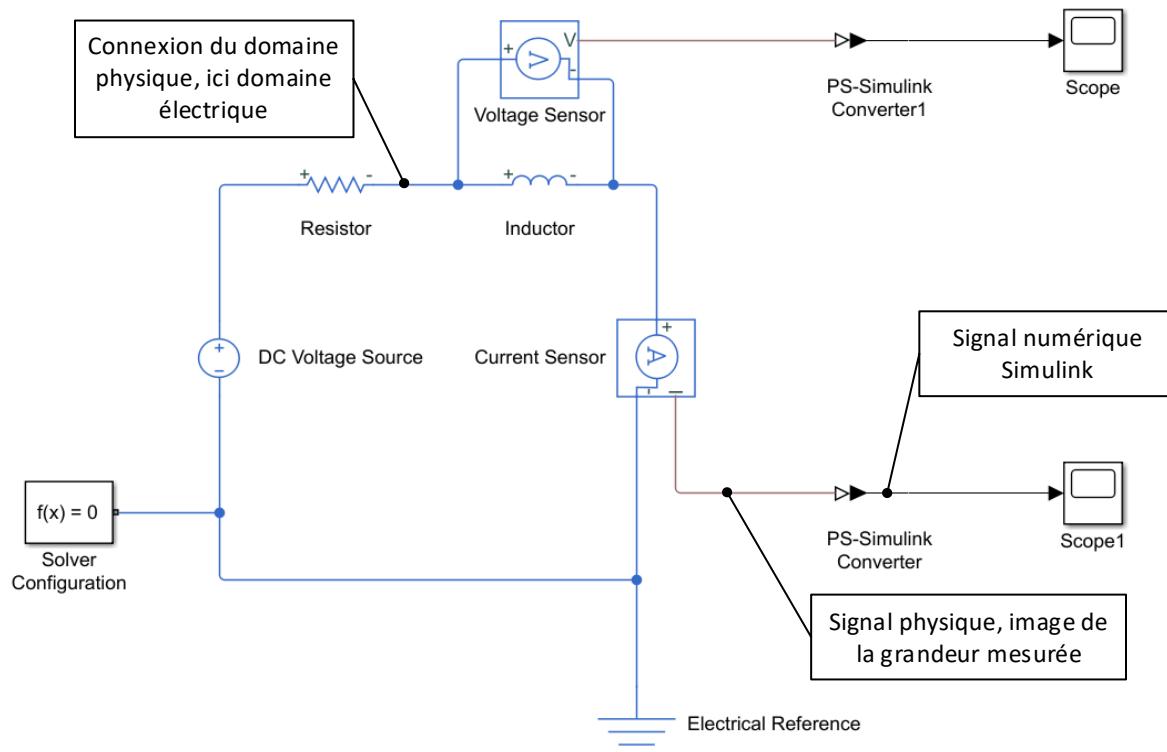


Figure 95 : identification des connexions d'un modèle Simscape

D. Paramétrage des composants

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

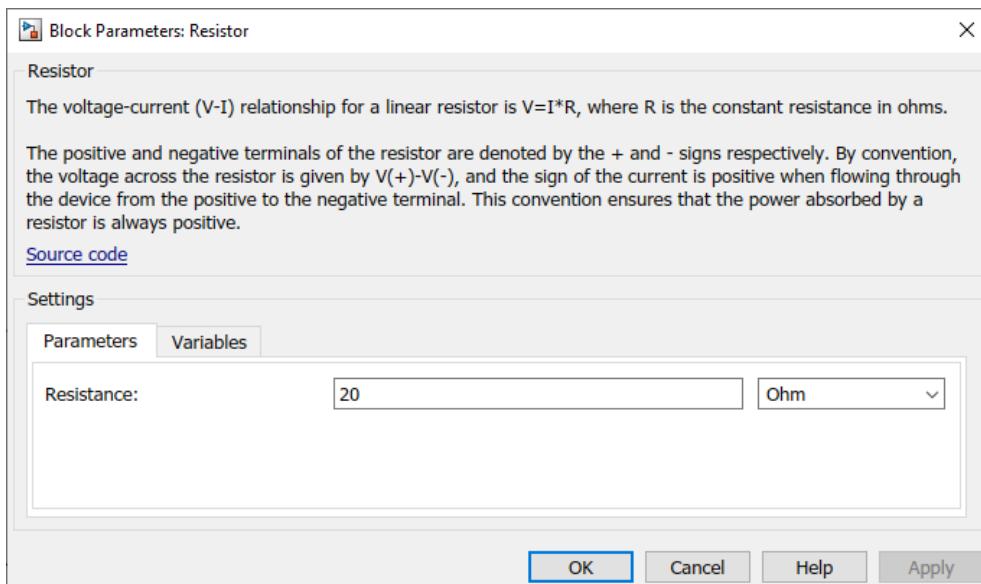
RESISTOR



Resistor

Simscape/Foundation Library/Electrical/Electrical Element

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



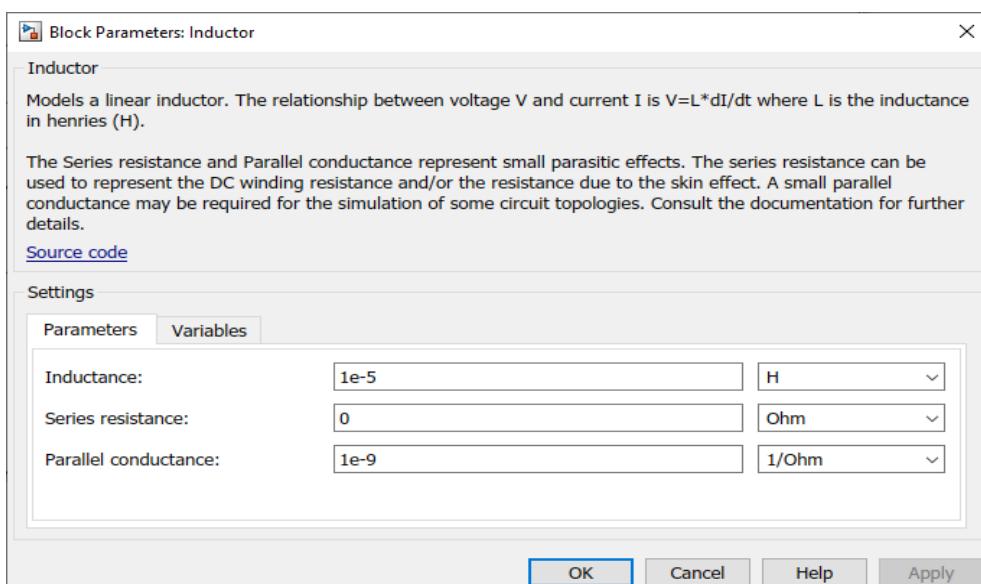
INDUCTOR



Inductor

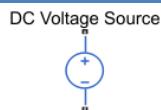
Simscape/Foundation Library/Electrical/Electrical Element

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



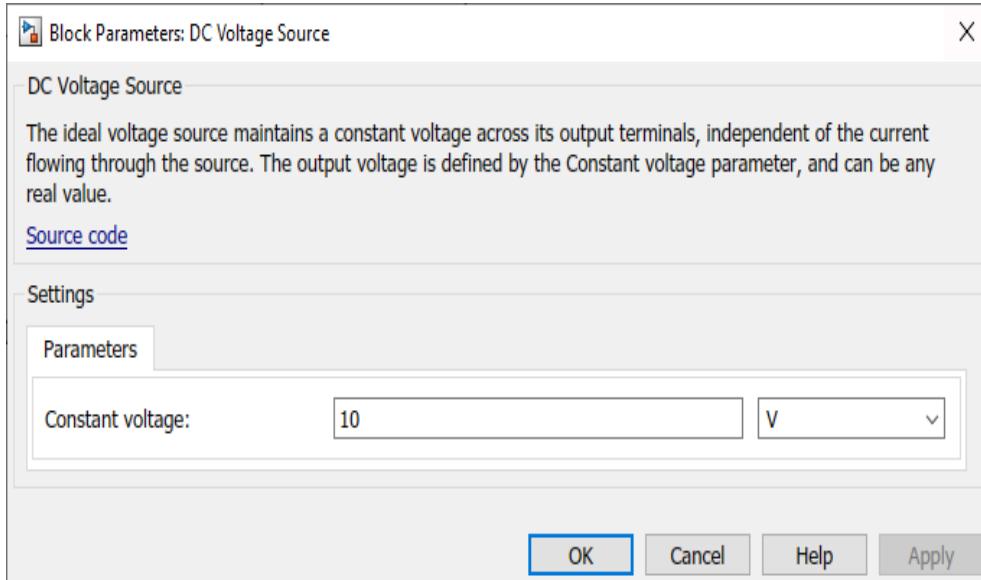
Paramétrage

DC VOLTAGE SOURCE



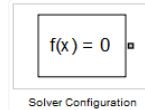
Simscape/Foundation Library/Electrical/Electrical Sources

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

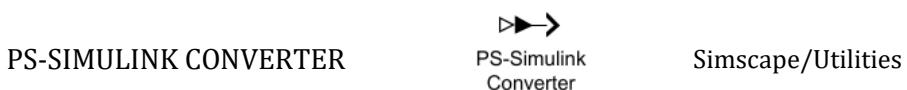
ELECTRICAL REFERENCE



Simscape/Foundation Library/Electrical/Electrical Element

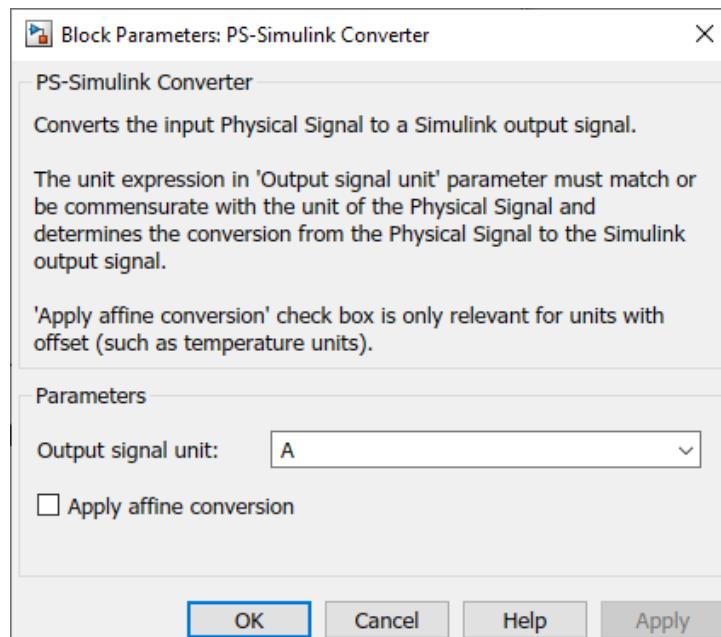
Aucun paramétrage n'est nécessaire pour ce composant. Un bloc référence du domaine physique utilisé doit obligatoirement être présent dans le modèle.

Paramétrage

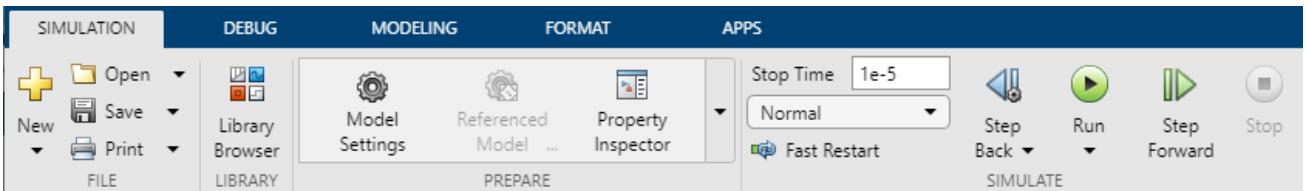


Ce composant permet de convertir un signal physique image d'une grandeur mesurée en signal **Simulink** afin de pouvoir être affiché dans un Scope. Il est souhaitable de préciser l'unité de la grandeur physique souhaitée pour l'affichage dans Simulink. Indiquer dans les deux blocs correspondants que 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 maîtriser 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 :

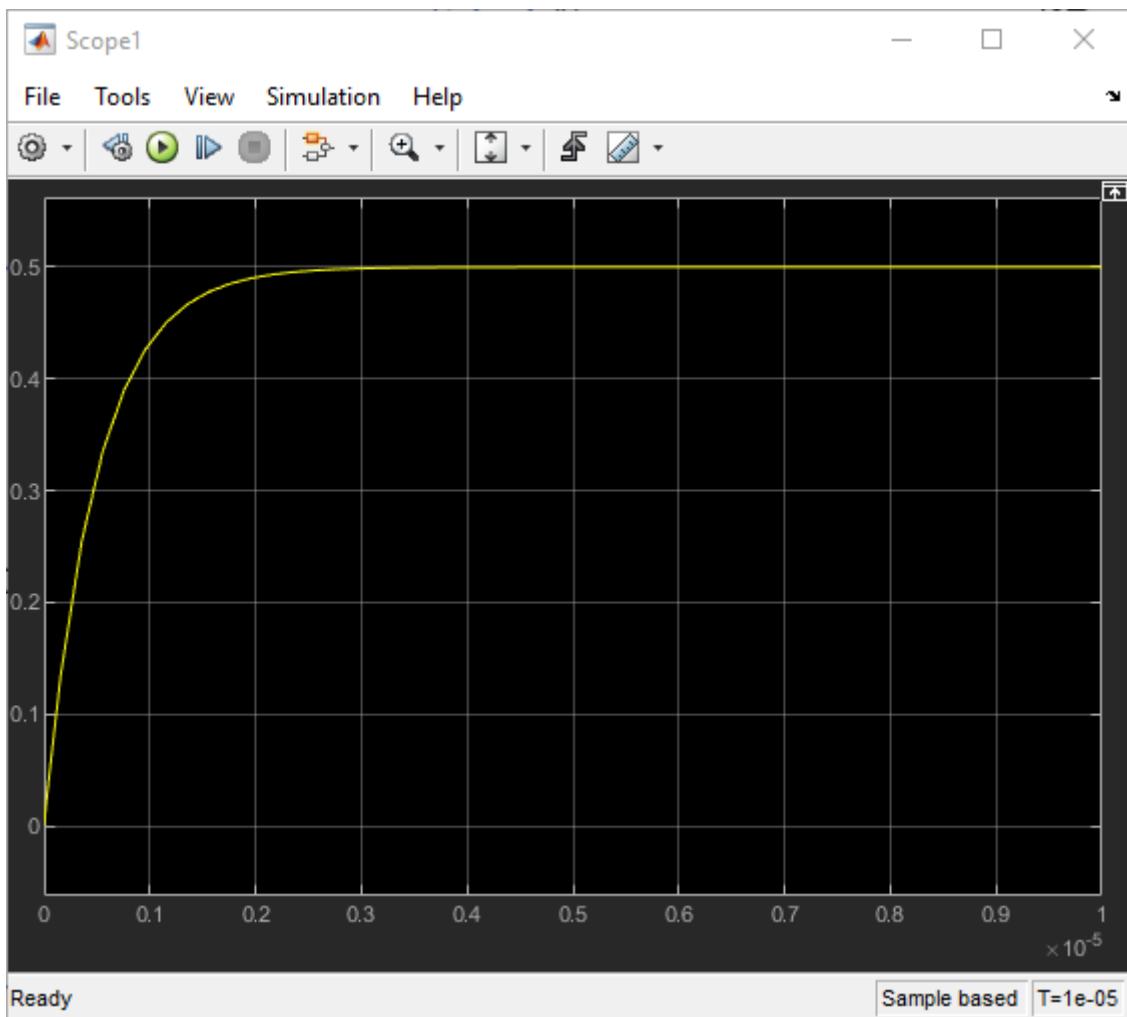


Figure 96 : évolution de l'intensité du courant dans le circuit RL

Les couleurs de mise en forme des scopes peuvent être modifiées. Cliquer avec le bouton droit de la souris sur l'icône dans le scope pour faire apparaître le menu déroulant de la Figure 97.

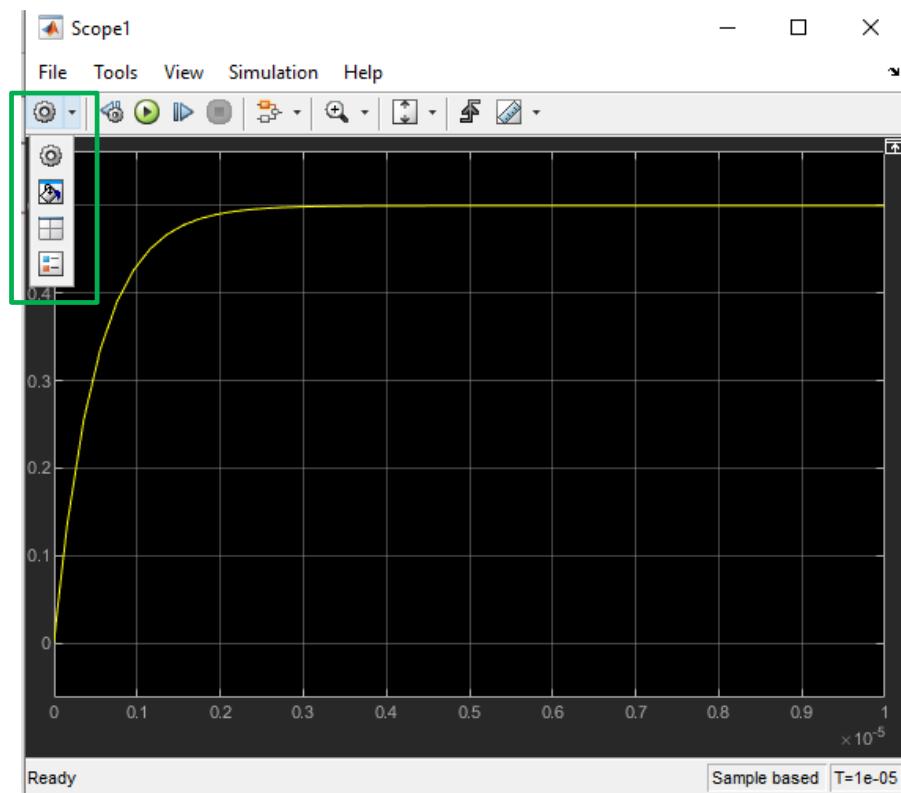


Figure 97 : mise en forme des scopes



Sélectionner l'icône et paramétrer la mise en forme conformément à la Figure 98.

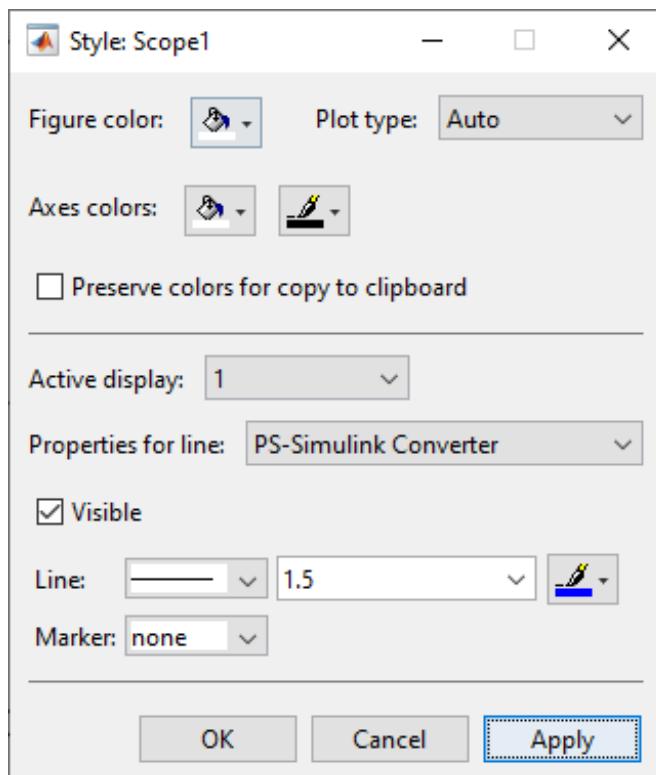


Figure 98 : modification de la mise en forme d'un scope

Vous obtenez alors le résultat de la Figure 99.

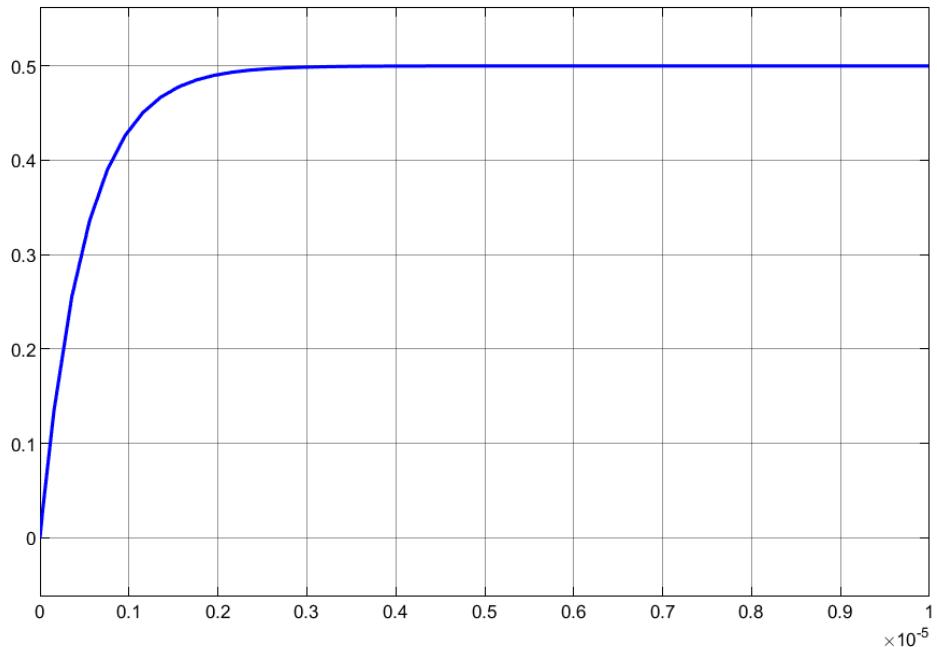


Figure 99 : modification de l'aspect des courbes d'un scope

Il est possible de choisir les options de mise en forme plus avancées. Pour cela vous pouvez vous reporter à l'annexe « Paramétrage des scopes » en fin d'ouvrage pour la mise en forme des courbes.

Nous procémons 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 :

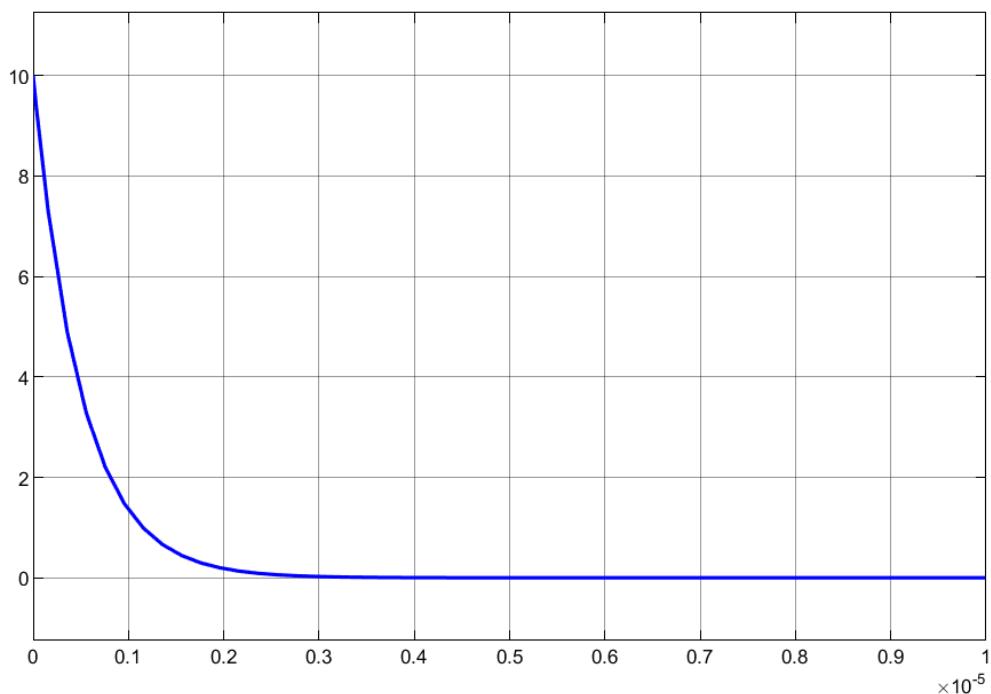


Figure 100 : évolution de la tension aux bornes de la bobine pour le circuit RL

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.

La courbe suivante représente l'évolution du courant avec une valeur de l'inductance de $5 \cdot 10^{-5}$ H.

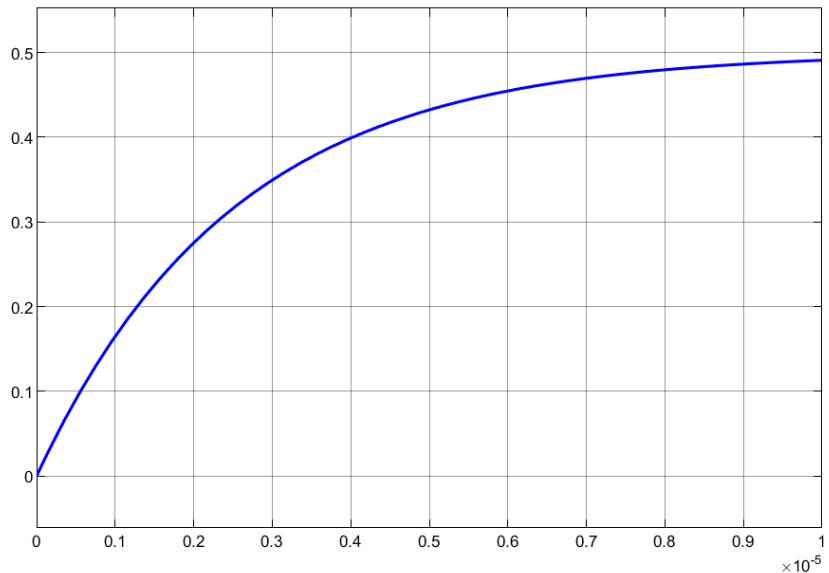


Figure 101 : évolution du courant avec une autre valeur de l'inductance

Nous constatons 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.

F. Utilisation du Data Inspector

Le **Data Inspector** est un outil qui permet de garder en mémoire le résultat de plusieurs simulations et de comparer les résultats. Il est très simple à utiliser et nous allons voir dans cet exemple une partie de ses fonctionnalités. Pour bien utiliser cet outil, il est nécessaire de réaliser quelques opérations sur le modèle.

Dans un premier temps, il est conseillé de nommer les signaux qui arrivent dans les scopes afin de pouvoir mieux les identifier par la suite. Pour cela, il suffit de double cliquer directement sur la connexion Simulink qui arrive dans le scope et de taper le nom du signal. Pour revenir à la ligne, vous devez utiliser la combinaison de touche Maj+Entrée.

Renommer les signaux comme indiqué sur la Figure 102.

Paramétrier la valeur de l'inductance à $1 \cdot 10^{-5}$ H comme initialement.

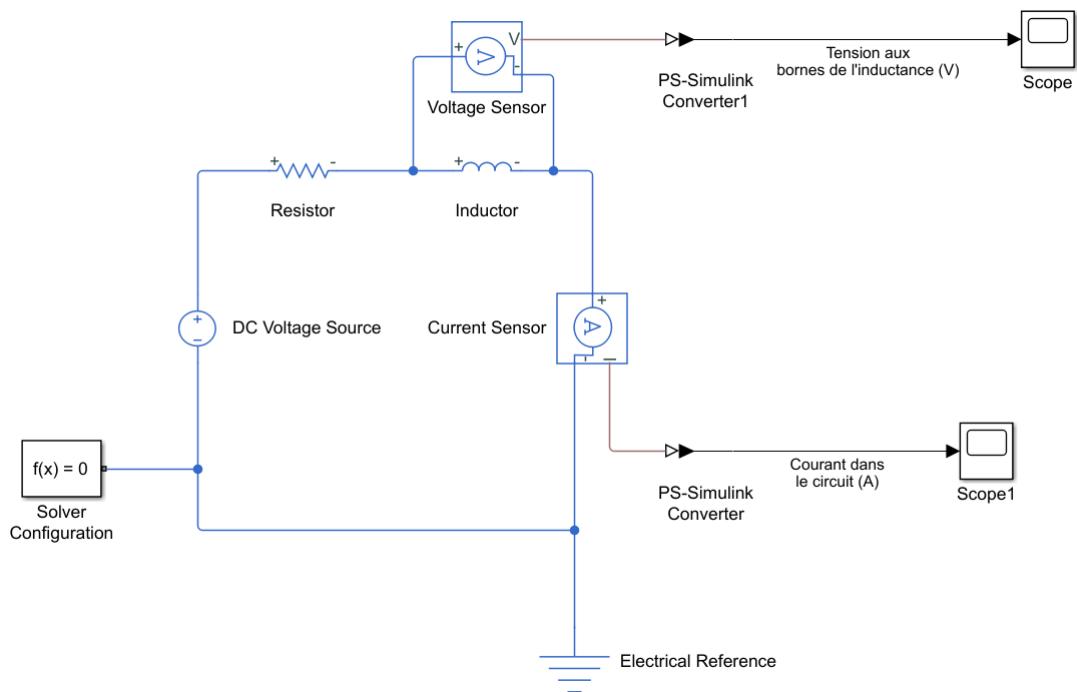


Figure 102 : définition du nom des signaux à analyser

Il faut maintenant indiquer au modèle les signaux que l'on veut visualiser à l'aide du **Data Inspector**. Dans notre exemple nous suivrons les signaux « Tension aux bornes de l'inductance » et « Courant dans le circuit ».

Ouvrir le menu contextuel de la connexion correspondant au signal « Tension aux bornes de l'inductance » en cliquant avec le bouton droit de la souris sur cette connexion et sélectionner **Log Selected Signals** dans ce menu (Figure 103).

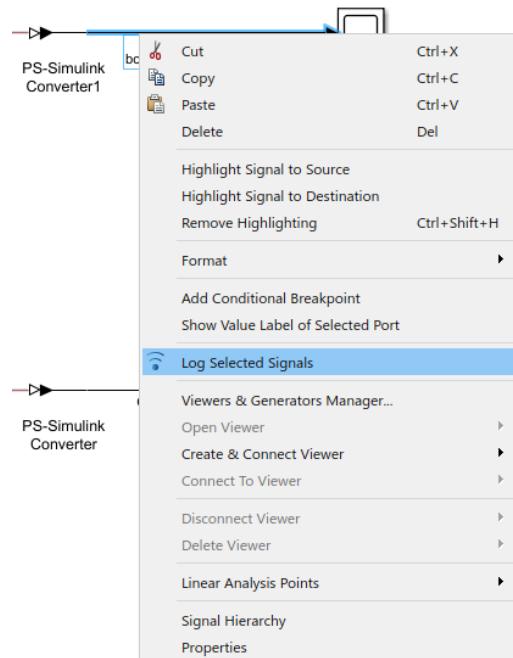


Figure 103 : sélection des signaux à suivre avec le Data Inspector

Reproduire la procédure avec le signal « Courant dans le circuit ». Vous devez obtenir le modèle de la Figure 104.

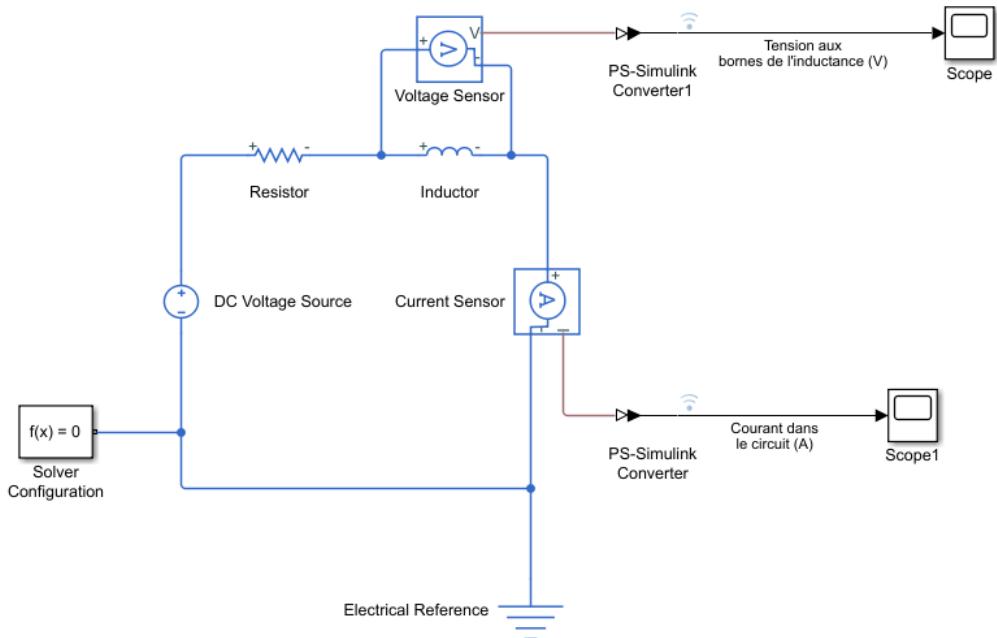


Figure 104 : visualisation des signaux à suivre avec le Data Inspector



Lancer la simulation et ouvrir le Data Inspector en cliquant sur l'icône dans la fenêtre de commande de Simulink. Fermer la fenêtre d'aide qui va s'ouvrir automatiquement. Vous devez voir la fenêtre du Data Inspector

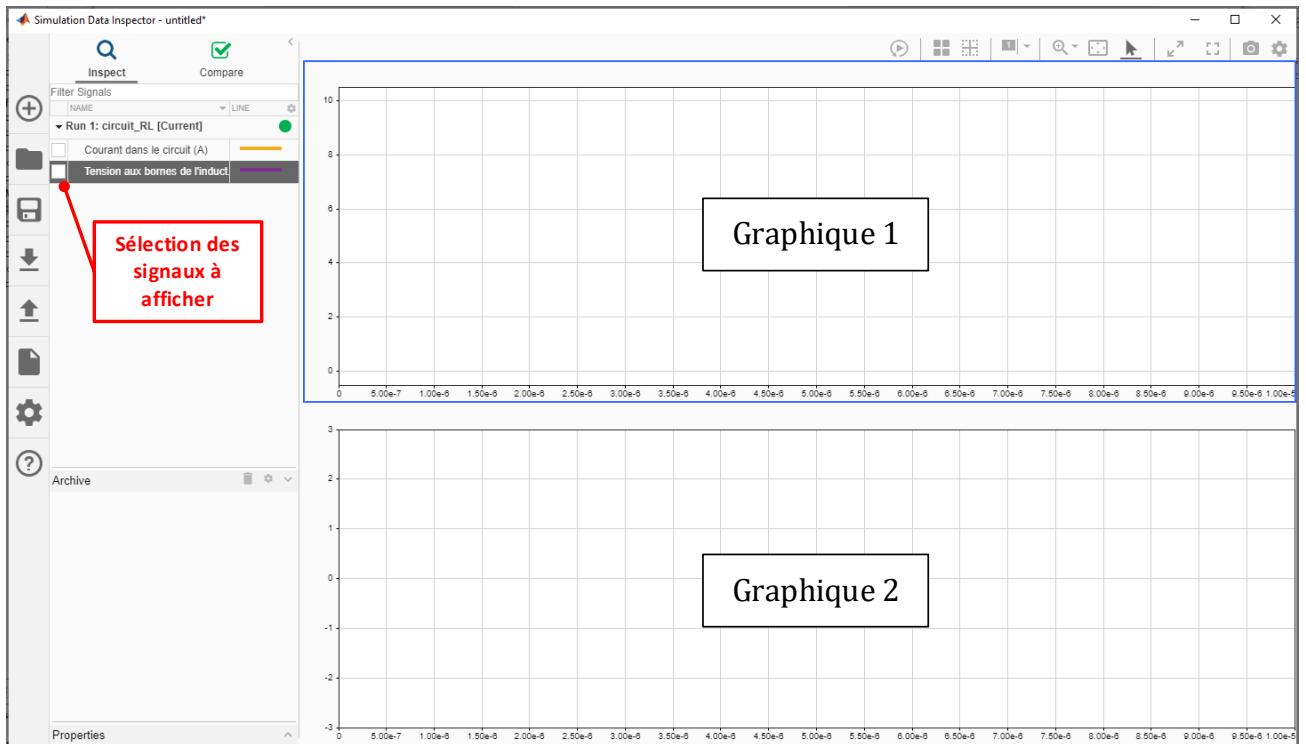


Figure 105 : fenêtre du Data Inspector

La fenêtre du Data Inspector dispose de deux fenêtres graphiques que vous pouvez sélectionner en cliquant dessus. Sur la Figure 105, la fenêtre « Graphique 1 » est sélectionnée (contour bleu). Afin de visualiser la courbe « Courant dans le circuit » dans la fenêtre « Graphique 1 », il suffit de cocher la case « Courant dans le circuit » (Figure 106).

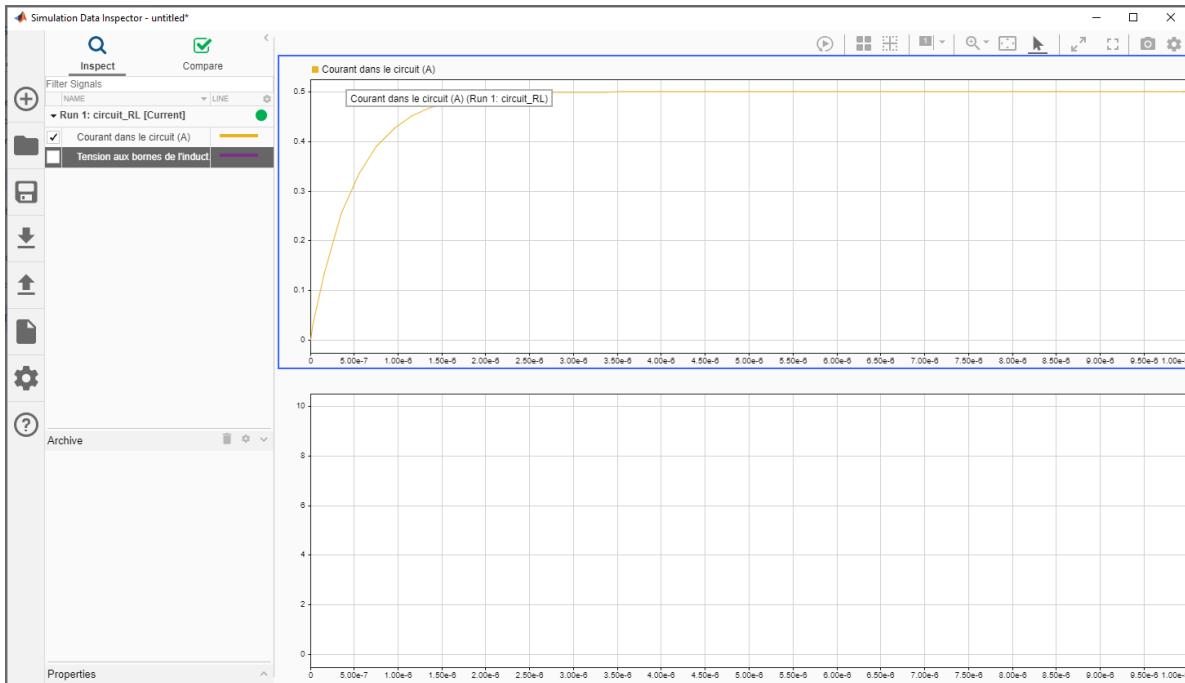


Figure 106 : visualisation d'une courbe à l'aide du Data Inspector

Sélectionner maintenant la fenêtre « Graphique 2 » et afficher le signal « Tension aux bornes de l'inductance » dans cette fenêtre (Figure 107).

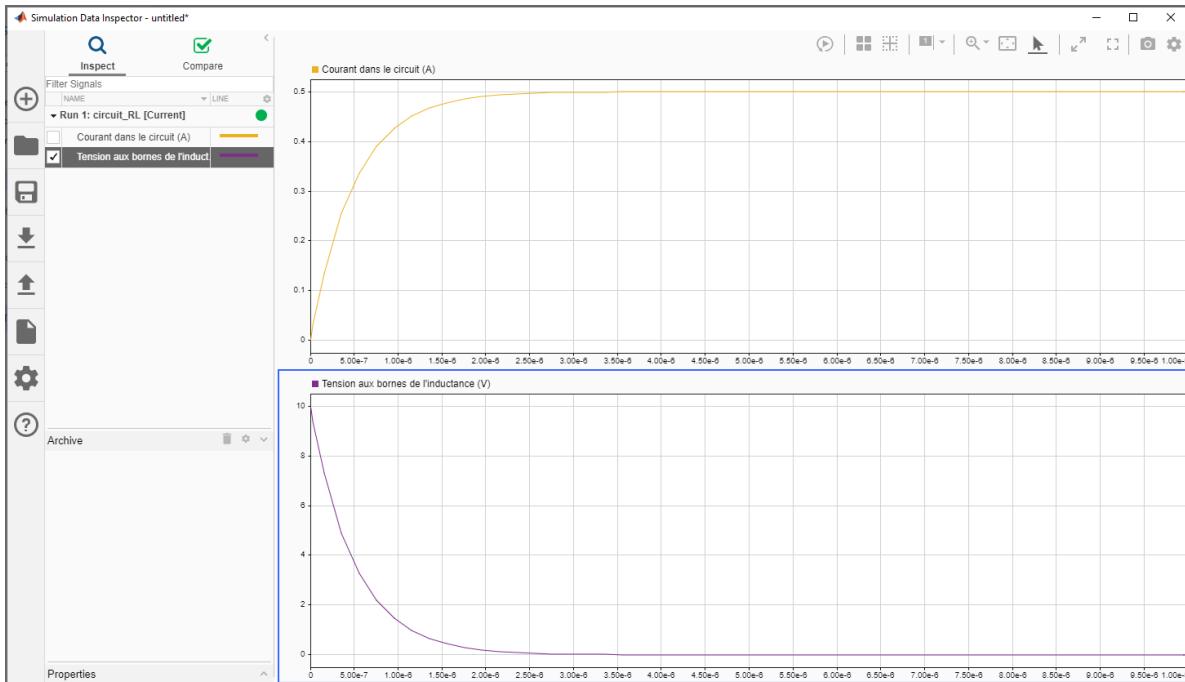


Figure 107 : visualisation de plusieurs courbes à l'aide du Data Inspector

Revenir dans le modèle et **paramétriser** la valeur de l'inductance à 5.10^{-5} H.

Lancer la simulation et revenir dans la fenêtre du **Data Inspector**. Le dernier Run de simulation correspondant à la dernière valeur de l'inductance s'affiche par défaut dans la fenêtre graphique (Figure 108). Le Data Inspector enregistrera les signaux marqués à chaque nouvelle simulation.

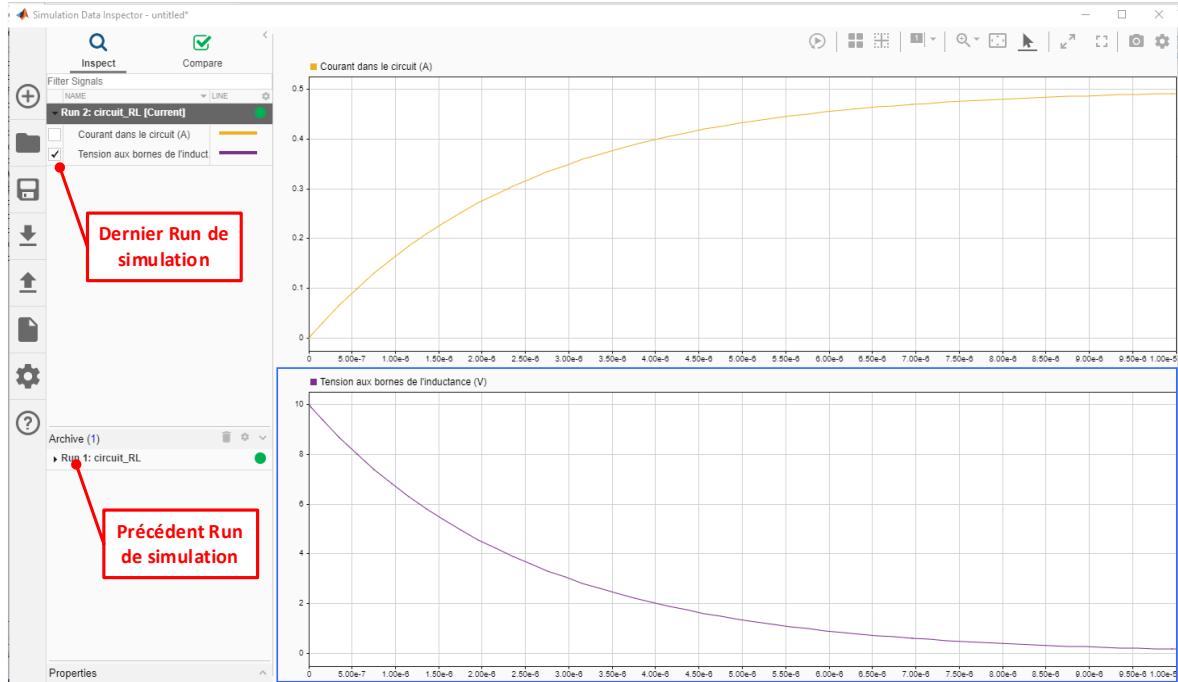


Figure 108 : visualisation des résultats de simulations différentes à l'aide du Data Inspector

Pour visualiser les courbes des deux simulations sur un même fenêtre graphique, il faut sélectionner une fenêtre graphique et cocher toutes les cases qui correspondent aux signaux que l'on veut afficher. La Figure 109 permet de comparer les courants et les tensions pour les deux simulations.

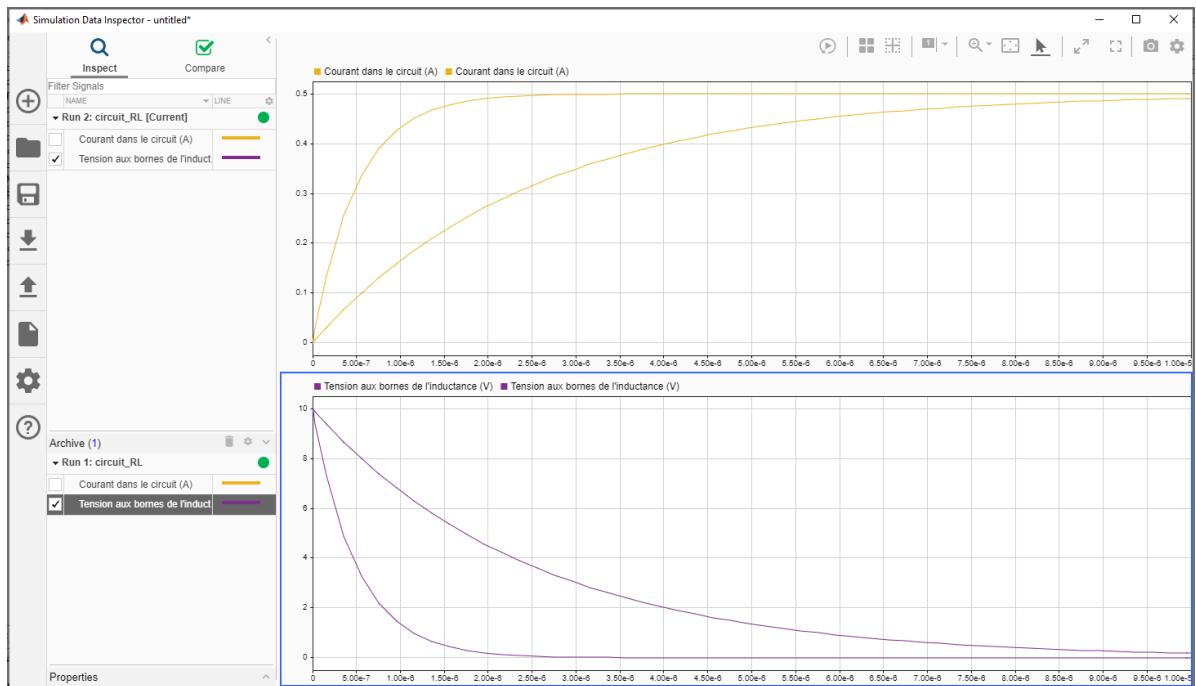


Figure 109 : affichage de plusieurs courbes de résultats à l'aide du Data Inspector

Il est possible de changer la couleur des courbes en cliquant sur la ligne de couleur à droite du signal (Figure 110 et Figure 111).

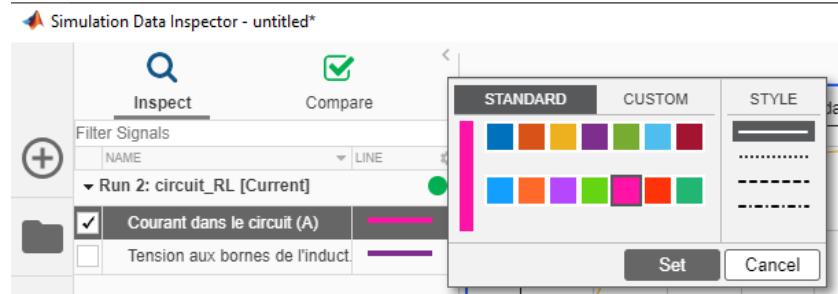


Figure 110 : modification de la couleur des courbes à l'aide du Data Inspector

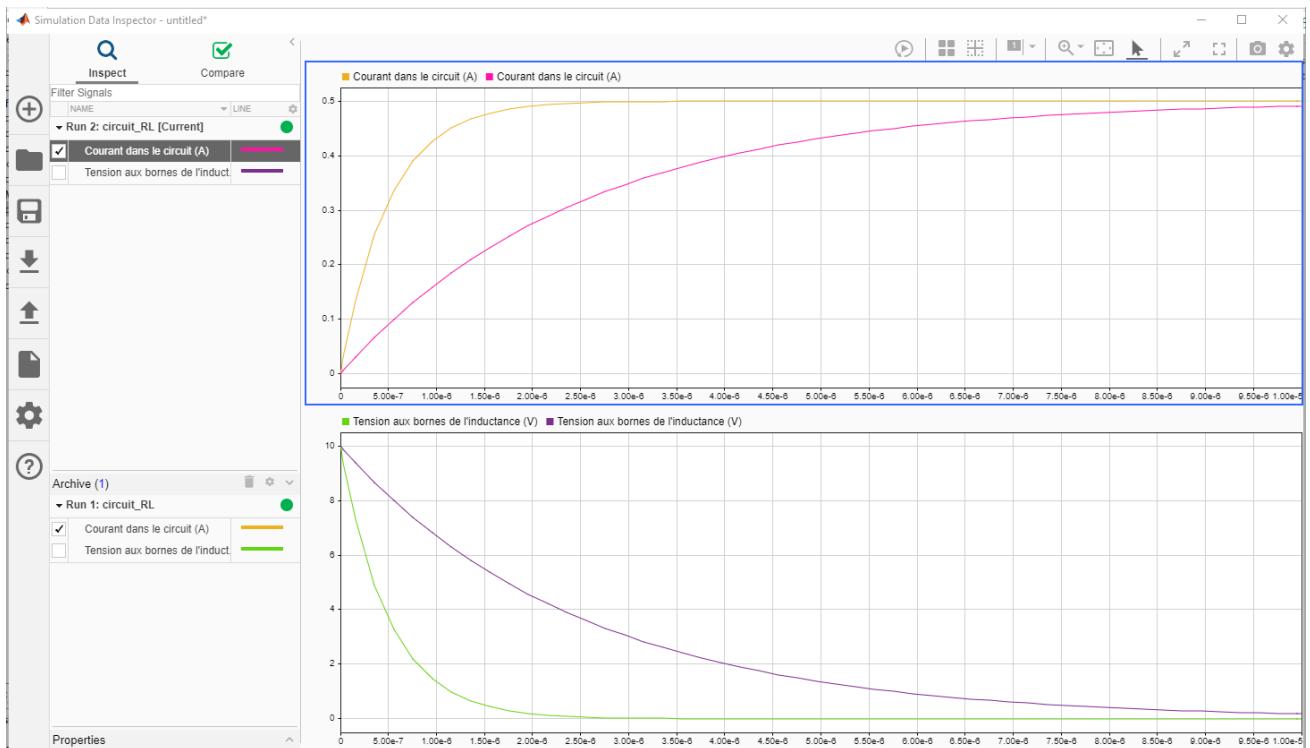


Figure 111 : visualisation de la modification de la couleur des courbes à l'aide du Data Inspector

L'une des fonctionnalités intéressantes du Data Inspector est la comparaison automatique des courbes.



Dans la partie supérieure de la fenêtre du Data Inspector cliquer sur **Compare**, puis à l'aide des menus déroulants, indiquer que vous voulez comparer les résultats du Run1 avec les résultats du Run2 (Figure 112) et cliquer sur **Compare**.



Figure 112 : sélection des simulations à comparer à l'aide du Data Inspector

Vous devez obtenir le résultat de la Figure 113.

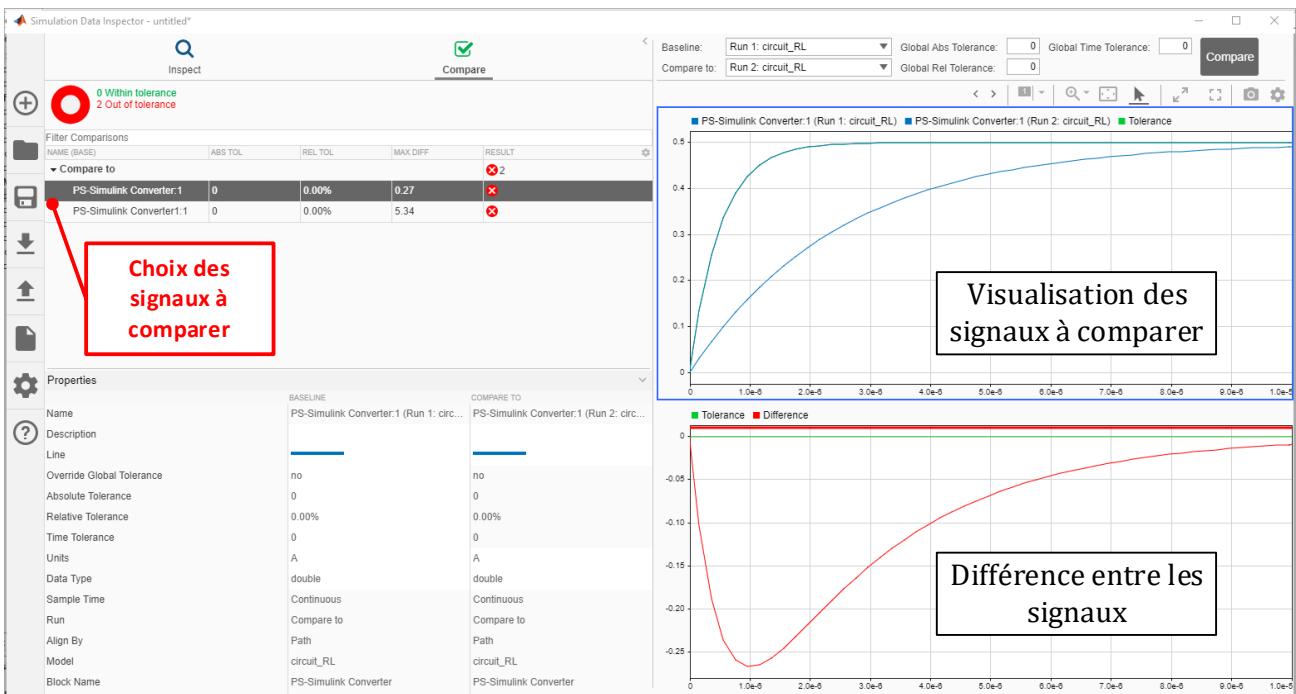


Figure 113 : visualisation de la comparaison entre deux simulations à l'aide du Data Inspector

Il est possible de comparer les signaux correspondant à la tension ou au courant en sélectionnant le signal dans la partie gauche de la fenêtre (Figure 113).

Il est possible de sauvegarder les résultats en cliquant sur dans la partie gauche de la fenêtre pour les consulter ultérieurement.

Pour utiliser plus en profondeur les fonctionnalités du Data Inspector, vous pouvez consulter l'aide en cliquant sur .

II. Comparaison avec l'approche causale

A. Equation de comportement du système

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

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

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

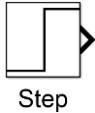
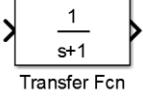
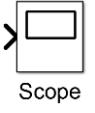
Fonction du composant	Représentation	Bibliothèque
Source échelon	 Step	Simulink/Sources
Fonction de transfert	 Transfer Fcn	Simulink/Continuous
Moniteur	 Scope	Simulink/Sinks

Figure 114 : les composants nécessaires à la modélisation du circuit R-L avec Simulink

C. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, cliquer sur  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 116 pour prendre connaissances des commandes utiles pour relier et ordonner les blocs dans **Simulink**.

Le modèle obtenu est le suivant :

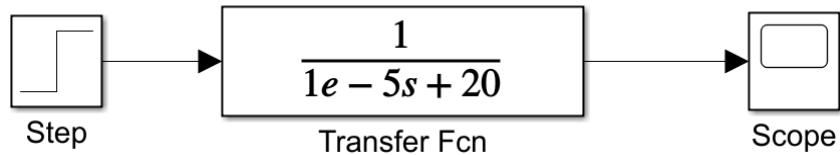


Figure 115 : modèle du circuit R-L réalisé avec Simulink

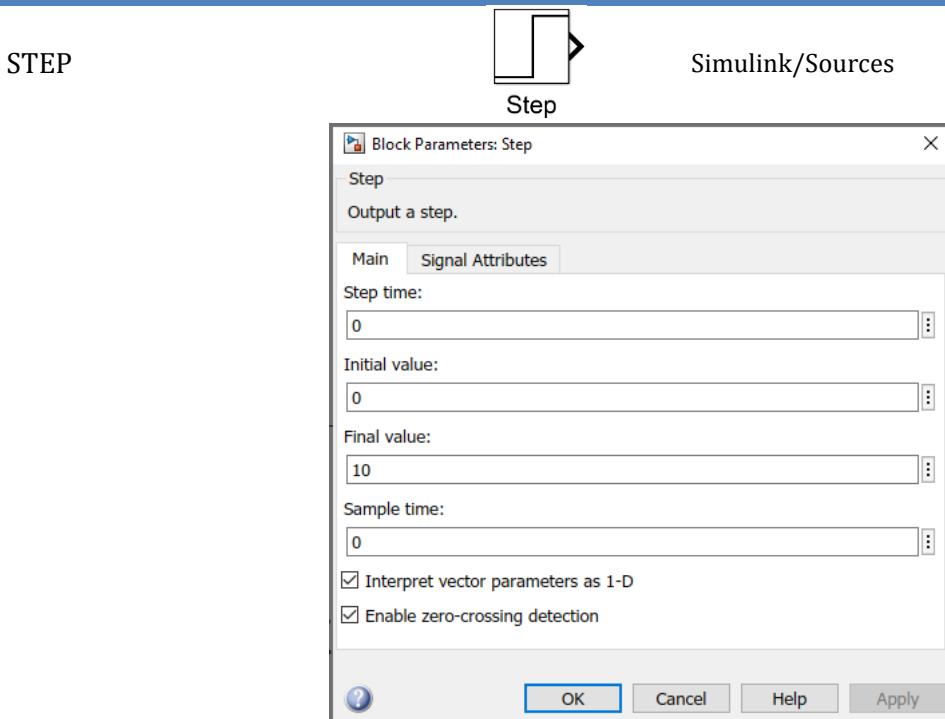
Commandes utiles

Fonctions	Actions
Connecter deux composants	Sélectionner le premier composant. En maintenant la touche contrôle enfoncee, 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 116 : Commandes utiles

D. Paramétrage des composants

Paramétrage



Step time : instant à partir duquel l'échelon passe à la valeur initiale, ici à $t=0$ s.

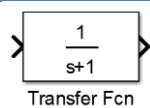
Initial value : valeur initiale de l'échelon, ici 0

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.

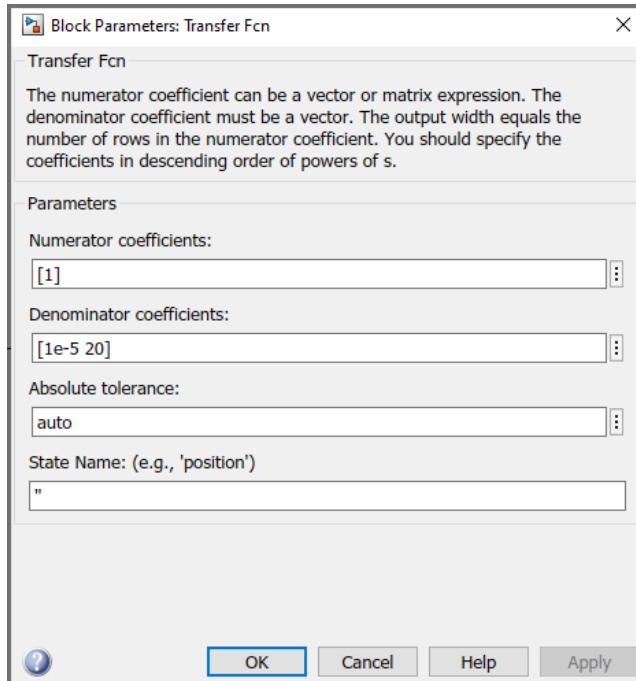
Paramétrage

TRANSFERT FUNCTION



Simulink/Continuous

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



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.

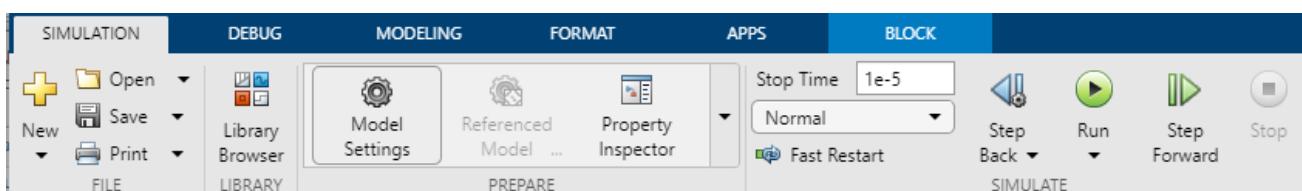
Exemple :

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

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 

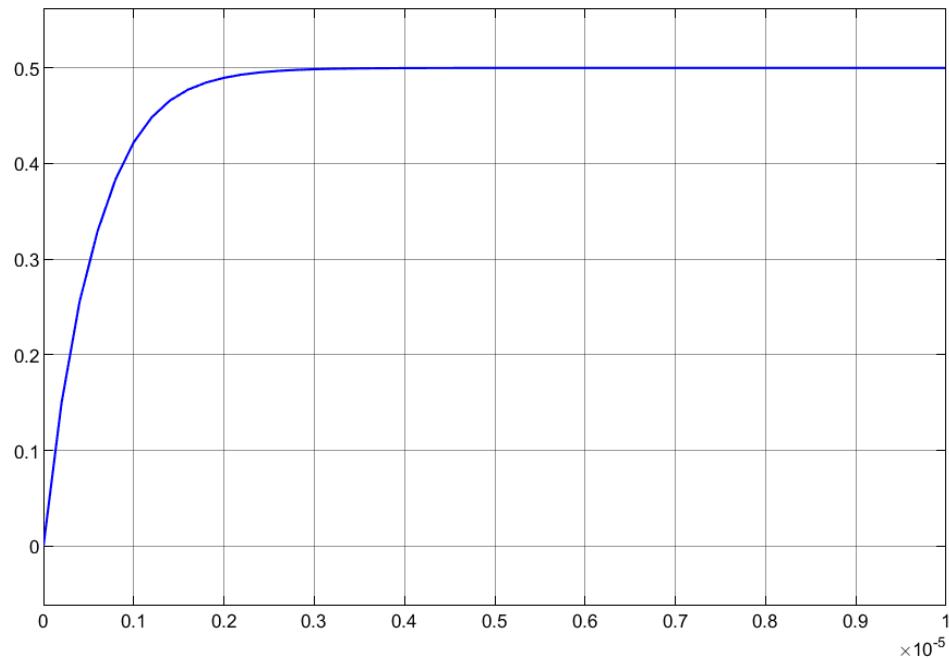


Figure 117 : évolution de l'intensité du courant dans le circuit RL

La courbe donnant l'évolution du courant dans le circuit est la même que pour le modèle acausal (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	<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • Parfaite maîtrise du niveau de modélisation • Structure du modèle plus adaptée à la mise en place d'une démarche de contrôle commande • Maîtrise de l'introduction des non linéarités
Inconvénients	<ul style="list-style-type: none"> • Peu de maîtrise 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 	<ul style="list-style-type: none"> • 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 118 : 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 119 présente quelques exemples de composants et détaille les types de ports associés.

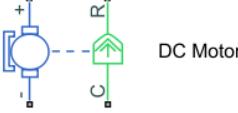
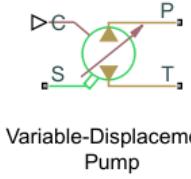
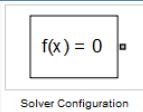
Fonction du composant	Représentation	Types de ports
Résistance	 Resistor	Ce composant possède 2 ports de type PCP du domaine électrique . <u>Paramètre</u> : résistance
Réducteur	 Gear Box	Ce composant possède 2 ports de type PCP du domaine mécanique de rotation . <u>Paramètre</u> : rapport de réduction
Conversion mécanique rotation/translation	 Wheel and Axle	Ce composant possède 2 ports de type PCP . Port A du domaine mécanique de rotation et Port B du domaine mécanique de translation . <u>Paramètre</u> : rayon de la roue
Conversion de puissance électrique en puissance mécanique	 DC Motor	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 . <u>Paramètre</u> : k_t en N.m/A
Source de débit hydraulique	 Hydraulic Constant Mass Flow Rate Source	Ce composant possède 2 ports de type PCP du domaine hydraulique . <u>Paramètre</u> : débit
Conversion de puissance mécanique en puissance hydraulique	 Variable-Displacement Pump	Ce composant possède 3 ports de type PCP . 1 port (S) du domaine mécanique et 2 ports (T et P) du domaine hydraulique . Le port mécanique (S) transmet un couple qui fait tourner le bâillet de la pompe générant un transfert de fluide du port T au port P. <u>Paramètre</u> : cylindrée de la pompe (et d'autres paramètres annexes caractérisant le rendement de la conversion de puissance)

Figure 119 : 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 s'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.

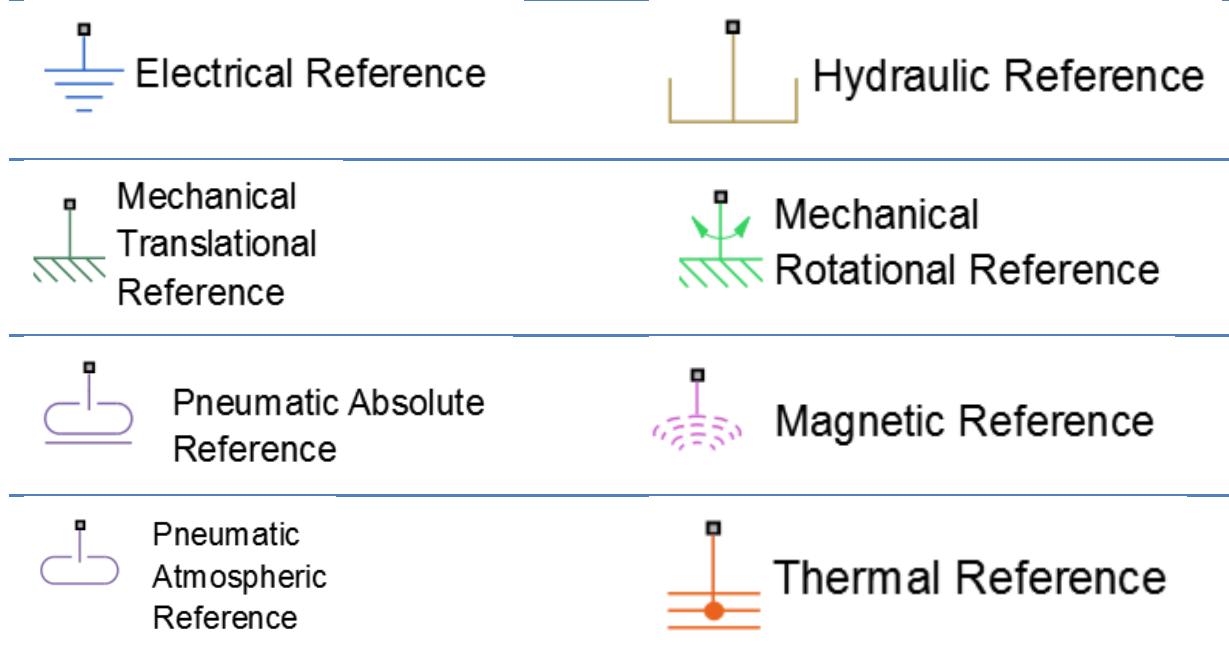


Figure 120 : les références des principaux domaines physiques de Simscape

C. Variables de type « Across » et « Through » 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 entraîner 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 température	Débit massique et flux d'entropie
Thermique	Température	Flux thermique et flux d'entropie

Figure 121 : 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

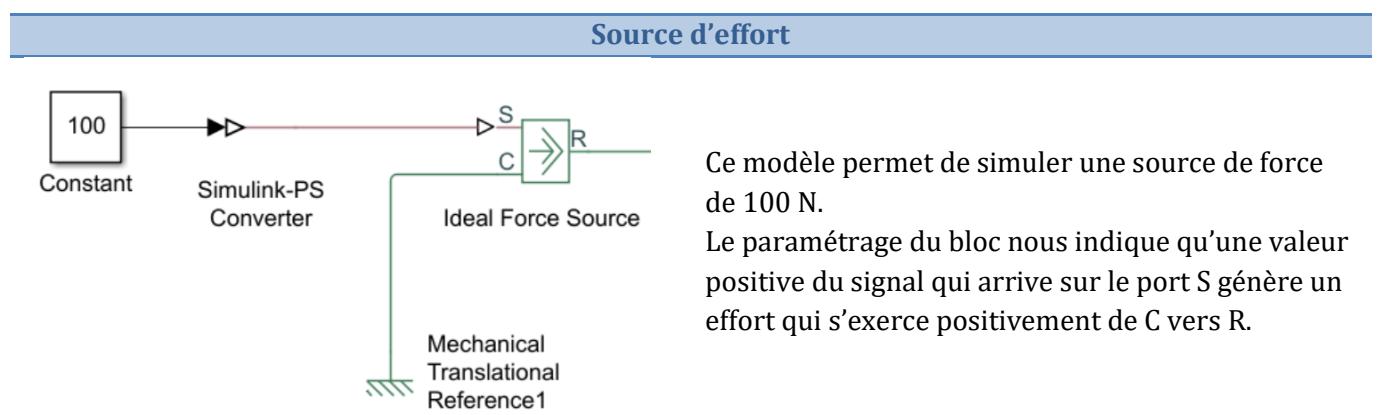
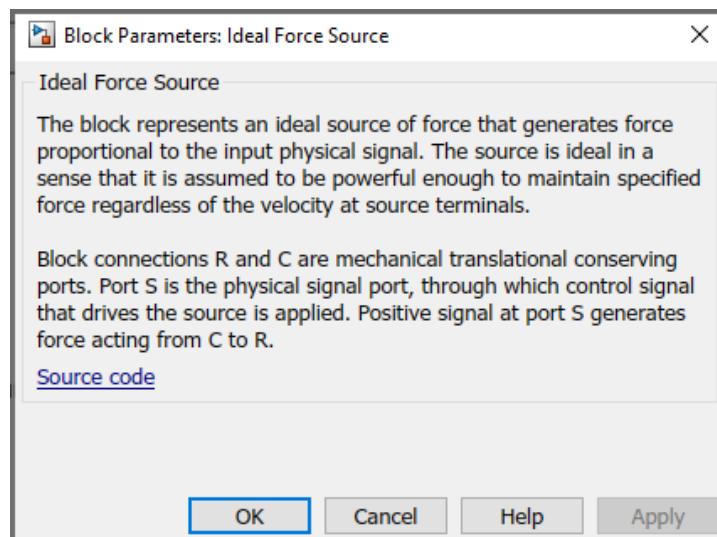
1. Utilisation de composants actifs

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/Foundation Library/Mechanical/Mechanical Sources

Fenêtre de paramétrage:



Sur le modèle de la Figure 122, une force de 100 N, dirigée de C vers R va agir sur l'ensemble ressort amortisseur.

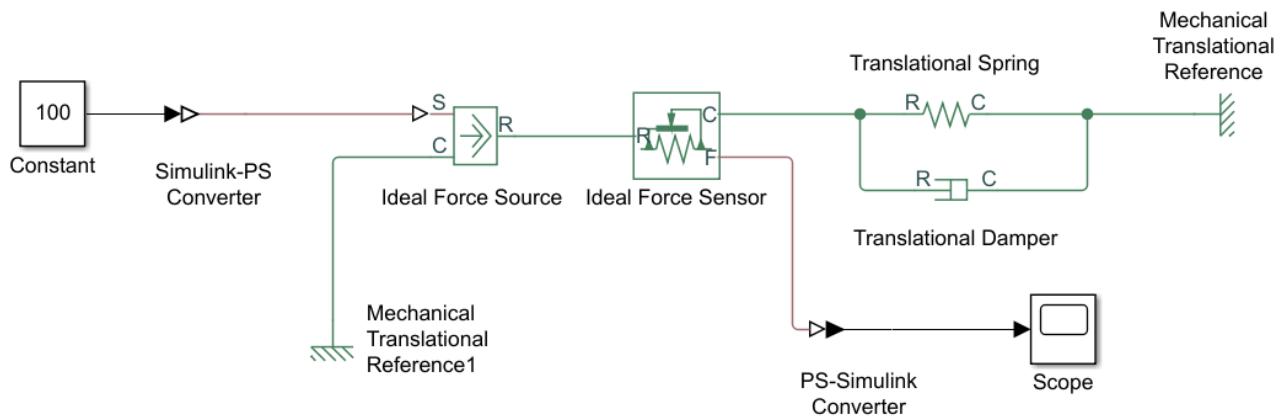


Figure 122 : source de force orientée positivement

Sur le modèle de la Figure 123, les ports C et R ont été inversés, et la force va s'appliquer à l'opposé de la configuration de la Figure 122.

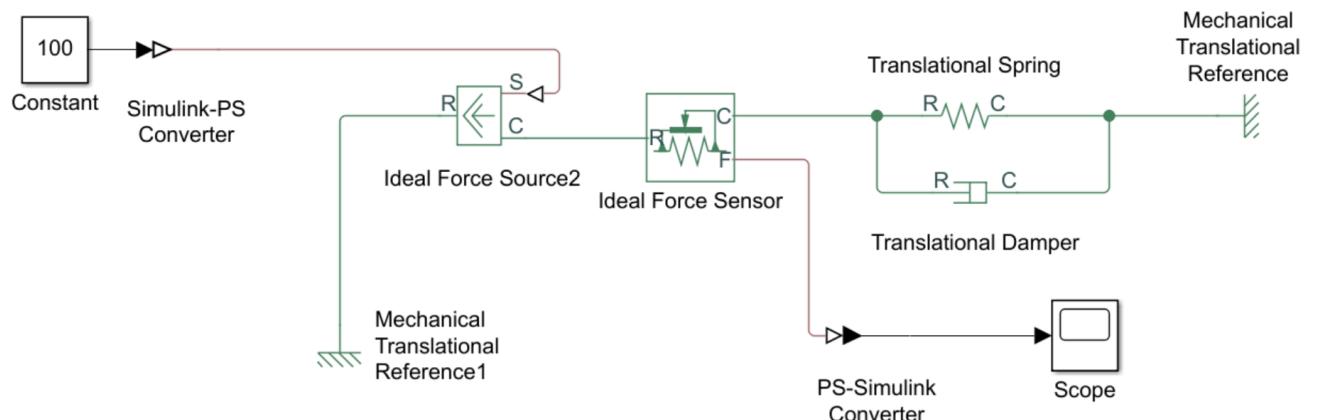


Figure 123 : source de force orientée négativement

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

Ouvrir le fichier « orientation_source_force.slx ».

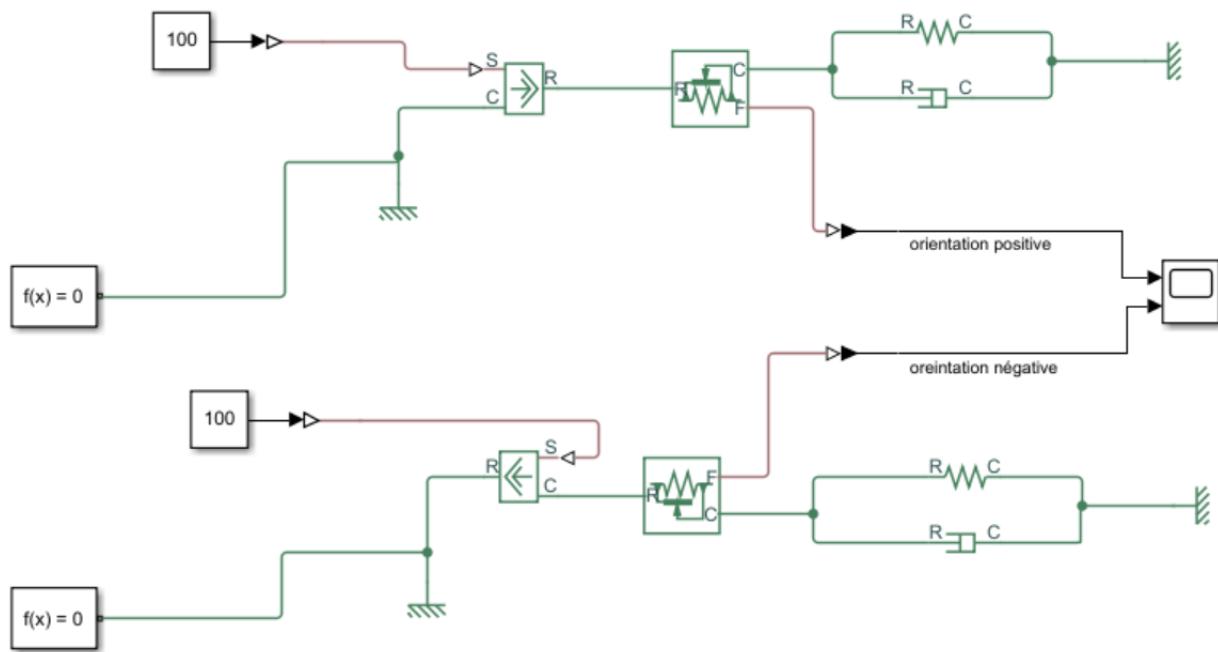


Figure 124 : orientation d'une source de force

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 124, 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

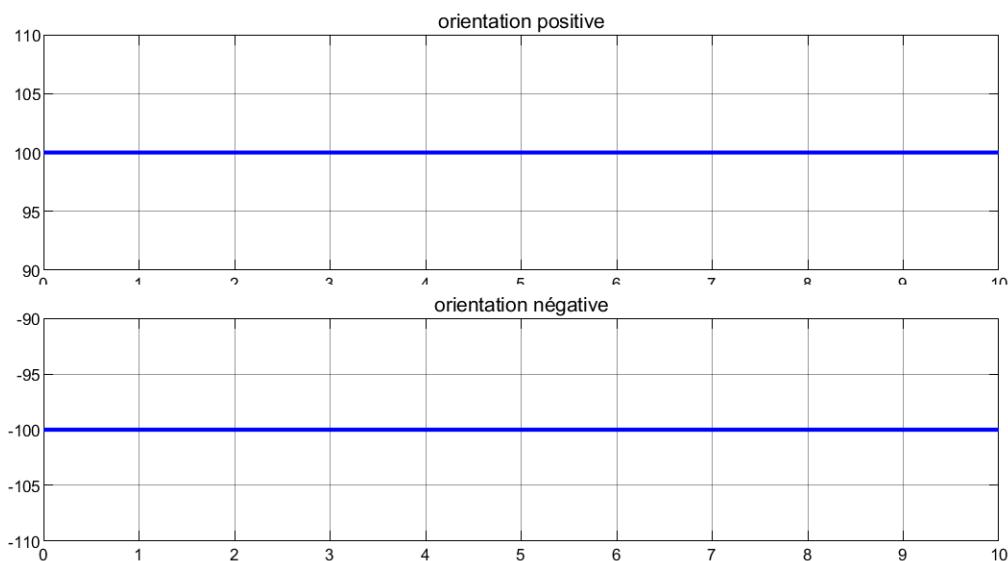


Figure 125 : visualisation de l'influence de l'orientation de la source de force

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 :

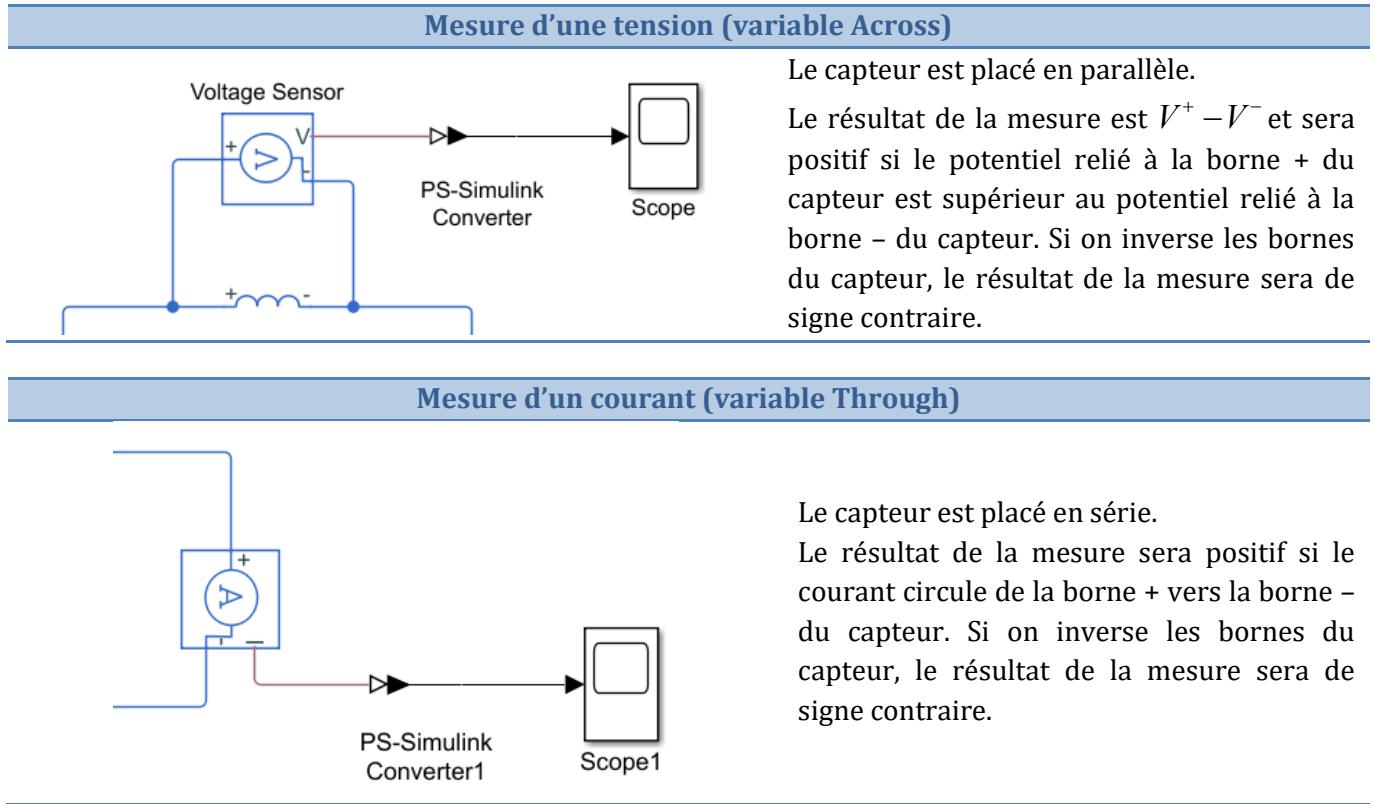


Figure 126 : exemple de mesure de grandeurs électriques

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.

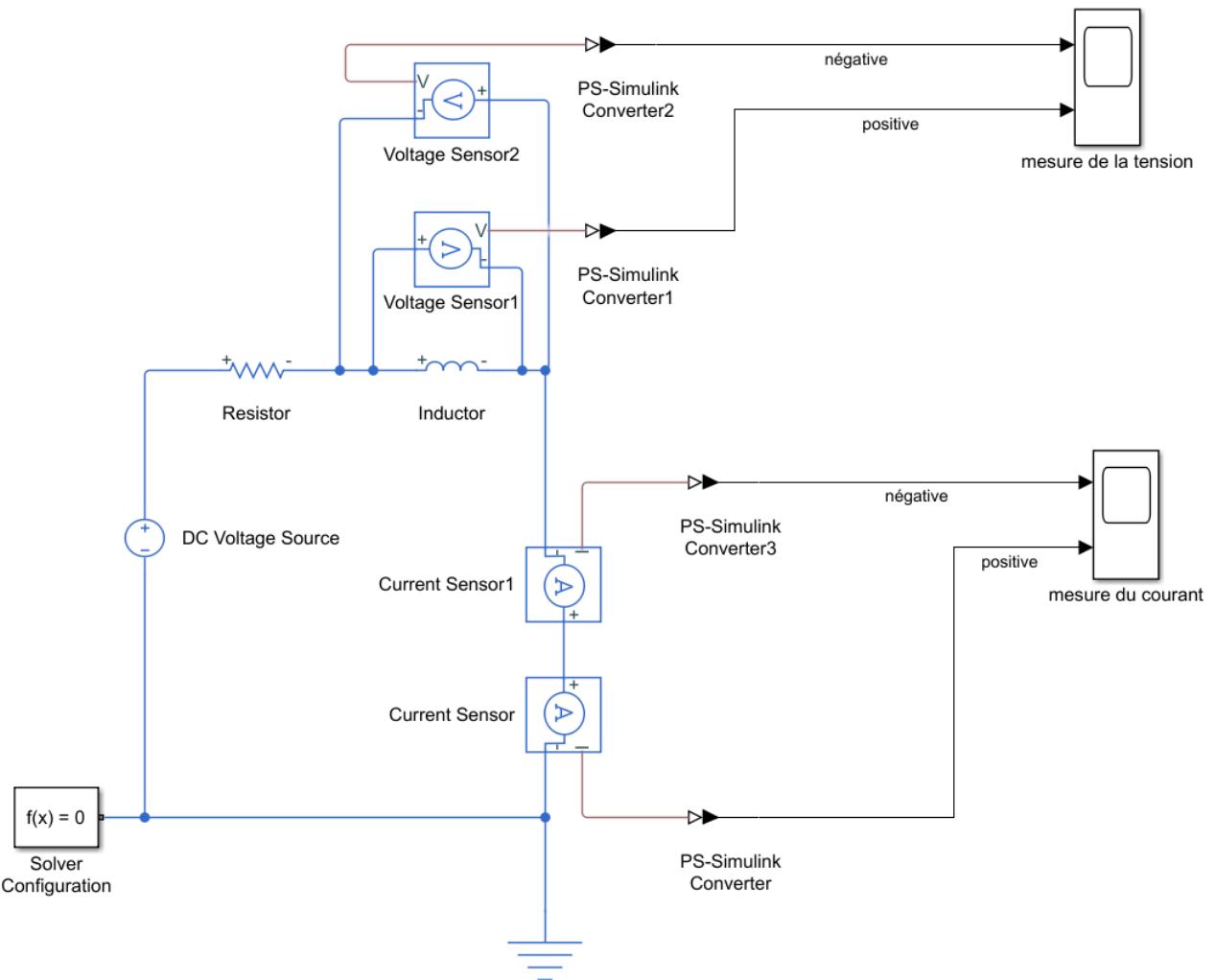


Figure 127 : implantation des capteurs

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

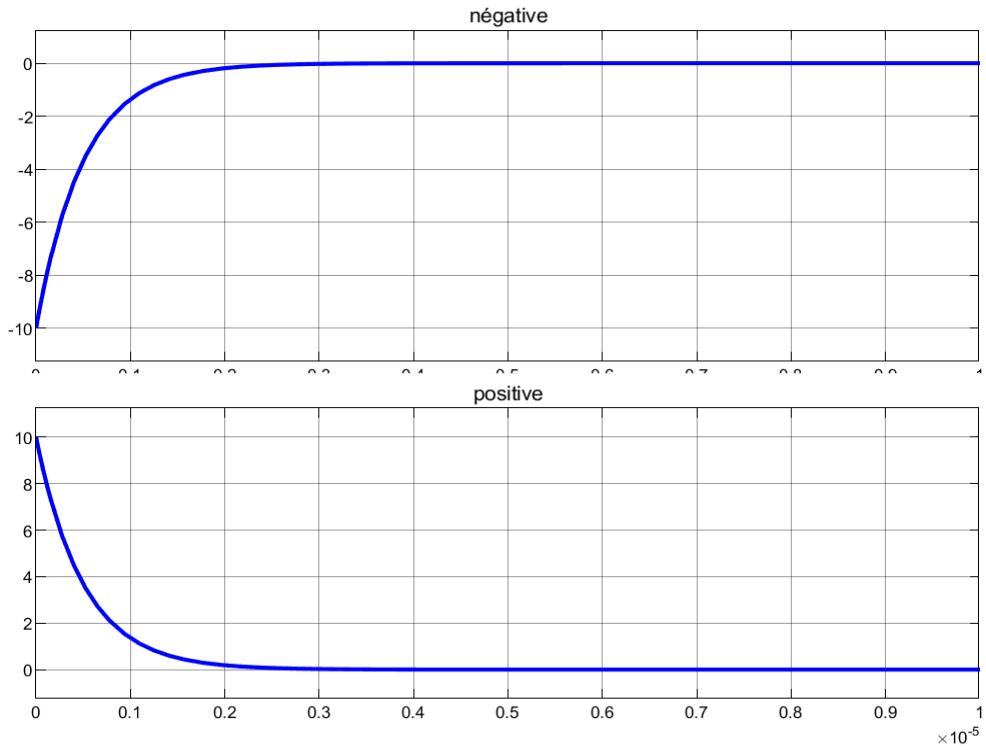


Figure 128 : visualisation de la tension aux bornes de la bobine pour les deux orientations des capteurs

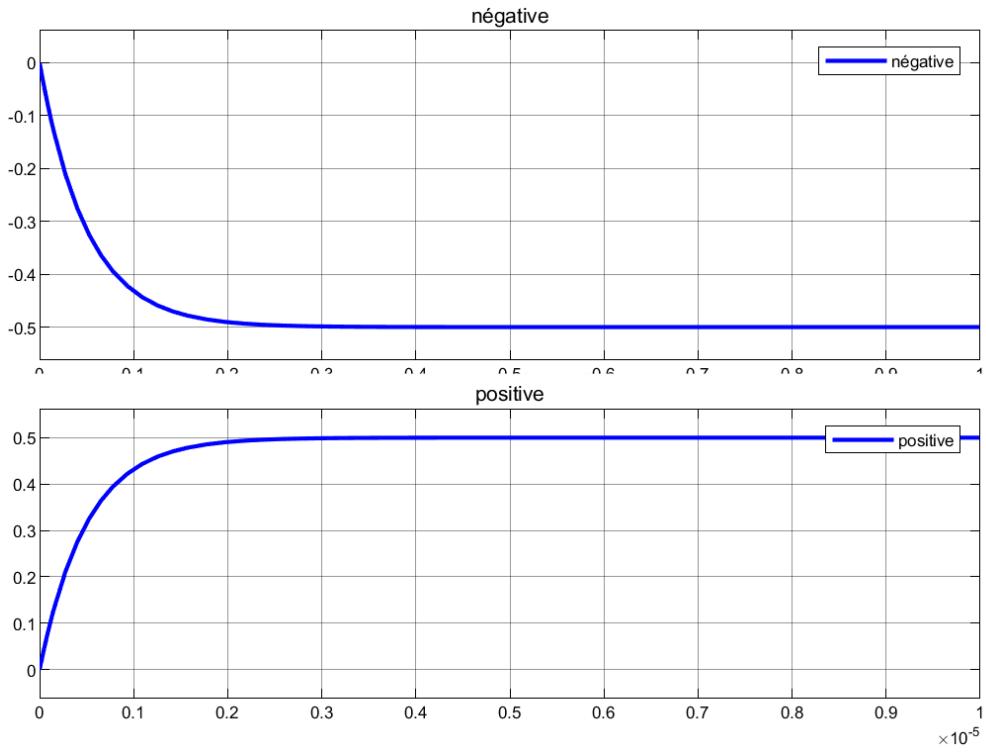


Figure 129 : visualisation de l'intensité du courant dans le circuit en fonction des deux orientations des capteurs

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.

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 130). Développer la fenêtre **Solver details**.

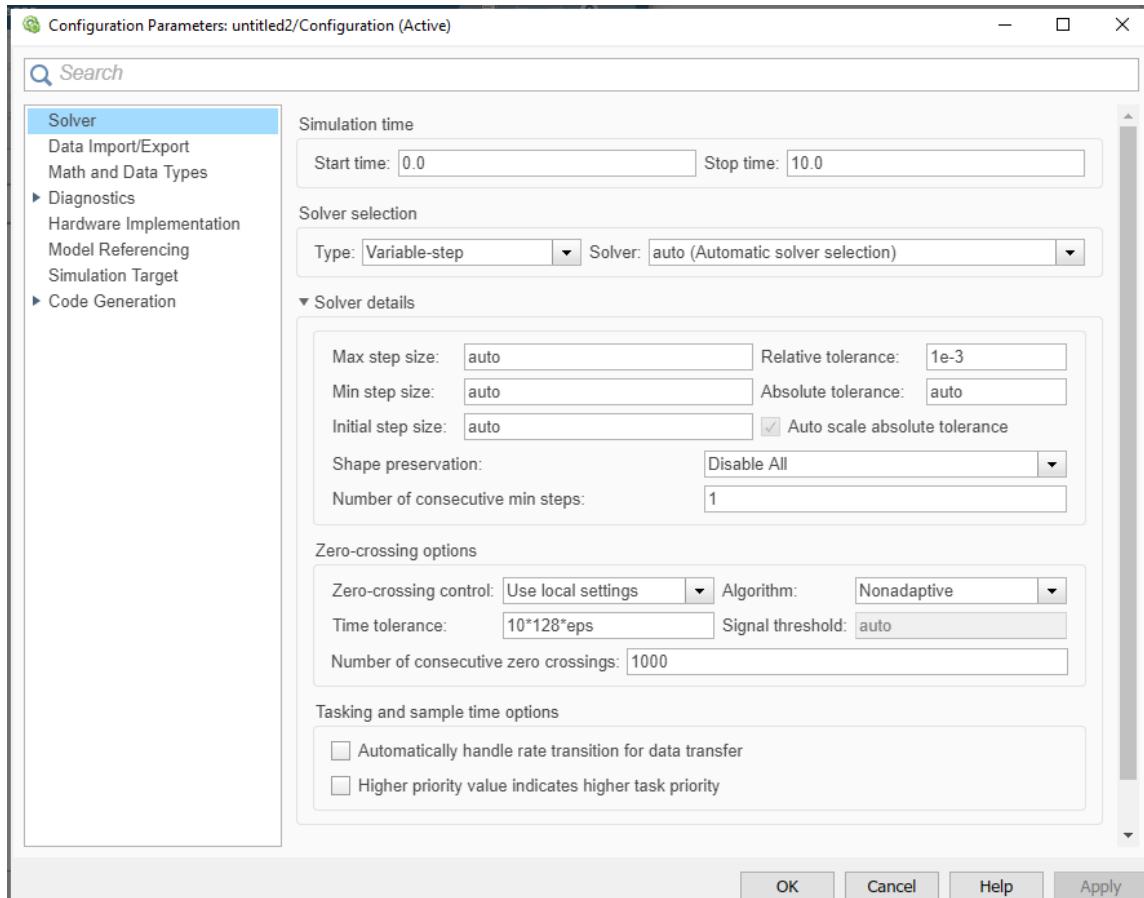


Figure 130 : fenêtre de paramétrage du solveur

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.

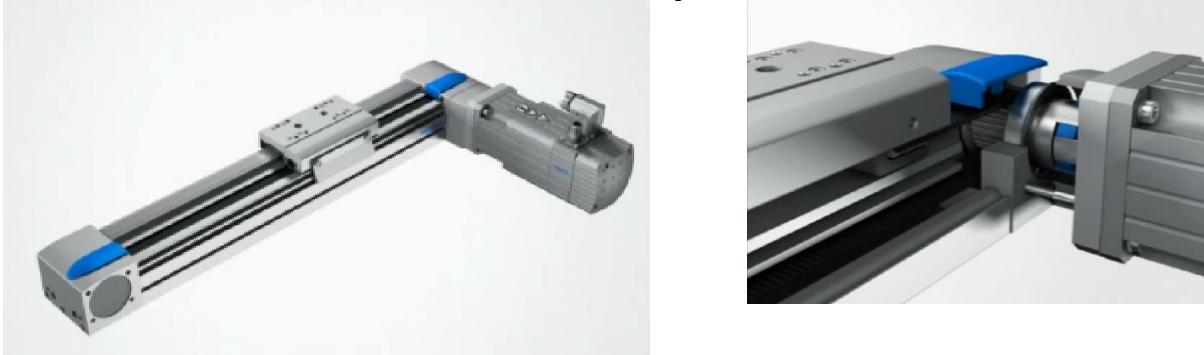


Figure 131 : photo de l'axe linéaire

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 132, les composants qui n'ont pas encore été utilisés auparavant dans le document.

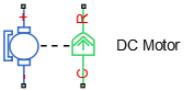
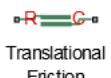
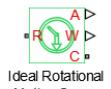
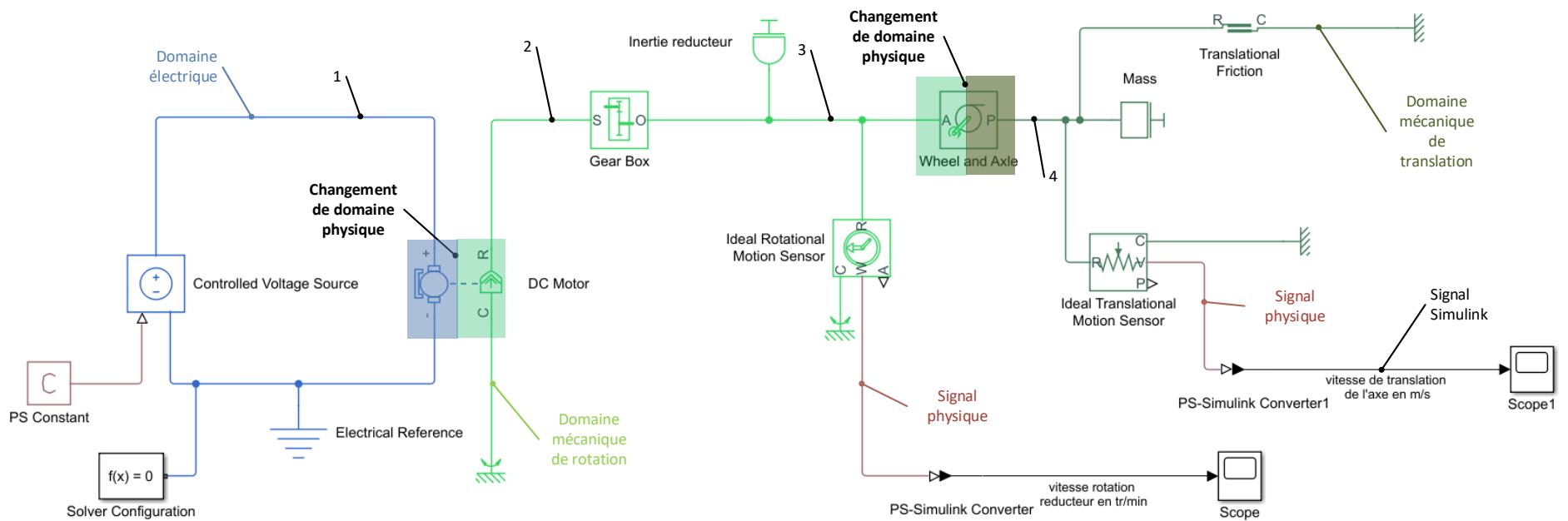
Fonction du composant	Représentation	Bibliothèque
Source de tension variable	 Controlled Voltage Source	Simscape/Foundation Library/Electrical/Electrical Sources
Source de signal physique	 PS Constant	Simscape/Foundation Library/Physical Signals/Sources
Moteur à courant continu		Simscape/Electrical/Electromechanical/Brushed Motors
Réducteur		Simscape/Foundation Library/Mechanical/Mechanisms
Inertie	 Inertia	Simscape/Foundation Library/Mechanical/Rotational Elements
Conversion mouvement de rotation en mouvement de translation		Simscape/Foundation Library/Mechanical/Mechanisms
Masse	 Mass	Simscape/Foundation Library/Mechanical/Translational Elements
Frottement sec et visqueux en translation		Simscape/Foundation Library/Mechanical/Translational Elements
Capteur de position ou de vitesse de translation		Simscape/Foundation Library/Mechanical/Mechanical Sensors
Capteur d'angle ou de vitesse de rotation		Simscape/Foundation Library/Mechanical/Mechanical Sensors

Figure 132 : les composants nécessaires à la modélisation de l'axe linéaire avec Simscape

Le modèle **Simscape** de l'axe linéaire est présenté Figure 133 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 133 : 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**, cliquer sur l'icône 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 134.

La Figure 135 donne quelques commandes utiles à la mise en forme du modèle.

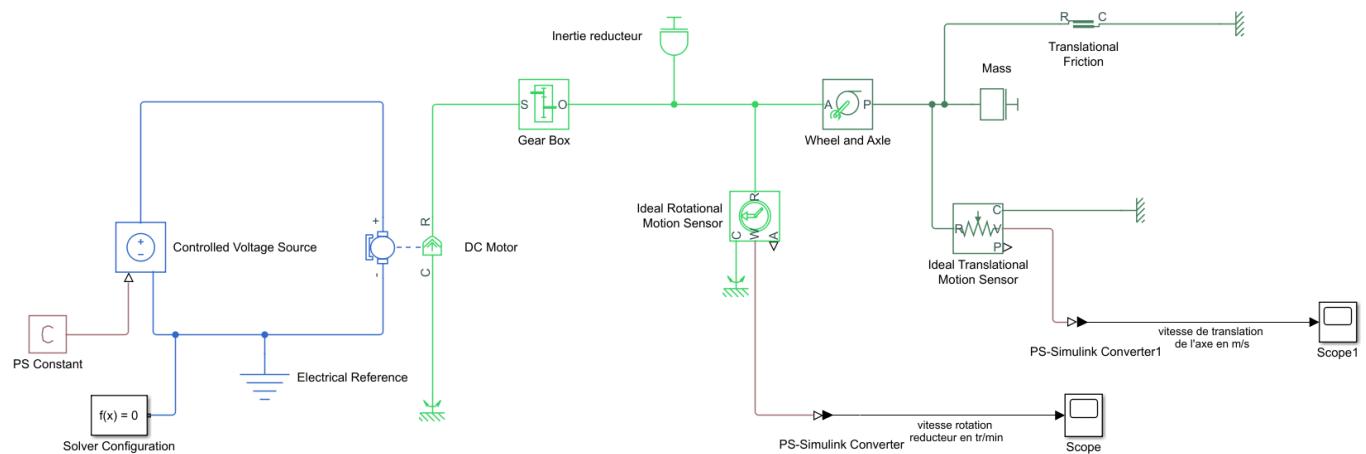


Figure 134 : modèle Simscape de l'axe linéaire

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 apparaître 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 135 : commandes utiles

3. Paramétrage des composants

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

Paramétrage

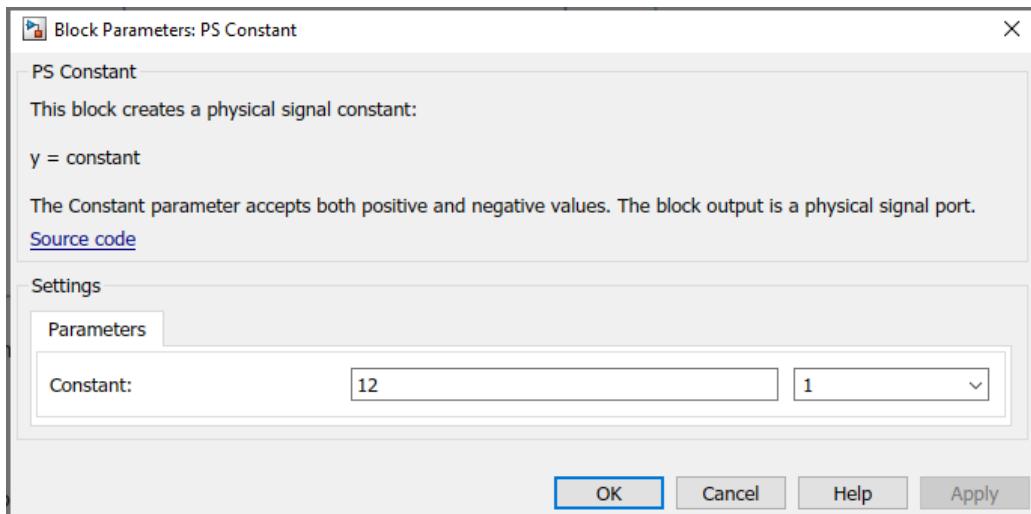
PS Constant



Simscape/Foundation 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

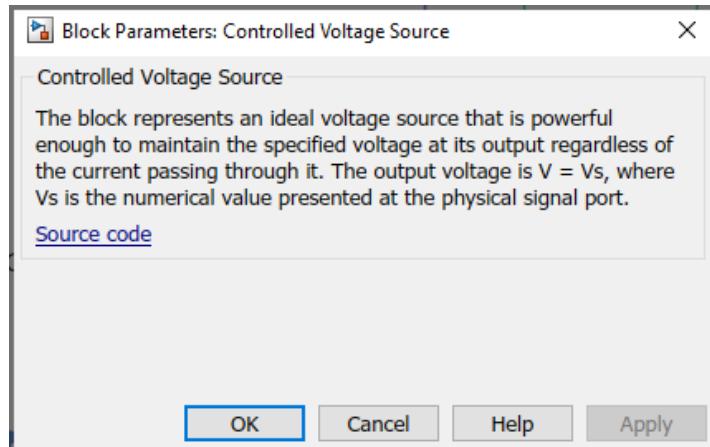


Controlled Voltage Source

Simscape/Foundation
Library/Electrical/Electrical Element

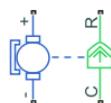
Ce composant ne nécessite aucun paramétrage.

Le port correspondant au signal physique, indique à la source de tension variable la valeur de la tension imposée aux bornes de la source.



Paramétrage

DC MOTOR

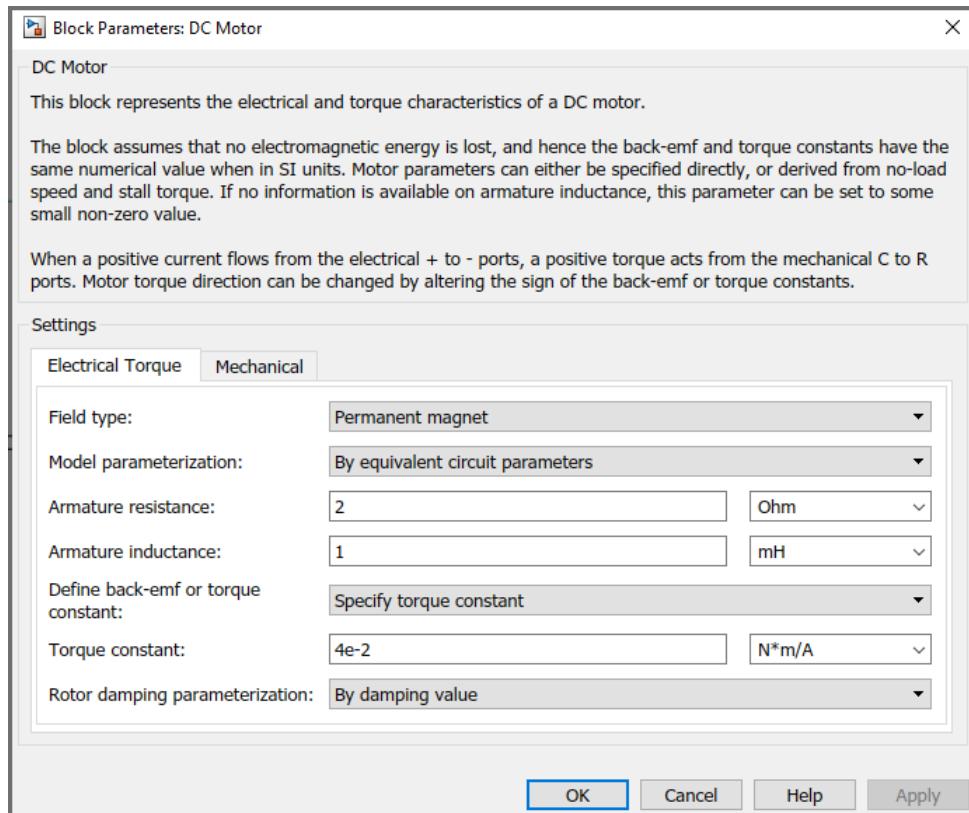


Simscape/Electrical/Electromechanical/Brushed Motors

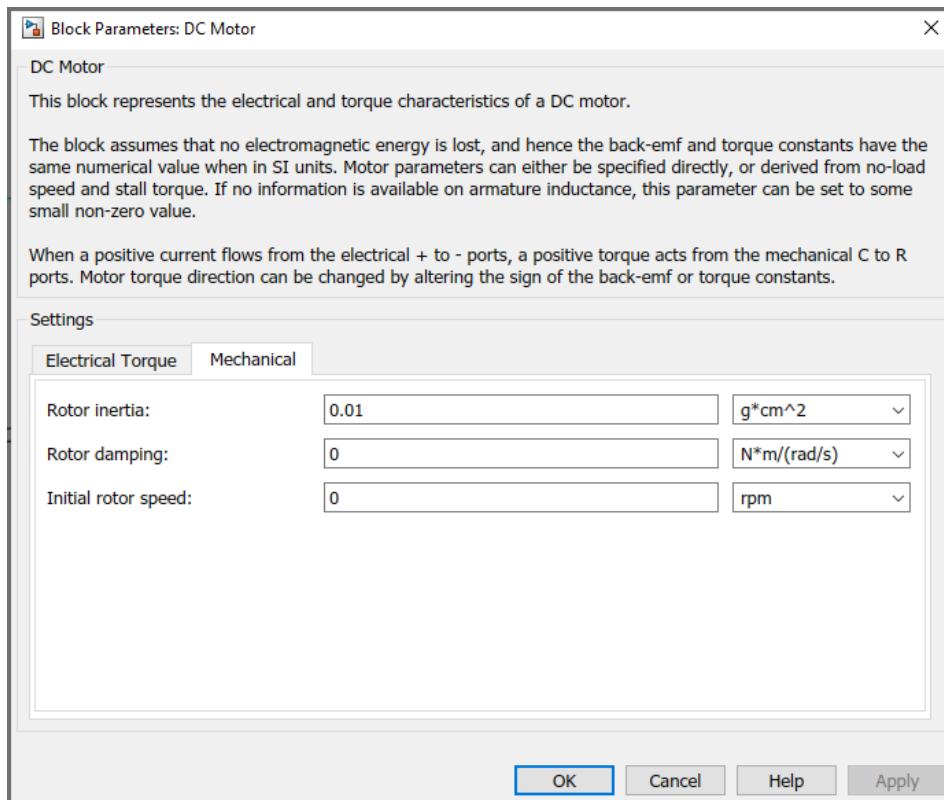
Ce composant modélise un moteur à courant continu. Il possède 4 ports de type **PCP**, 2 du domaine électrique (+ et -) et 2 du domaine mécanique de rotation (**C** et **R**). Le PCP du domaine mécanique **C** (Cage, partie fixe) sera relié à une référence mécanique de rotation que l'on pourra associer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

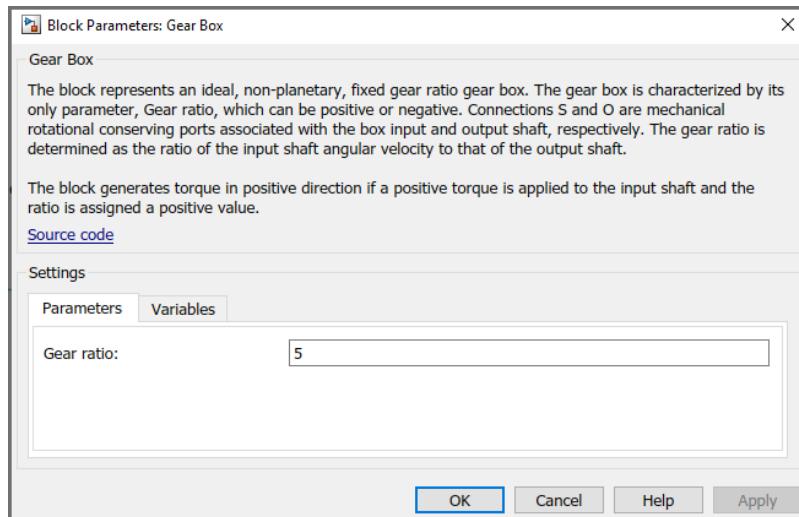


Simscape/Foundation
Library/Mechanical/Mechanisms

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.



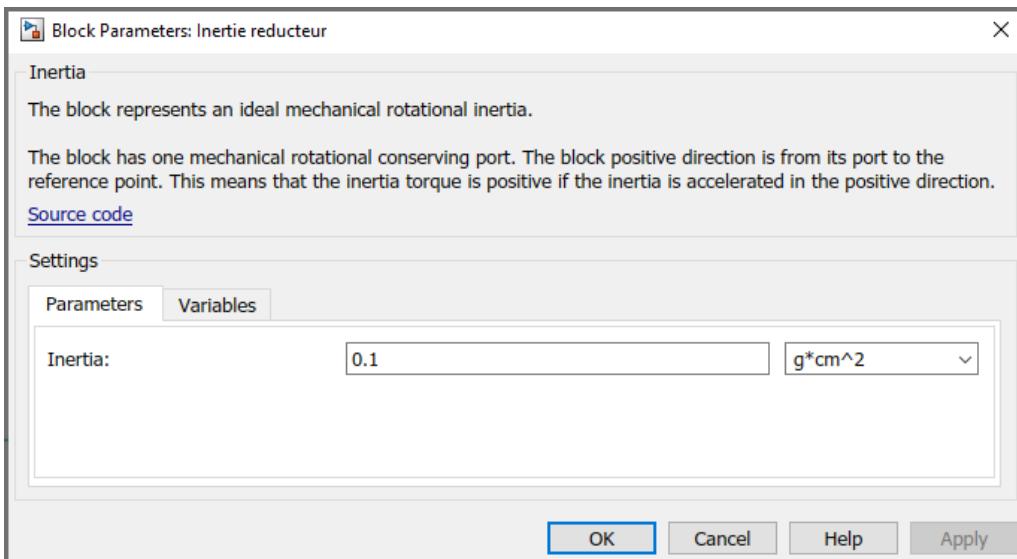
Paramétrage

INERTIA



Simscape/Foundation
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

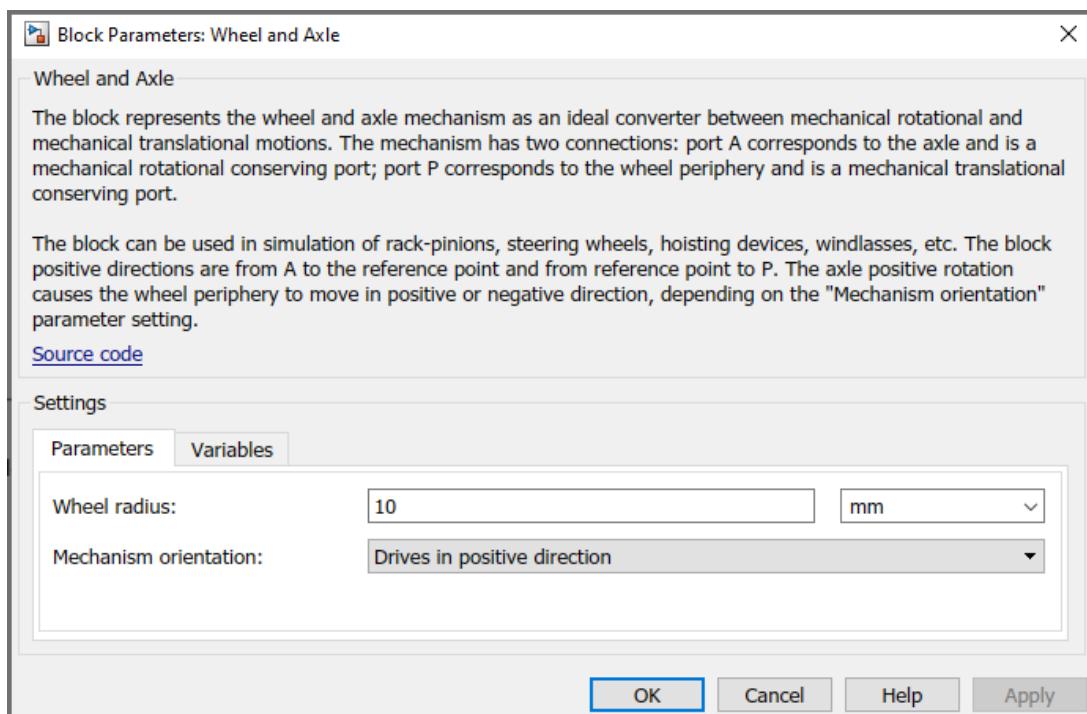


Simscape/Foundation
Library/Mechanical/Mechanisms

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

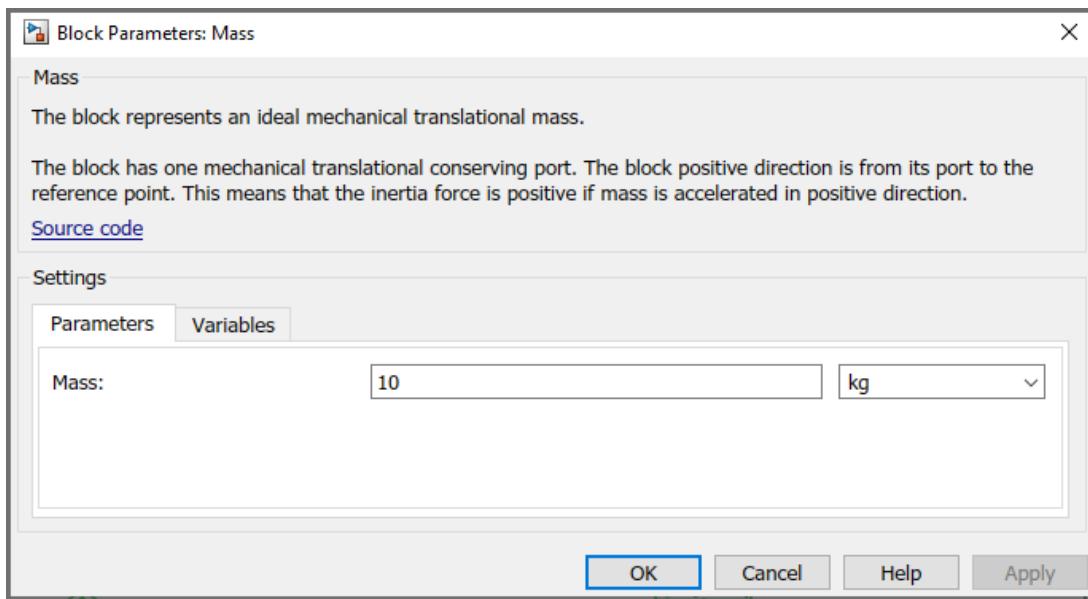
MASS



Simscape/Foundation
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 entraîné en translation par la poulie.



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

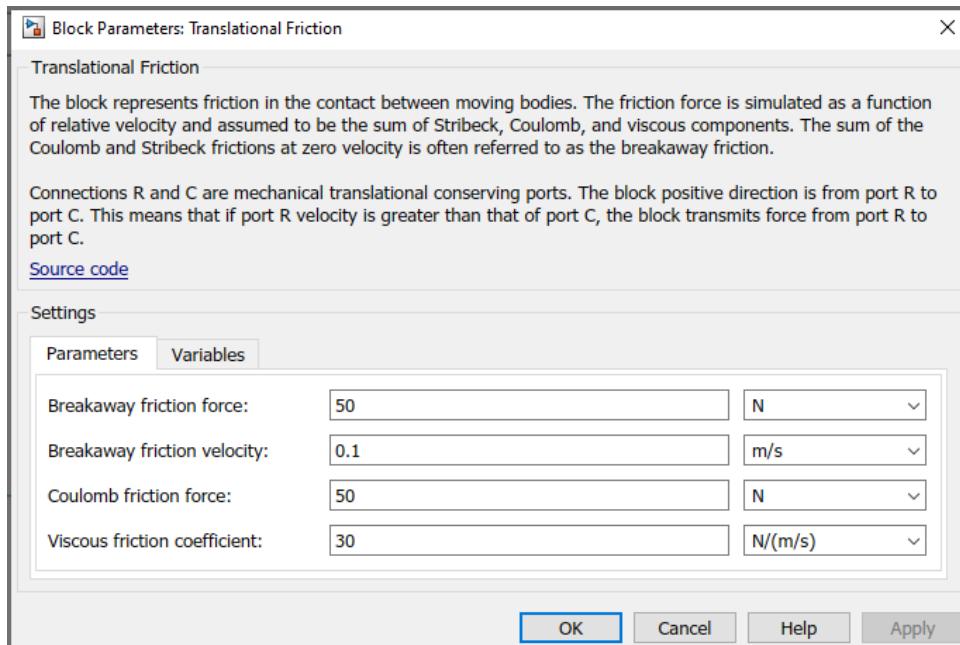
Paramétrage

TRANSLATIONAL FRICTION



Simscape/Foundation
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)

Breakaway friction velocity : ce terme permet de caractériser le passage du frottement d'adhérence au frottement de glissement. Plus ce paramètre est faible et plus le passage se fera brutalement.

Coulomb friction force : force de frottement de glissement (qui s'exerce pendant le mouvement)

Viscous friction coefficient : coefficient de frottement visqueux

Paramétrage

IDEAL TRANSLATIONAL
MOTION SENSOR

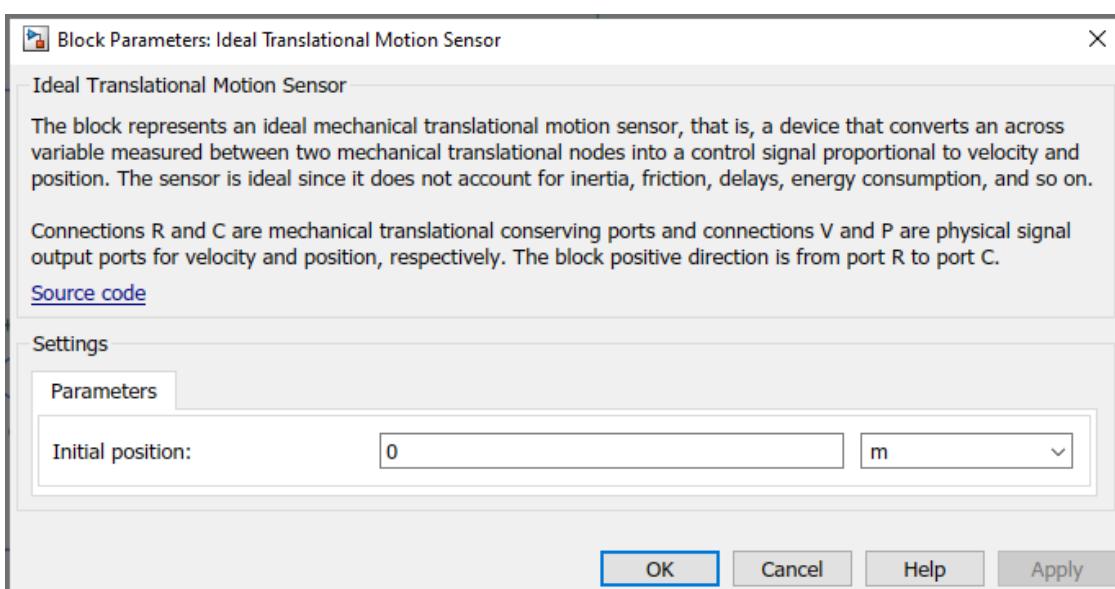


Simscape/Foundation
Library/Mechanical/Mechanical Sensors

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

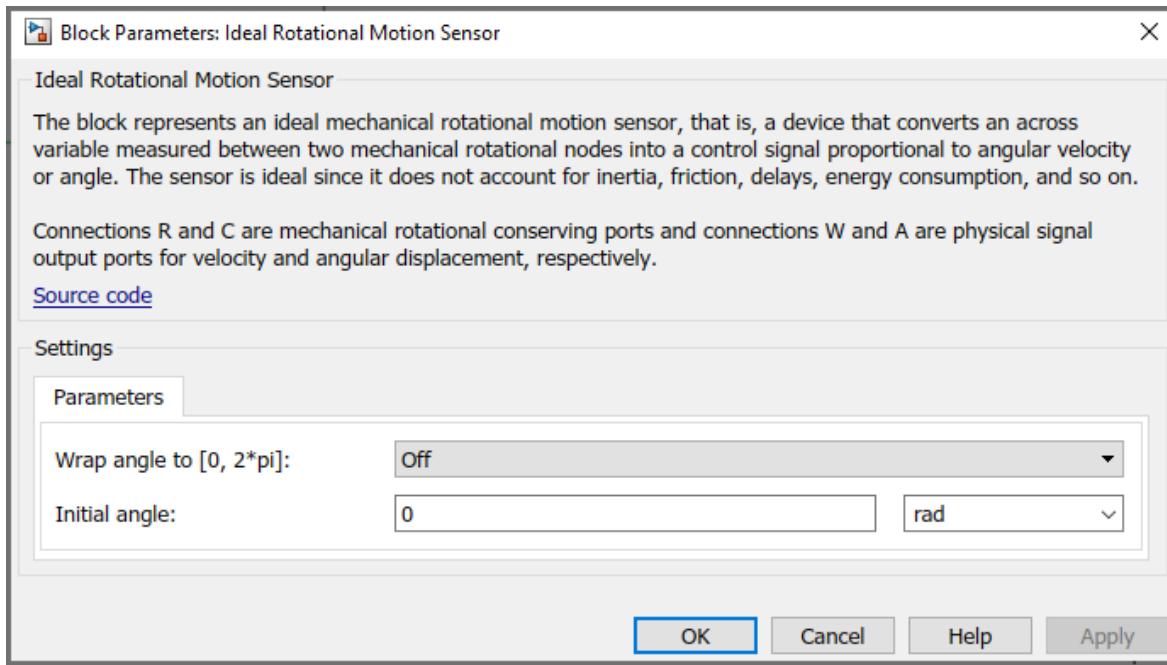


Simscape/Foundation
Library/Mechanical/Rotational Sensors

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

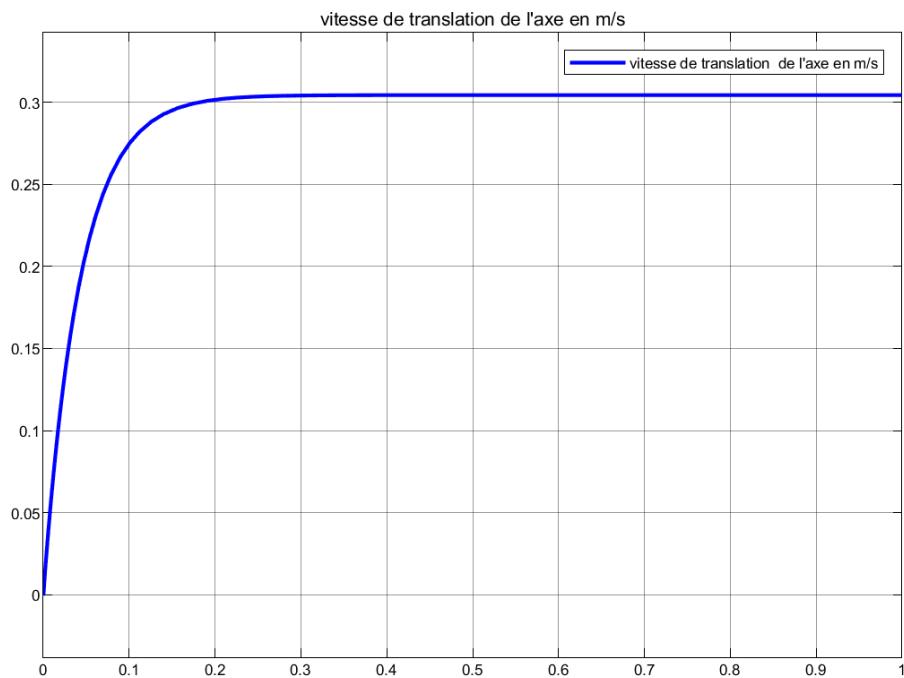
Ce composant permet de convertir un signal physique image d'une grandeur mesurée en signal Simulink afin de pouvoir être affiché dans un Scope. Spécifier que la vitesse de translation du chariot est relevée en m/s et la vitesse de rotation de l'arbre en tr/min (rpm pour Simscape)

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

4. Simulation du modèle en boucle ouverte

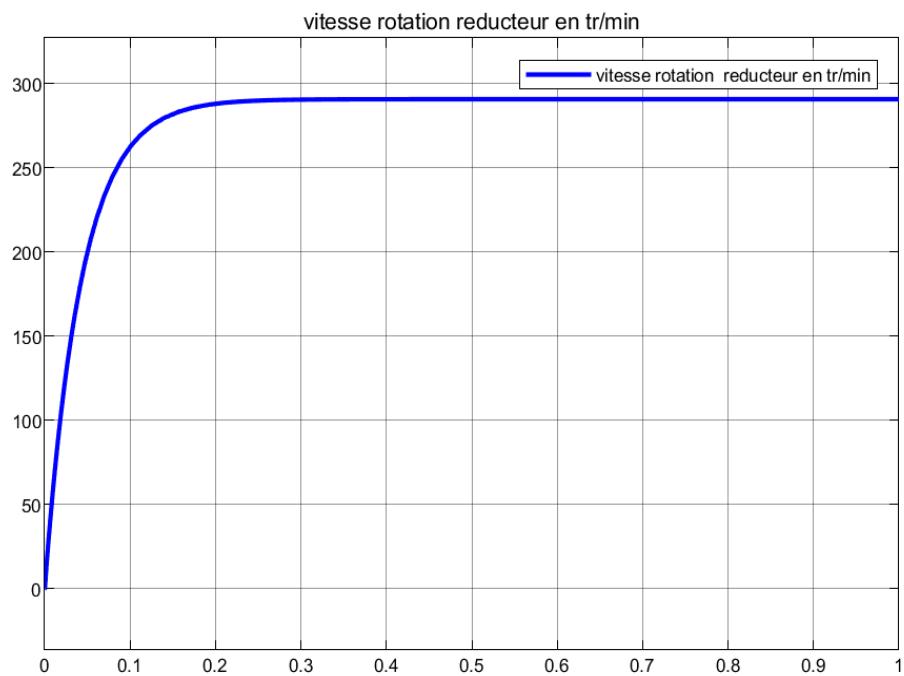
Lancer la simulation en spécifiant un temps de simulation de **1s**.

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 Simscape Results Explorer

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 planter 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 **Simscape Results Explorer** de visualiser l'évolution de toutes les variables.

Il faut dans un premier temps spécifier le nom de la variable qui stockera toutes les données de la simulation.



Cliquer sur l'icône **Model Setting**, puis choisir l'onglet **Simscape** dans la partie gauche de la fenêtre.

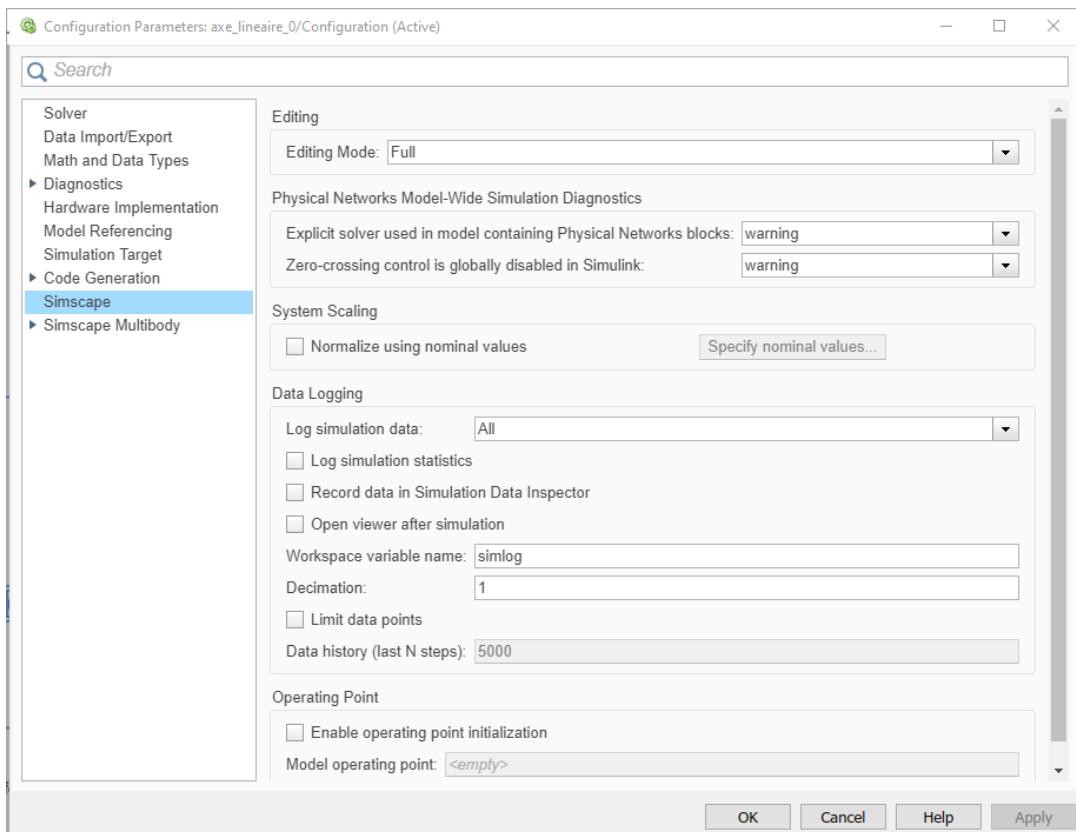


Figure 136 : spécification des variables Simscape à prendre en compte dans le logger

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 et cliquer sur l'icône **Simscape Results** pour ouvrir la fenêtre **Simscape Results Explorer** (Figure 137).

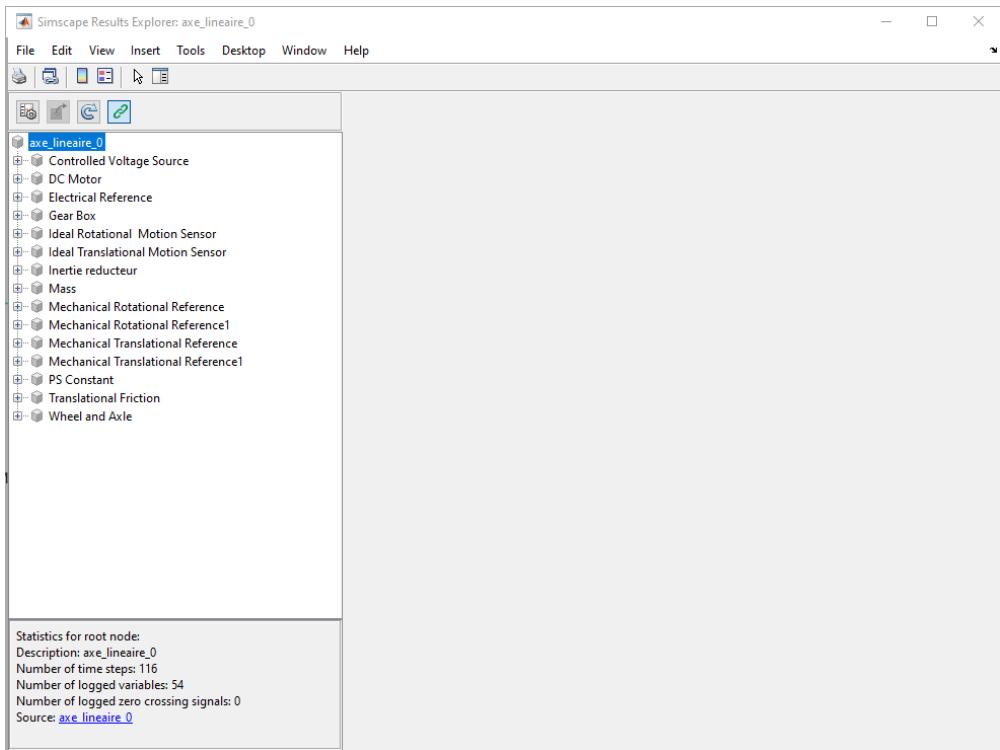


Figure 137 : fenêtre du Simscape Result Explorer

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 (Figure 138).

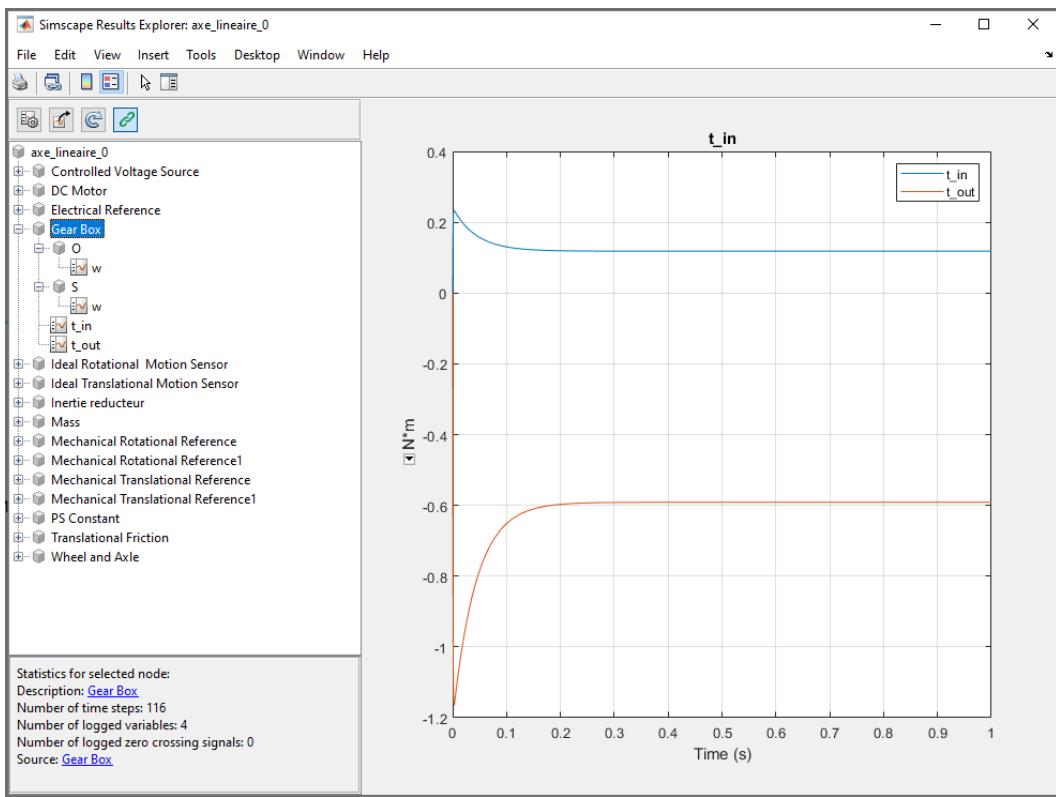
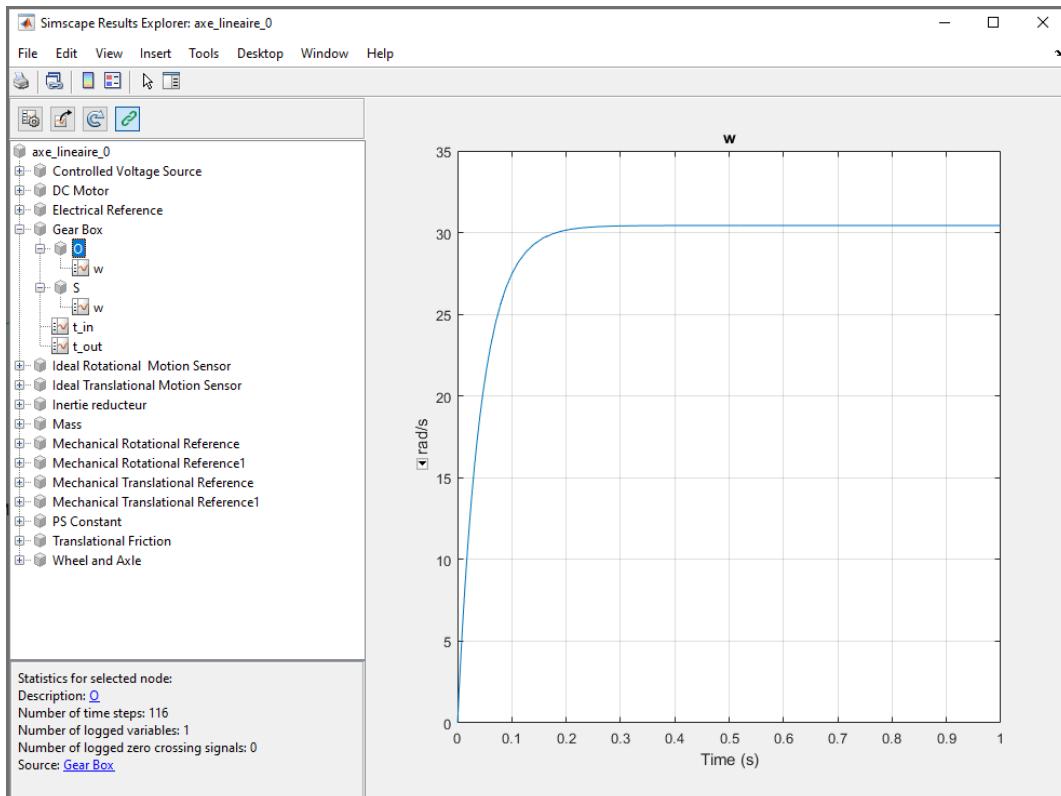


Figure 138 : visualisation des résultats de la simulation

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 Simscape Results Explorer 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 Simscape Result Explorer 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 139 les composants qui composeront le sous-système puis faire glisser la souris sur l'icône **Create Subsystem** en bas à droite de la sélection.

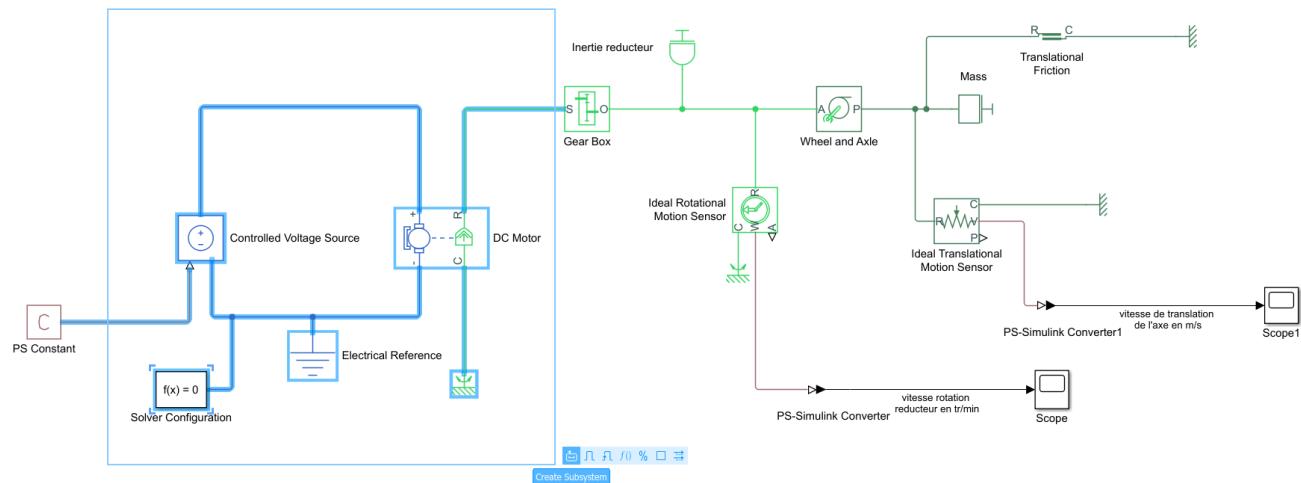


Figure 139 : création d'un sous-système

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 ou pivoter éventuellement le sous-système pour voir apparaître les ports **Controlled Voltage Source1** et **DC Motor1** du sous-système (Figure 141).

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 140 : commandes utiles

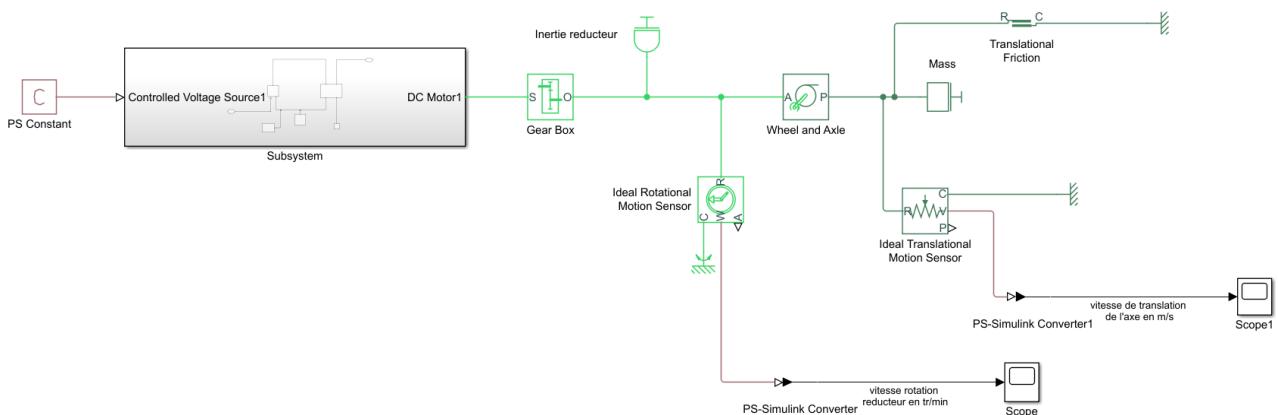


Figure 141 : création d'un sous-système

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.

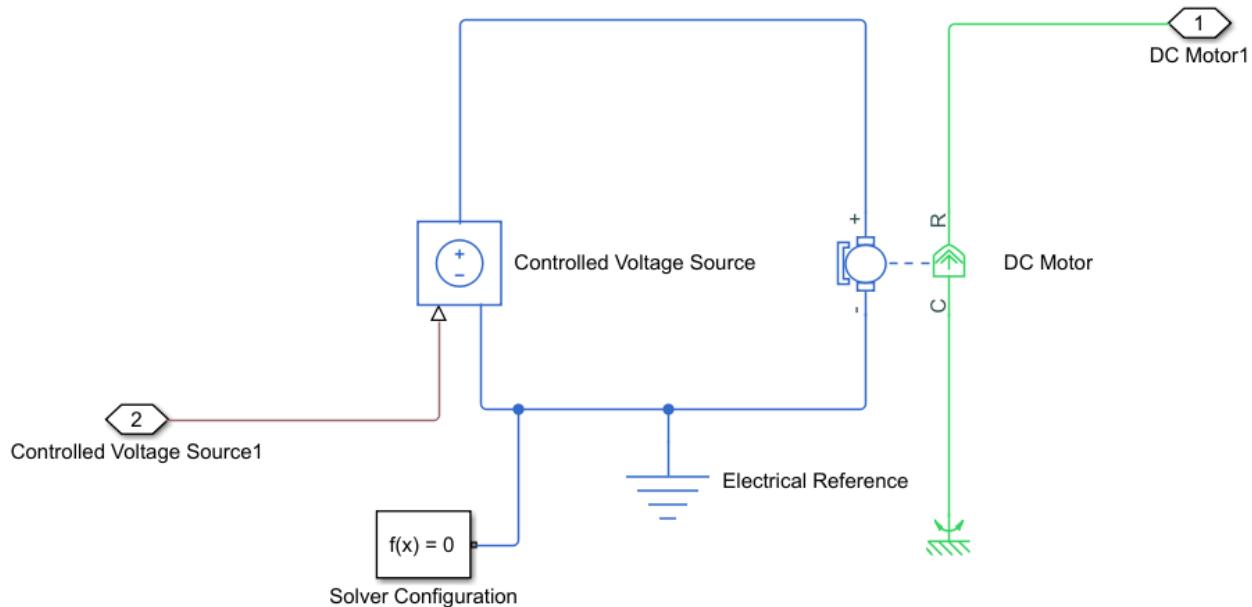


Figure 142: visualisation du contenu du sous-système

Nous pouvons observer que les connexions avec l'extérieur apparaissent sous la forme de ports **Controlled Voltage Source1** et **DC Motor1**. Il est également possible de renommer ces ports.

Double-cliquer directement sur le nom du port **Controlled Voltage Source1** et le remplacer par « **Tension de commande** ».

Faire de même en remplaçant le nom **DC Motor1** par « **Arbre moteur** » (Figure 143)

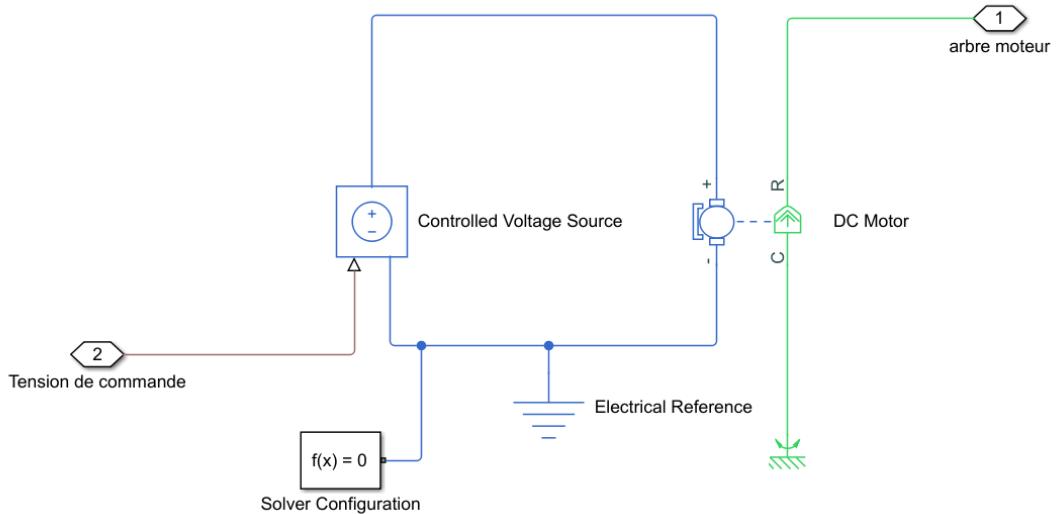


Figure 143 : renommer les ports d'un sous-système

Revenir au modèle global en cliquant sur le nom du modèle global dans la barre de navigation ou en utilisant le Model Browser. Le Model Browser (situé à gauche de la fenêtre) permet de naviguer facilement d'un sous-système au système global (Figure 144). Il est également possible d'utiliser les flèches de navigation dans les différents niveaux du modèle.

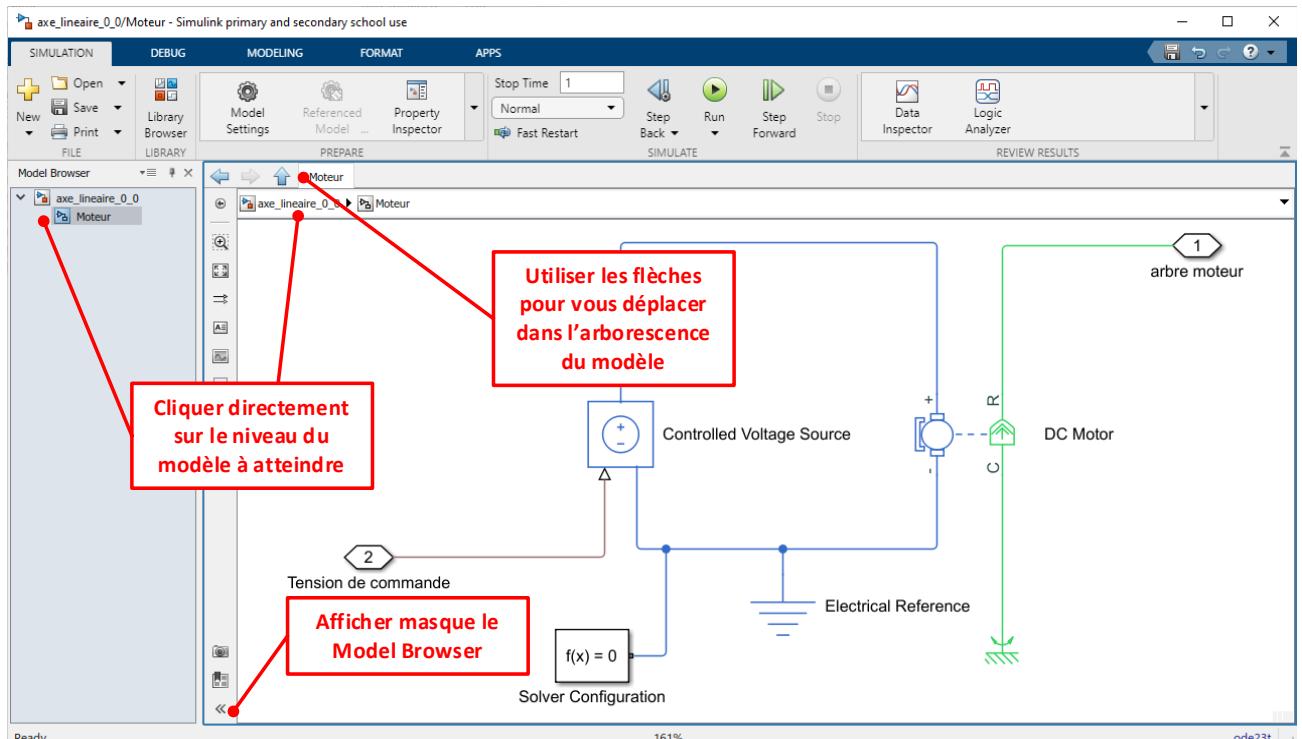


Figure 144 : naviguer dans les sous-systèmes d'un modèle

Vous devez obtenir la configuration de la Figure 145.

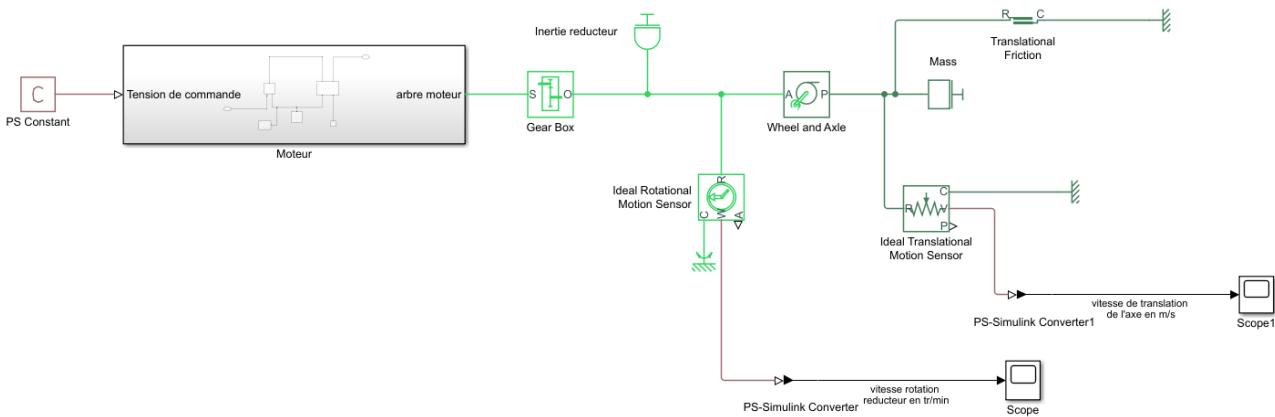


Figure 145 : modèle Simscape de l'axe linéaire avec sous-système « moteur »

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 146.

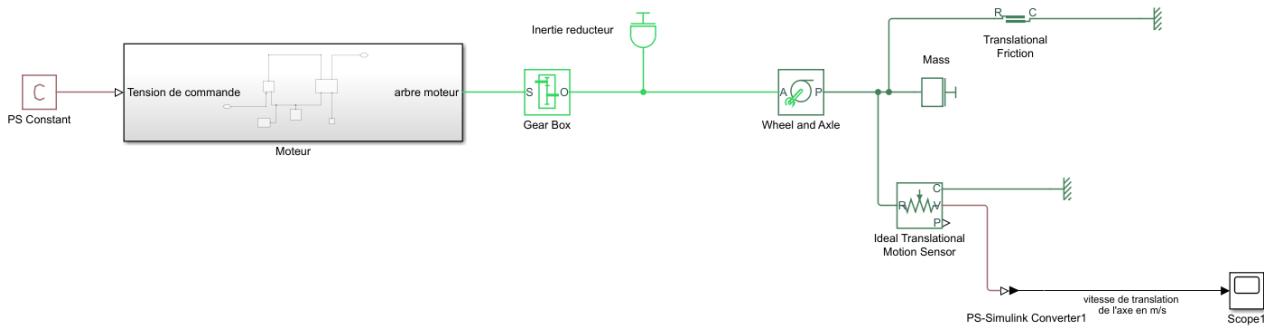


Figure 146 : modèle Simscape de l'axe linéaire sans le capteur de vitesse de rotation

Modifier la grandeur physique prélevée sur l'**Ideal Translational Motion Sensor** afin de relever la position linéaire du chariot (Figure 147).

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 147.

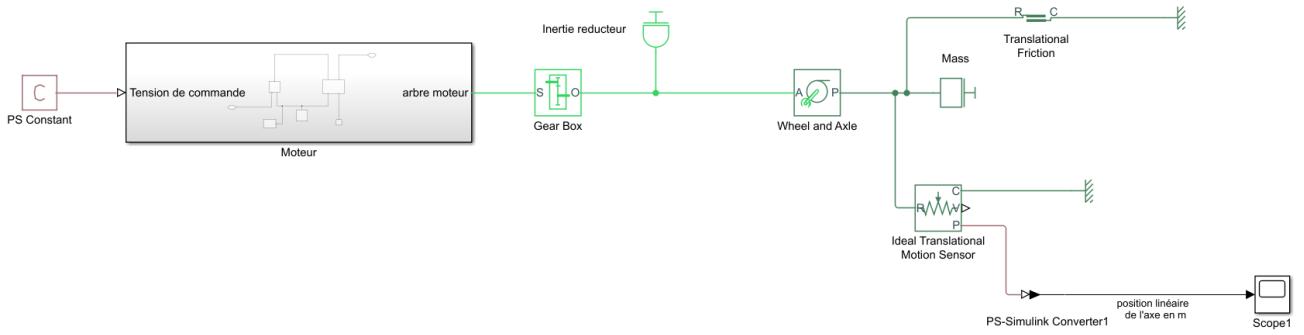


Figure 147 : modèle Simscape de l'axe linéaire avec le nom des signaux

Créer un sous-système « Axe » en sélectionnant les composants conformément à la Figure 148.

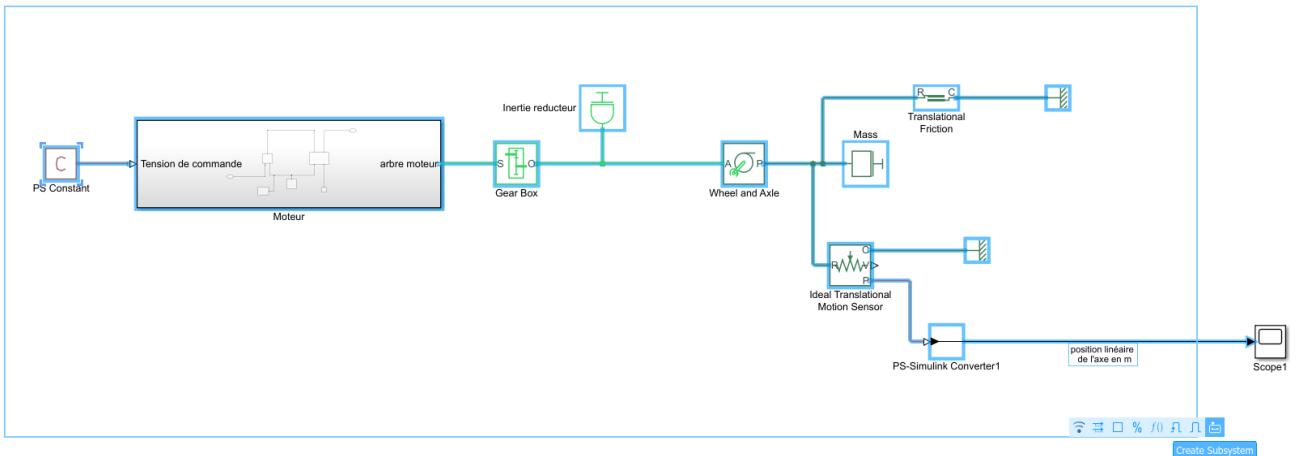


Figure 148 : modèle Simscape de l'axe linéaire, création du sous-système « Axe »

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

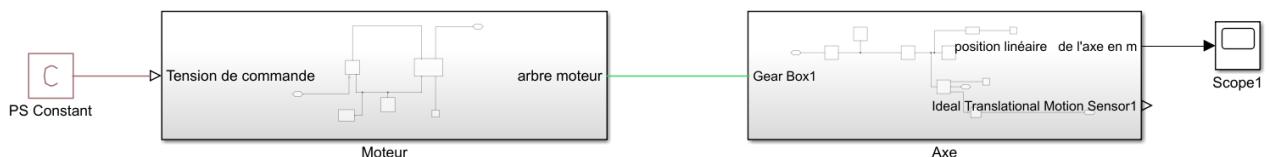


Figure 149 : modèle Simscape de l'axe linéaire avec sous-système « moteur » et « Axe »

On remarque qu'un port supplémentaire **Ideal Translational Motion Sensor1** a été créé.

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

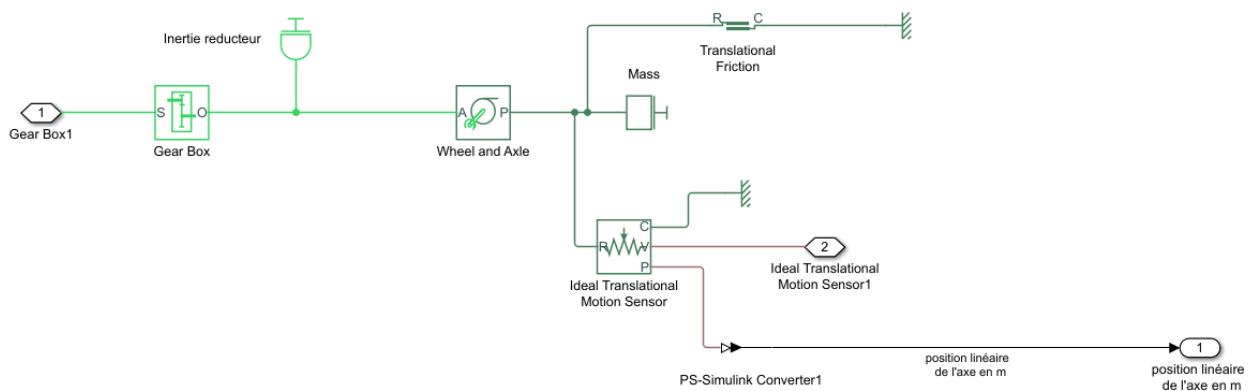


Figure 150 : suppression du port d'un sous-système

Le port **Ideal Translational Motion Sensor1** 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 **Ideal Translational Motion Sensor1** et le format du port **position linéaire de l'axe en m** sont différents. Le port **Ideal Translational Motion Sensor1** est un port de type **Simscape** (transmetteur de puissance, transmetteur de signal physique) et le port **position linéaire de l'axe en m** est un signal **Simulink** (signal numérique orienté, modélisation causale). Le logiciel permet de faire visuellement la distinction entre ces deux types de ports.

Vous devez obtenir le modèle de l'axe linéaire de la Figure 151.

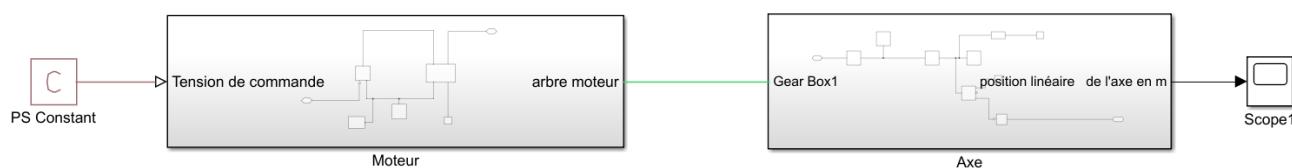


Figure 151 : modèle Simscape terminé de l'axe linéaire

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 afin d'améliorer la lisibilité du modèle. Pour cela cliquer droit sur le sous-système et sélectionner **Mask/Add Icon Image** (Figure 152).

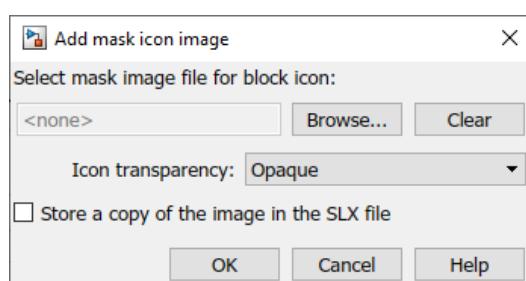


Figure 152 : mettre une image sur un sous-système

Cliquer sur **Browse** afin de chercher l'image **axe.jpg**, puis cliquer sur **Store a copy of the image in the SLX file** qui permettra de sauvegarder l'image dans le modèle. Cette image restera donc toujours reliée au Mask du système. Cliquer ensuite sur **OK**.

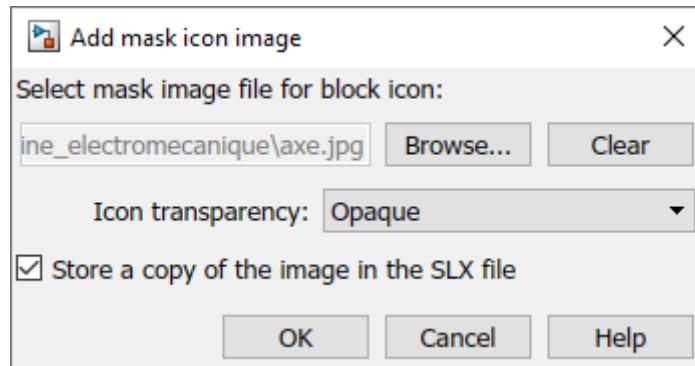


Figure 153 : configuration de la fenêtre « Add mask icon image »

Redimensionnez éventuellement la taille du sous-système pour que l'image retrouve ses proportions initiales (Figure 154).

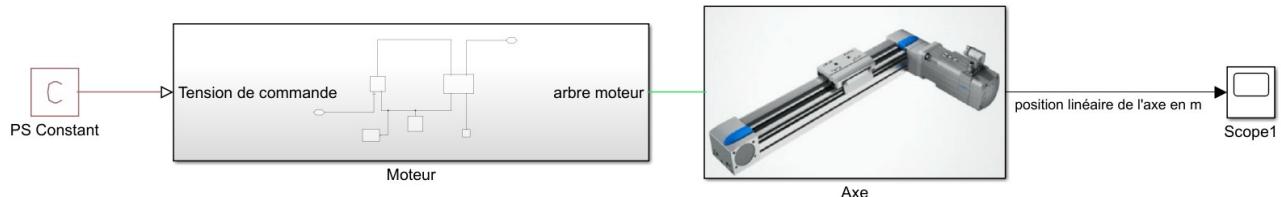


Figure 154 : affichage d'une image sur un sous-système

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 :

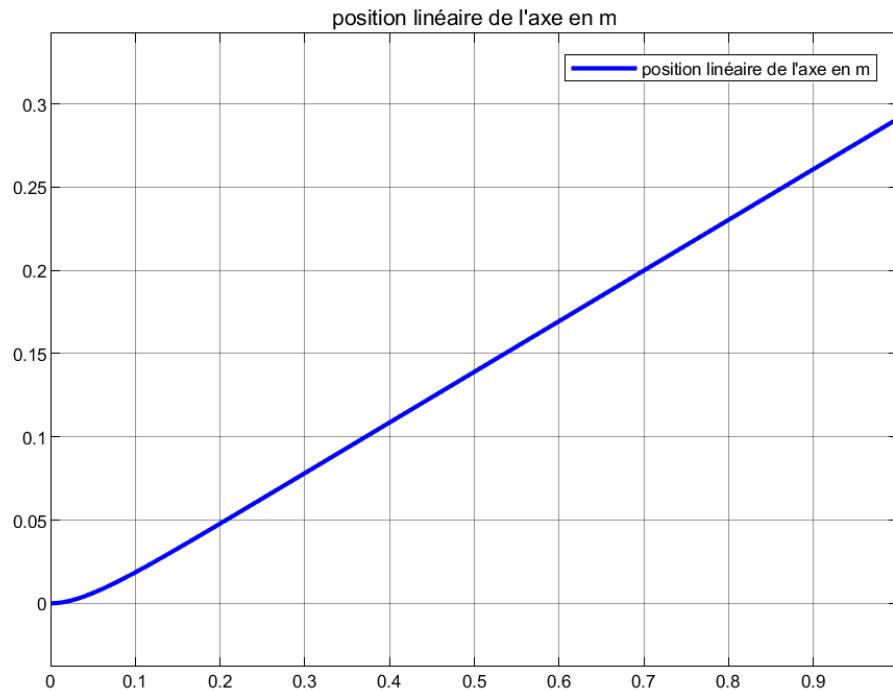


Figure 155 : réponse en position de l'axe linéaire non asservi

Nous pouvons maintenant réaliser le modèle de l'asservissement en position de l'axe représenté sur la Figure 156. Les nouveaux composants ajoutés sont décrits Figure 157.

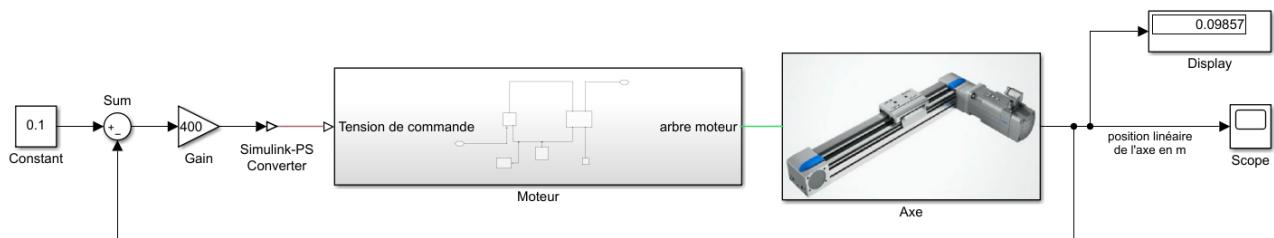


Figure 156 : modèle de l'asservissement en position linéaire de l'axe

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

Seuls sont référencés dans le tableau de la Figure 157, les composants qui n'ont pas encore été utilisés auparavant dans le document.

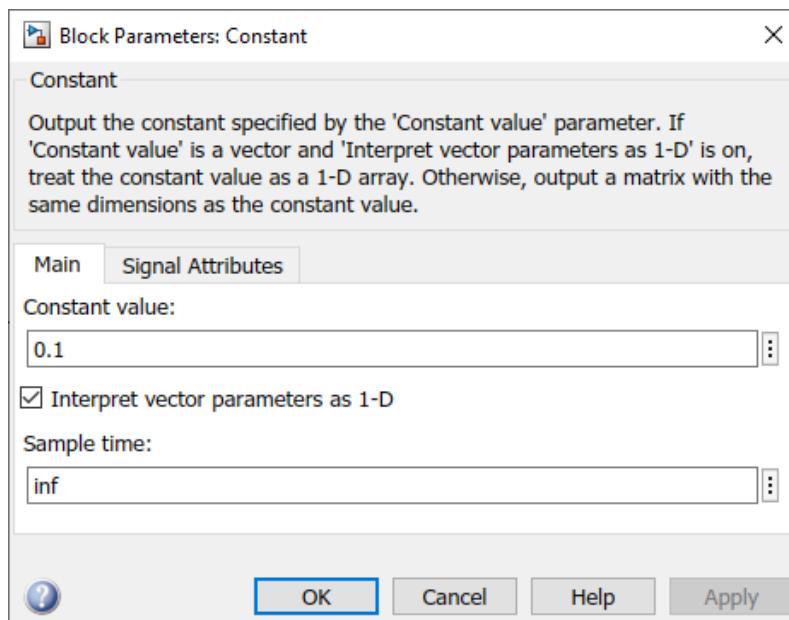
Fonction du composant	Représentation	Bibliothèque
Constante		Simulink/Sources ou Simulink/Commonly used Blocks
Gain		Simulink/Math Operations ou Simulink/Commonly used Blocks
Sommateur		Simulink/Math Operations ou Simulink/Commonly used Blocks
Afficheur		Simulink/Sinks

Figure 157 : composants à ajouter au modèle

Paramétrage



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 succession 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

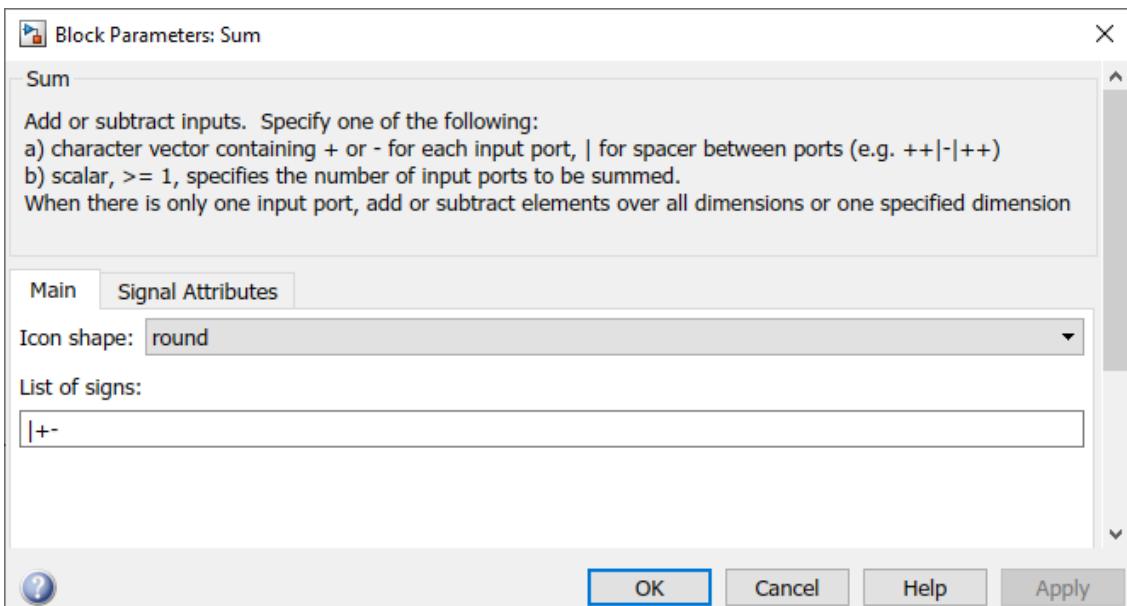
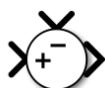
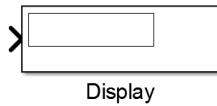


Figure 158 : paramétrage du bloc sommateur

Paramétrage

DISPLAY



Simulink/Sinks

Display

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.

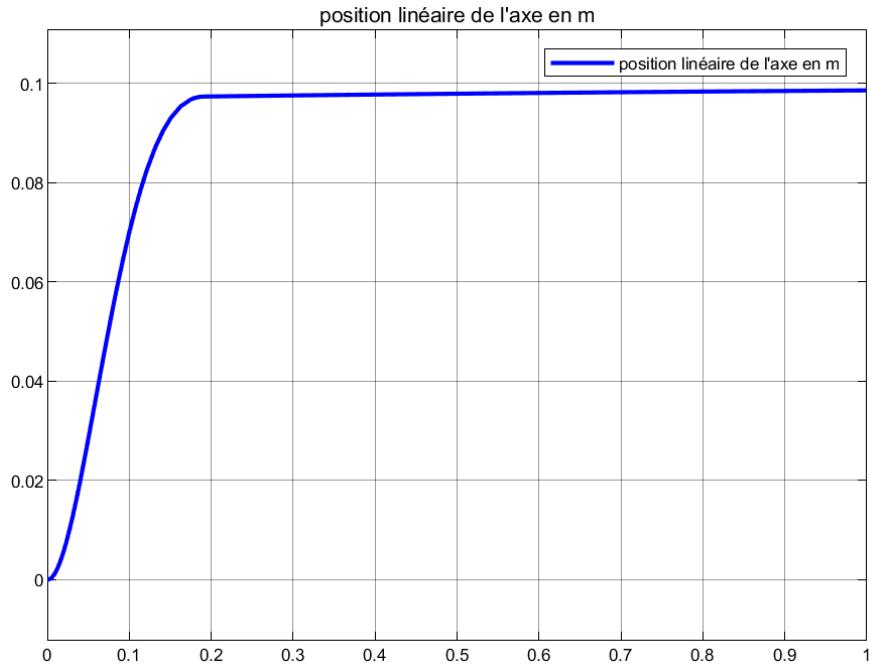


Figure 159 : réponse en position de l'axe linéaire asservi

On remarque que la position finale atteinte par l'axe est de 0.09857 m, ce qui nous permet d'évaluer que l'écart statique est de l'ordre de 2%. 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.

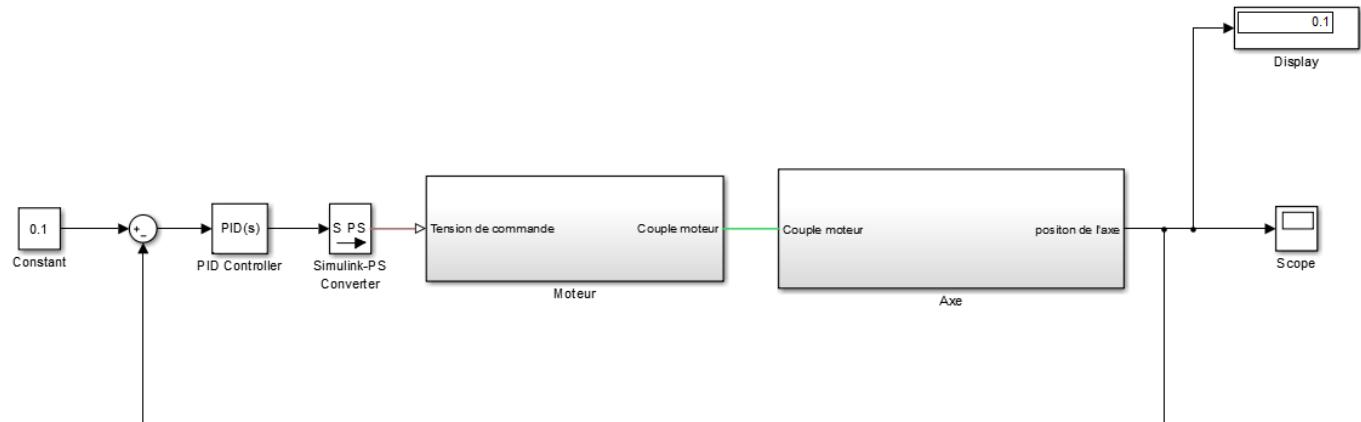
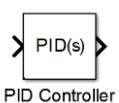


Figure 160 : modèle de l'asservissement en position de l'axe linéaire

Paramétrage

PID Controller



Simulink/Continues

Ce bloc permet de paramétriser un correcteur PID. Nous nous contenterons ici de choisir un correcteur de type **PI** (proportionnel et intégral) avec un gain proportionnel de 400 et un gain intégral de 50 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. Des méthodes permettant de choisir la valeur de ces coefficients sont présentées en fin d'ouvrage.

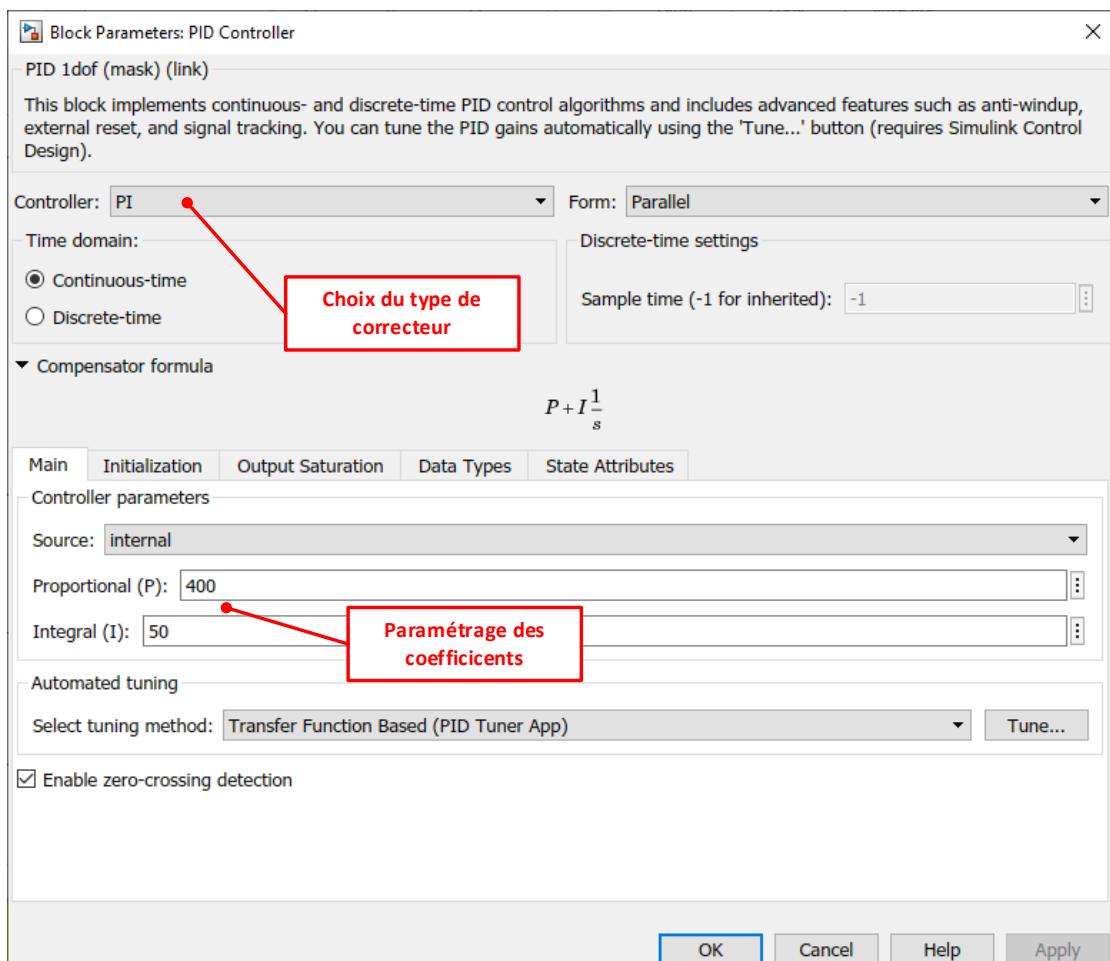


Figure 161 : paramétrage du bloc PID

Lancer la simulation.

Visualiser la réponse en position avec correction proportionnelle et intégrale.

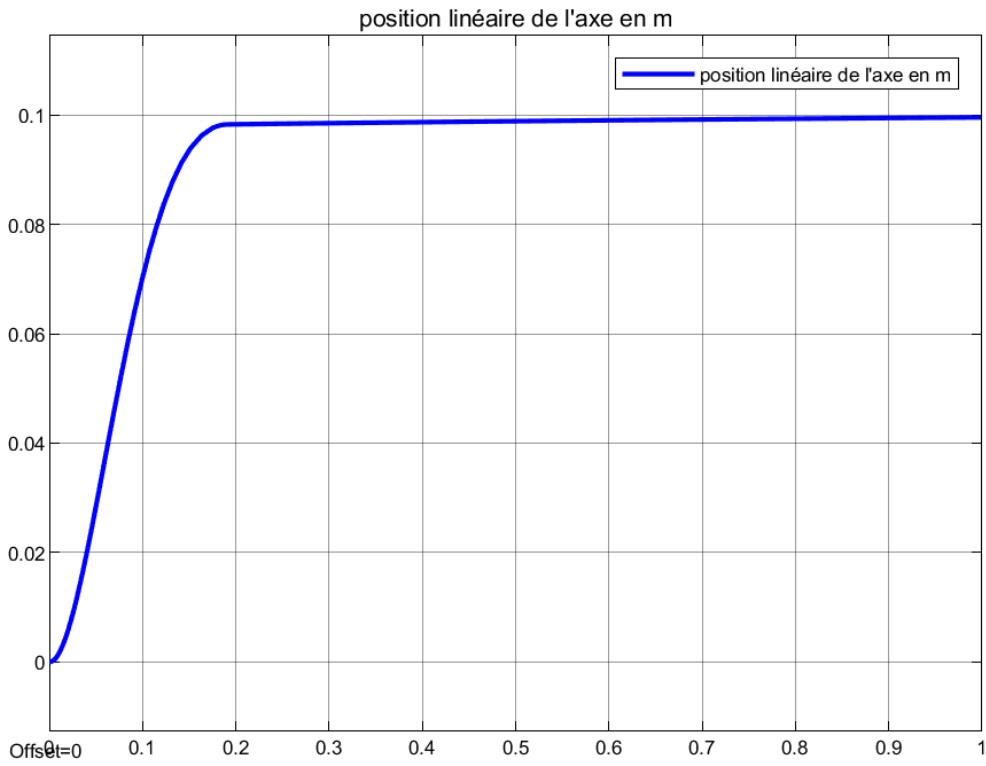


Figure 162 : réponse corrigée de l'asservissement en position de l'axe linéaire

Nous constatons 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 PI pour une consigne de 0.1 m.

B. Domaines hydraulique– 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 163 les composants qui n'ont pas encore été utilisés auparavant dans le document.

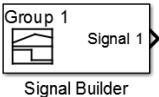
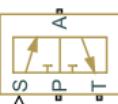
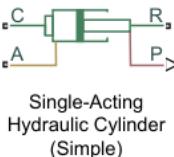
Fonction du composant	Représentation	Bibliothèque
Source de signal paramétrable		Simulink/Sources
Source de pression hydraulique	 Hydraulic Pressure Source	Simscape/Foundation Library/Hydraulic/Sources
Distributeur proportionnel 3/2	 3-Way Directional Valve	Simscape/Fluids/Hydraulics (Isothermal)/Valves/Directional Valves
Référence hydraulique	 Hydraulic Reference	Simscape/Foundation Library/Hydraulic/Hydraulic elements
Propriétés du fluide	 Hydraulic Fluid	Simscape/Fluids/ Hydraulics (Isothermal)/Hydraulic Utilities
Vérin hydraulique simple effet	 Single-Acting Hydraulic Cylinder (Simple)	Simscape/Fluids/ Hydraulics (Isothermal)/Hydraulic Cylinders
Ressort en translation	 Translational Spring	Simscape/Foundation Library/ Mechanical/Translational Elements
amortisseur en translation	 Translational Damper	Simscape/Foundation Library/ Mechanical/Translational Elements
masse	 Mass	Simscape/Foundation Library/ Mechanical/Translational Elements

Figure 163 : les composants nécessaires à la modélisation de la commande d'un vérin hydraulique

Le modèle Simscape du vérin hydraulique simple effet est représenté sur la Figure 164 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.

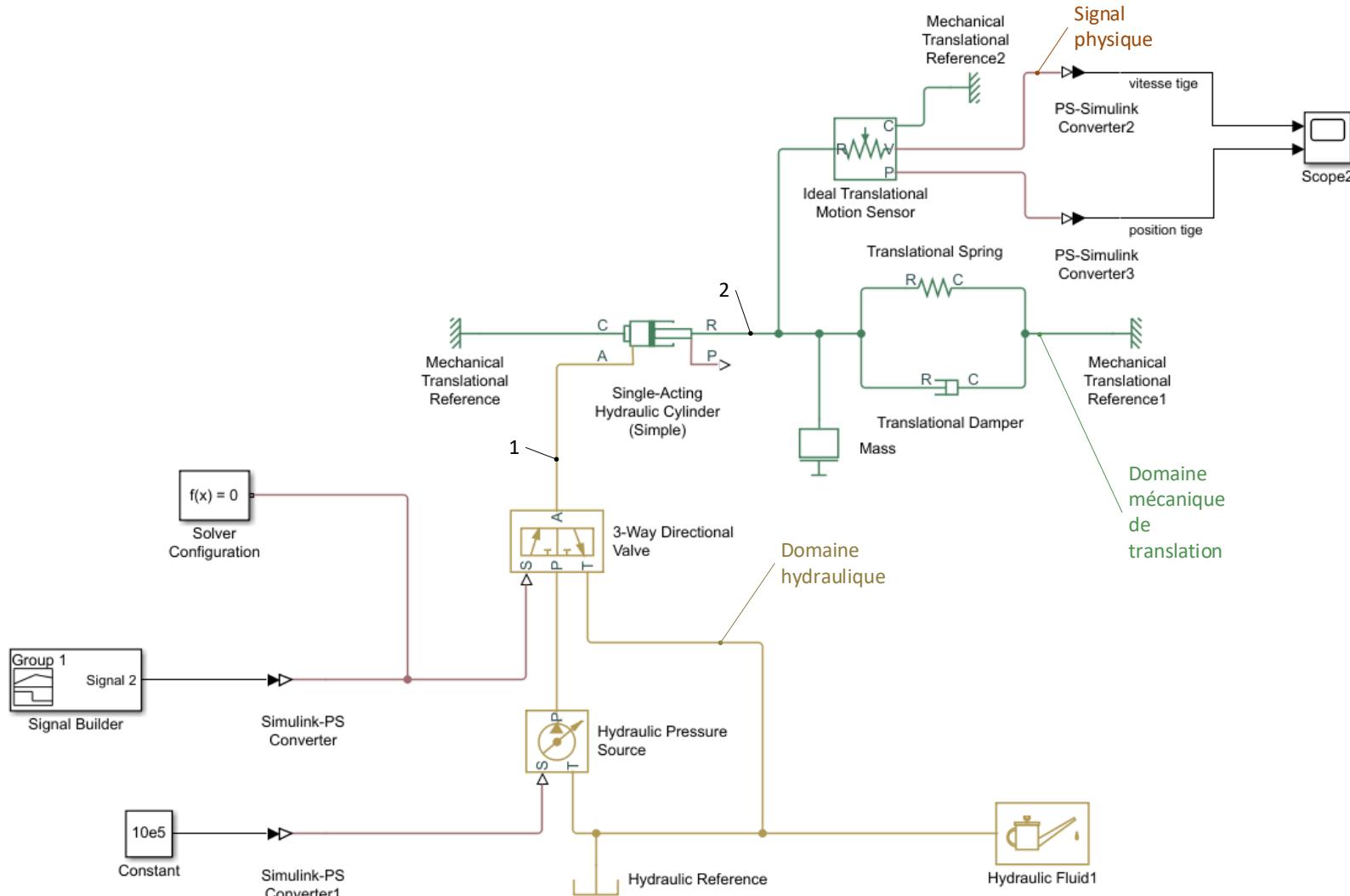


Figure 164 : visualisation des domaines physiques intervenant dans la modélisation de la commande du vérin hydraulique

1. Placement et assemblage des composants

Dans la fenêtre **Simulink Library Browser**, cliquer sur  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 165.

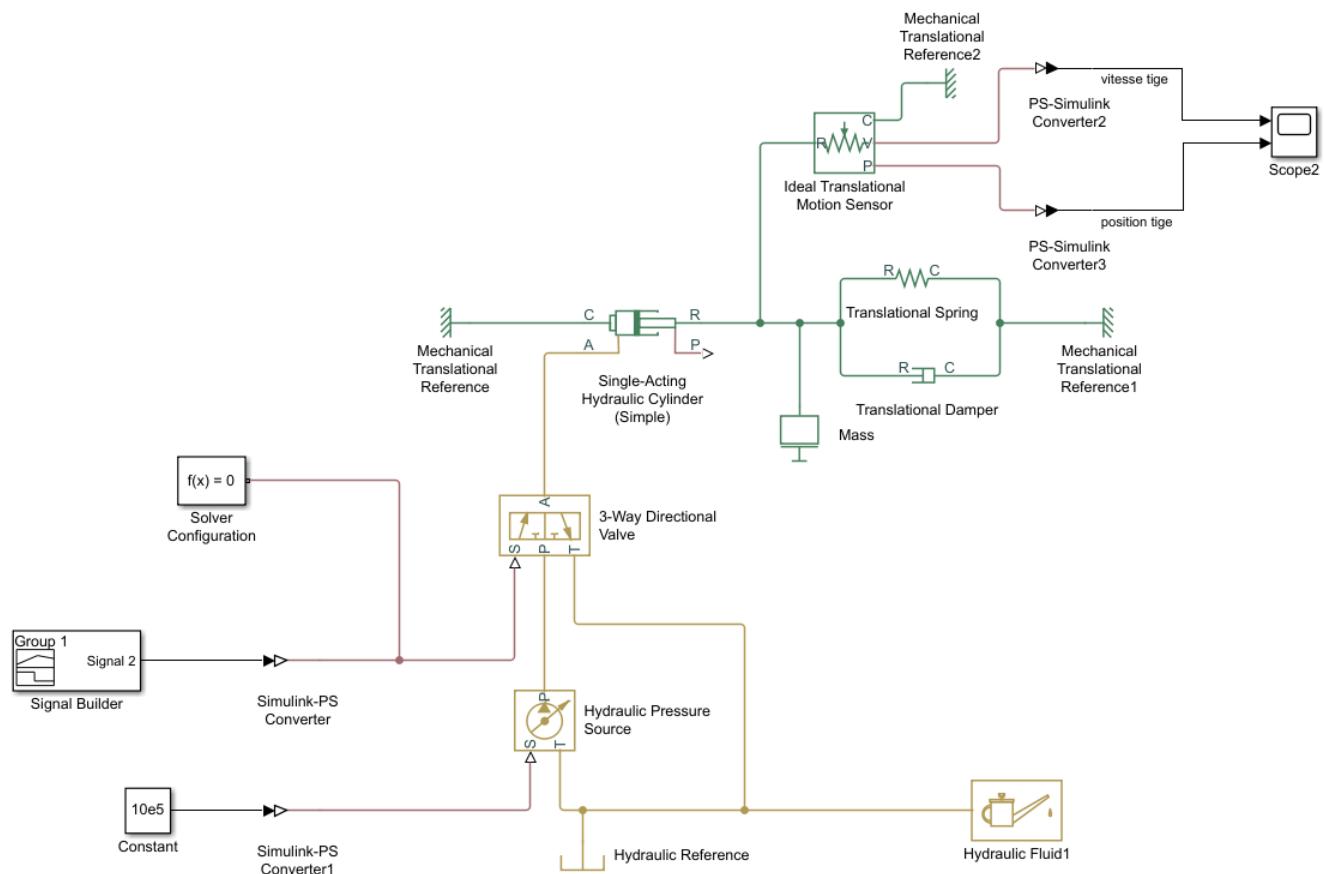


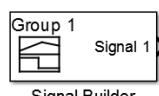
Figure 165 : modèle Simscape de la commande du vérin hydraulique

2. Paramétrage des composants

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

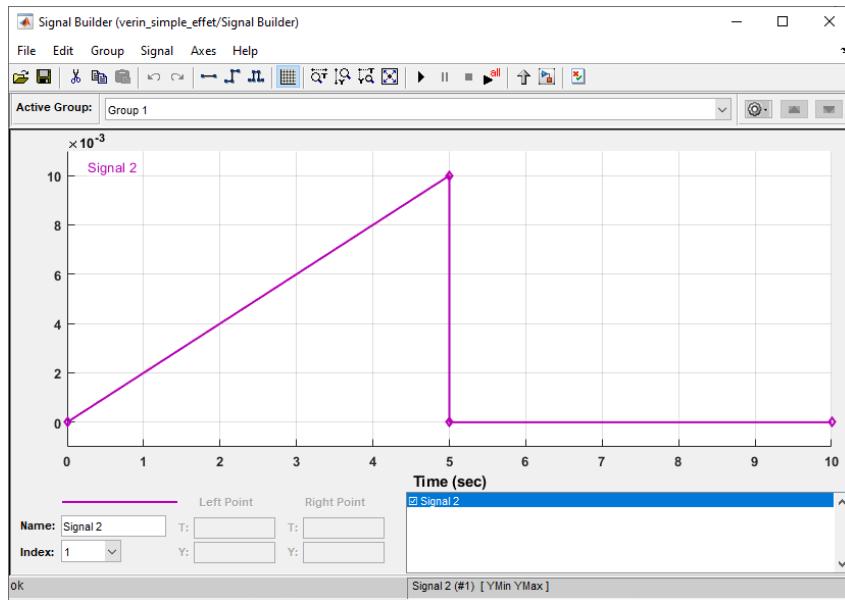
Paramétrage

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 paramétrier le signal.



Paramétrage

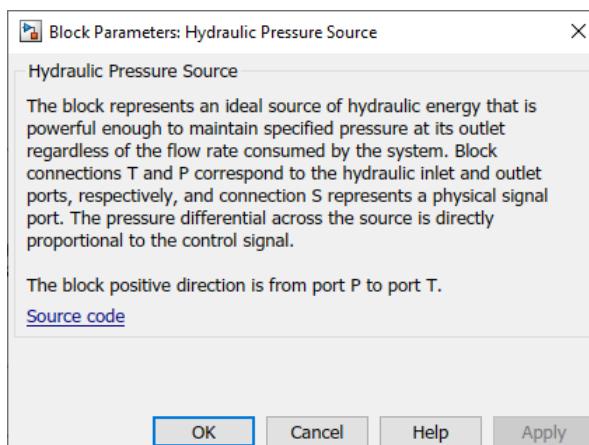
Hydraulic Pressure Source



Hydraulic Pressure Source

Simscape/Foundation 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**.



Paramétrage

3 – Way Directional Valve



3-Way Directional Valve

Simscape/Fluids/Hydraulics
(Isothermal)/Valves/Directional Valves

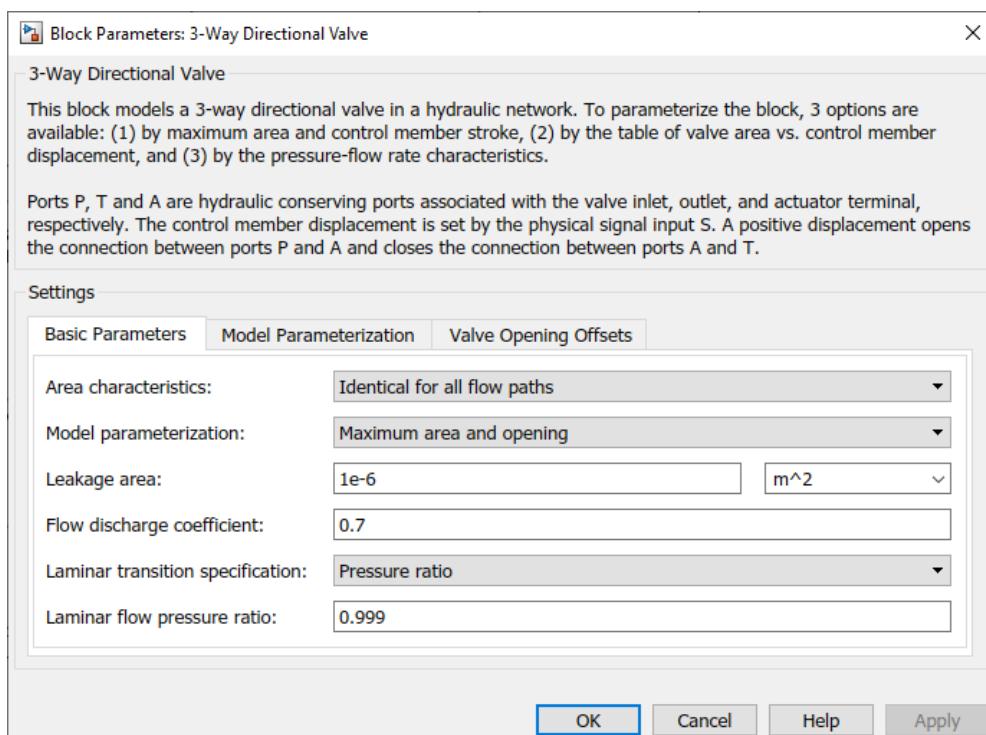
Ce composant permet de modéliser un distributeur hydraulique 3/2 à commande proportionnelle. Il possède 3 ports de type PCP (T, P et A) du domaine hydraulique et un port S de type signal physique qui permet de commander le distributeur.

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

La fenêtre de paramétrage de ce composant possède 3 onglets :

- Basic Parameters
- Model Parametrization
- Valve Openning Offsets

L'onglet **Basics Parameters** permet de spécifier Le paramétrage peut se faire de différentes manières en fonction des données constructeur disponibles. Ici le paramétrage est fait à partir de **By maximum area and opening**.

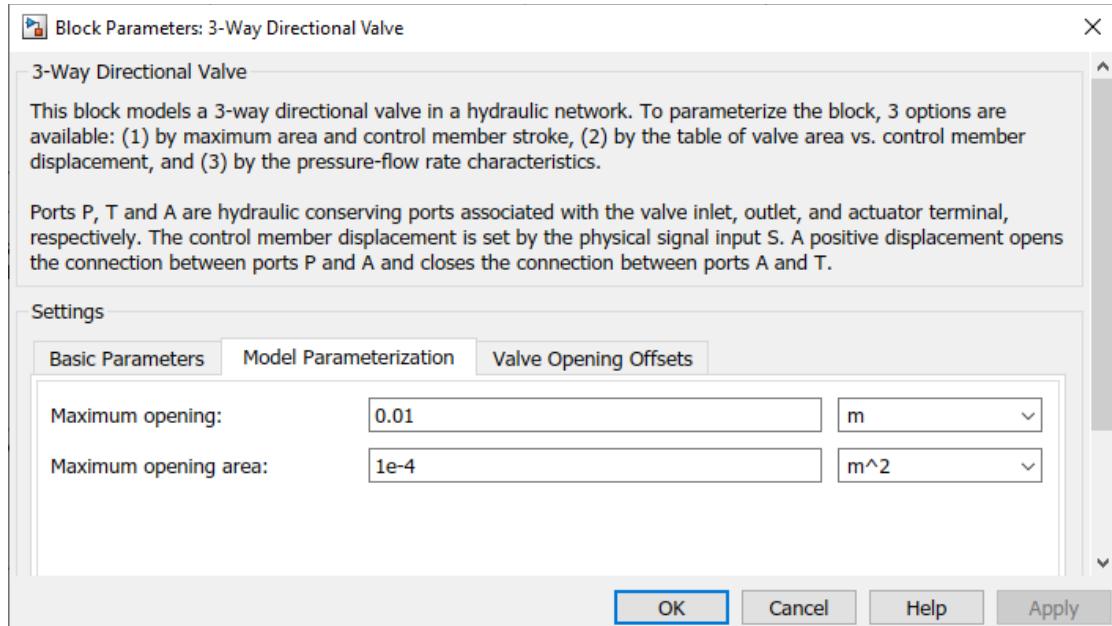


Leakage area : surface correspondant à un léger débit de fuite si la vanne est complètement fermée.

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)

Laminar transition Pressure Ratio : rapport de la pression de sortie sur la pression d'entrée auxquel l'écoulement passe de laminaire à turbulent

L'onglet **Model Parametrization** permet de spécifier les paramètres d'ouverture et fermeture de la valve.



Maximum opening : déplacement maximal de l'axe de commande de la valve

Maximum opening area: diamètre d'ouverture maximal des orifices de la valve

L'onglet **Valves opening Offsets** permet de spécifier les ouvertures initiales des différents orifices (en début de simulation)

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

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

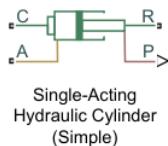
Si on appelle x la commande qui arrive sur le port S, les ouvertures respectives des orifices seront :

$$h_{PA} = h_{PA0} + x$$

$$h_{AT} = h_{AT0} - x$$

Paramétrage

Single-Acting Hydraulic Cylinder (Simple)

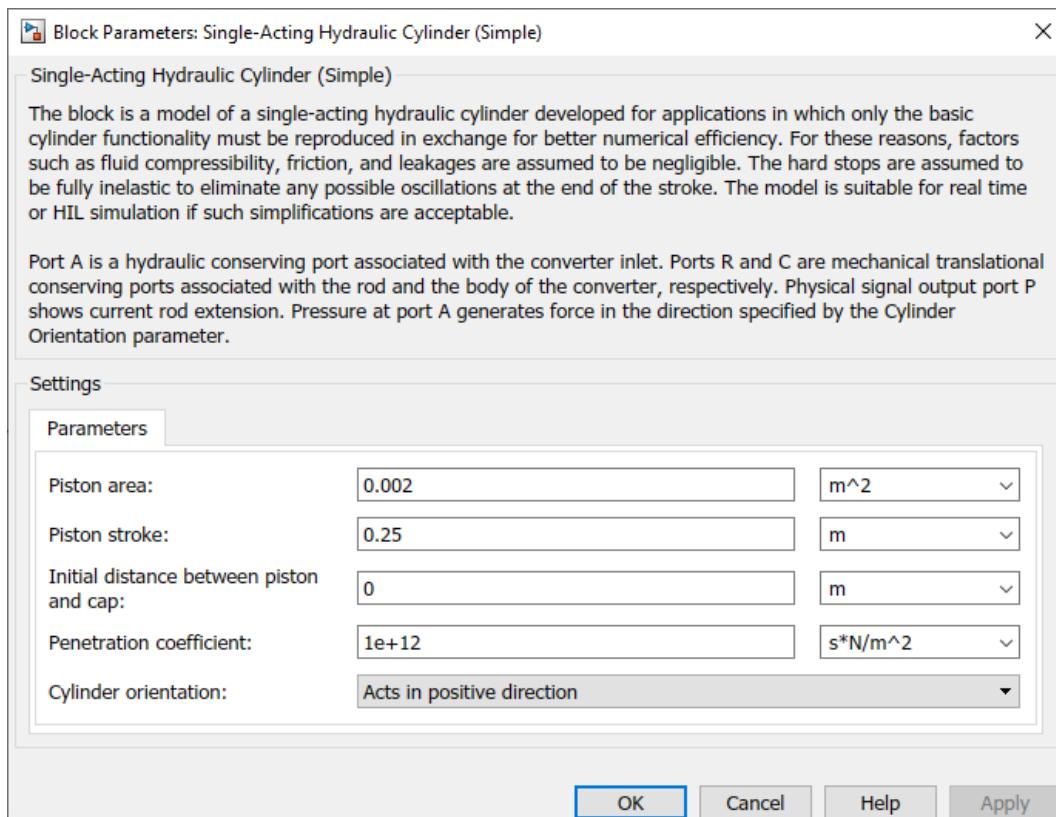


Simscape/Fluids/ Hydraulics
(Isothermal)/Valves/Directional Valves

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

Le port **P** est un signal physique qui représente le déplacement de la tige (dans notre exemple nous utiliserons un capteur et ce port ne sera pas utilisé)



Piston area : surface du piston

Piston stroke : course du piston

Initial distance between piston and 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

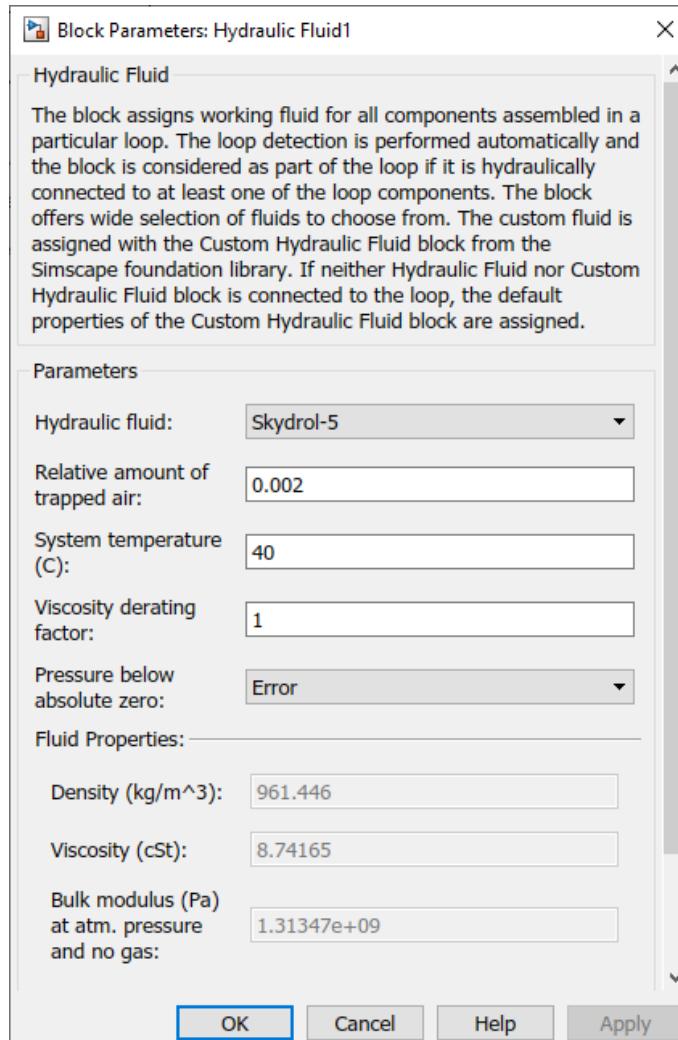
Hydraulic fluid



Simscape/Fluids/ Hydraulics
(Isothermal)/Hydraulic Utilities

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.



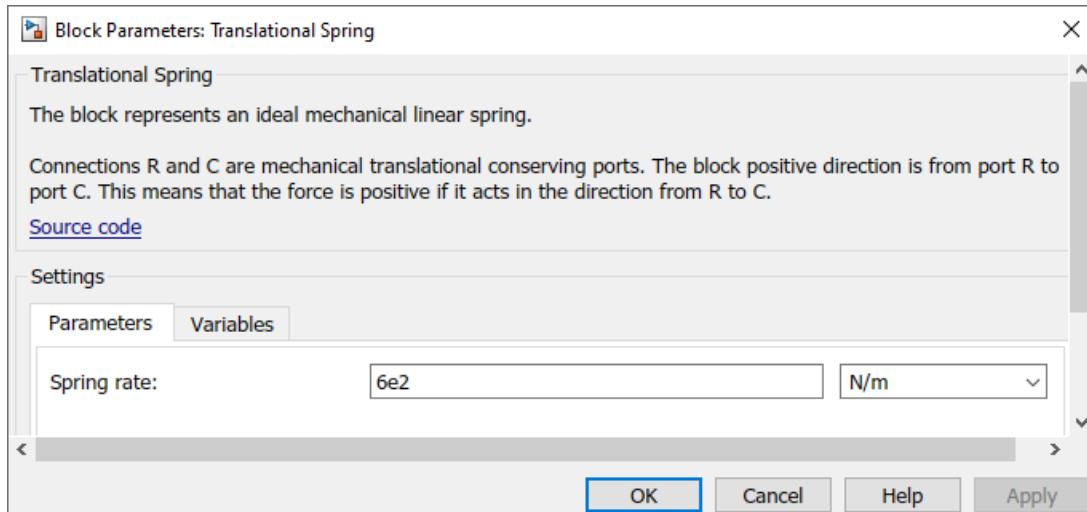
Paramétrage

Translational Spring



Simscape/Foundation Library/
Mechanical/Translational Elements

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



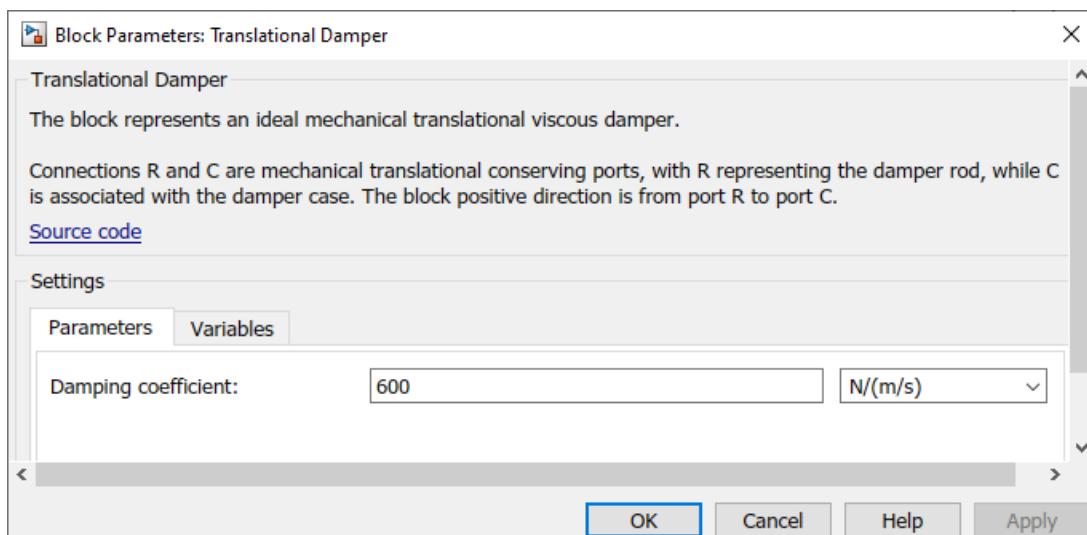
Paramétrage

amortisseur en
translation



Simscape/Foundation Library/
Mechanical/Translational Elements

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



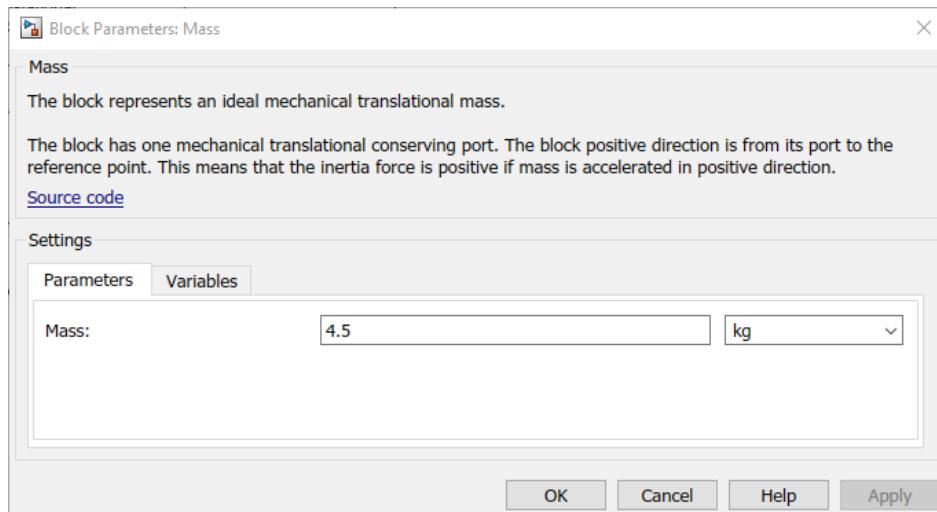
Paramétrage

Mass



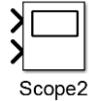
Simscape/Foundation Library/
Mechanical/Translational Elements

Le paramétrage de ce bloc se limite à indiquer la valeur de la masse.



Paramétrage

Scope



Simulink/Sinks

Pour paramétrier 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**.

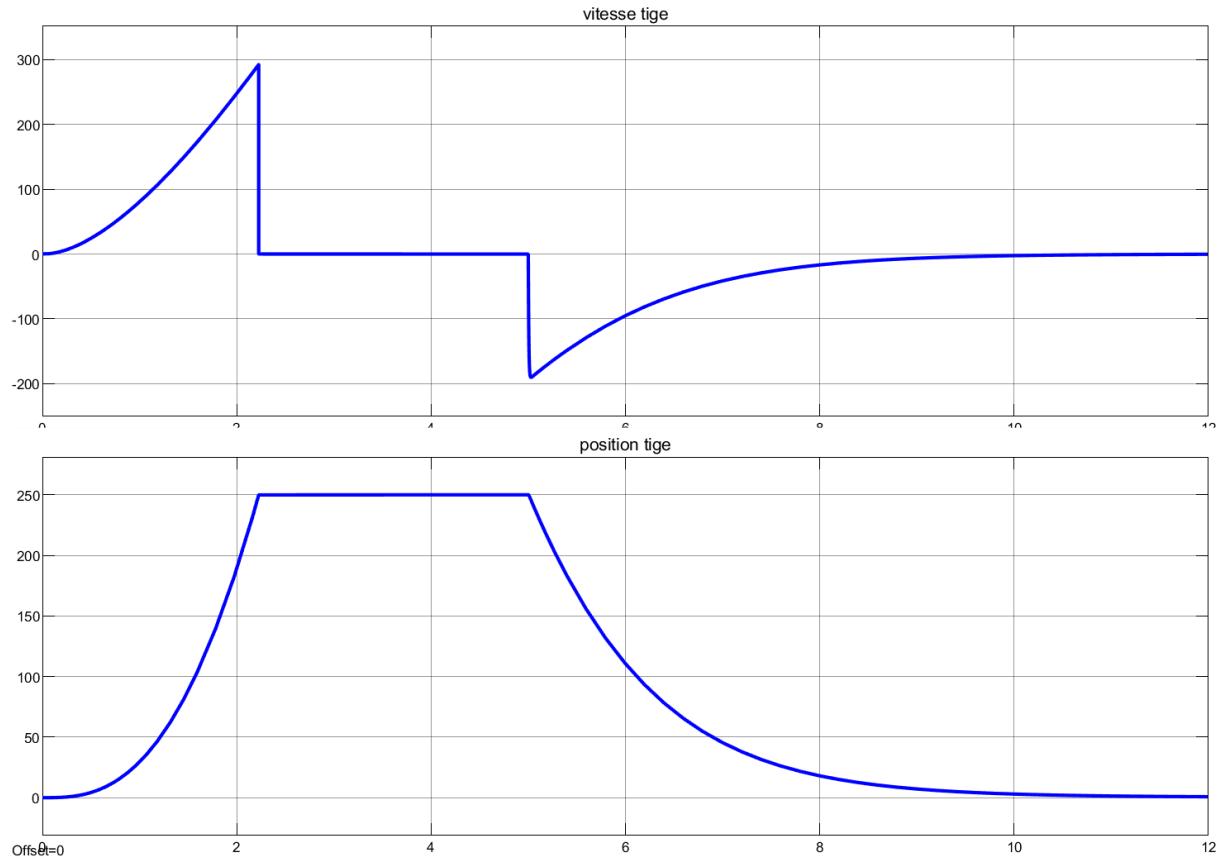


Figure 166 : évolution de la vitesse et de la position de la tige du vérin

Nous observons sur la Figure 166 que la tige du vérin sort, jusqu'à atteindre sa course maximale à $t=2.3$ s. La tige est à l'arrêt jusqu'à la disparition de la commande à $t=5$ s. La tige revient ensuite 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**.

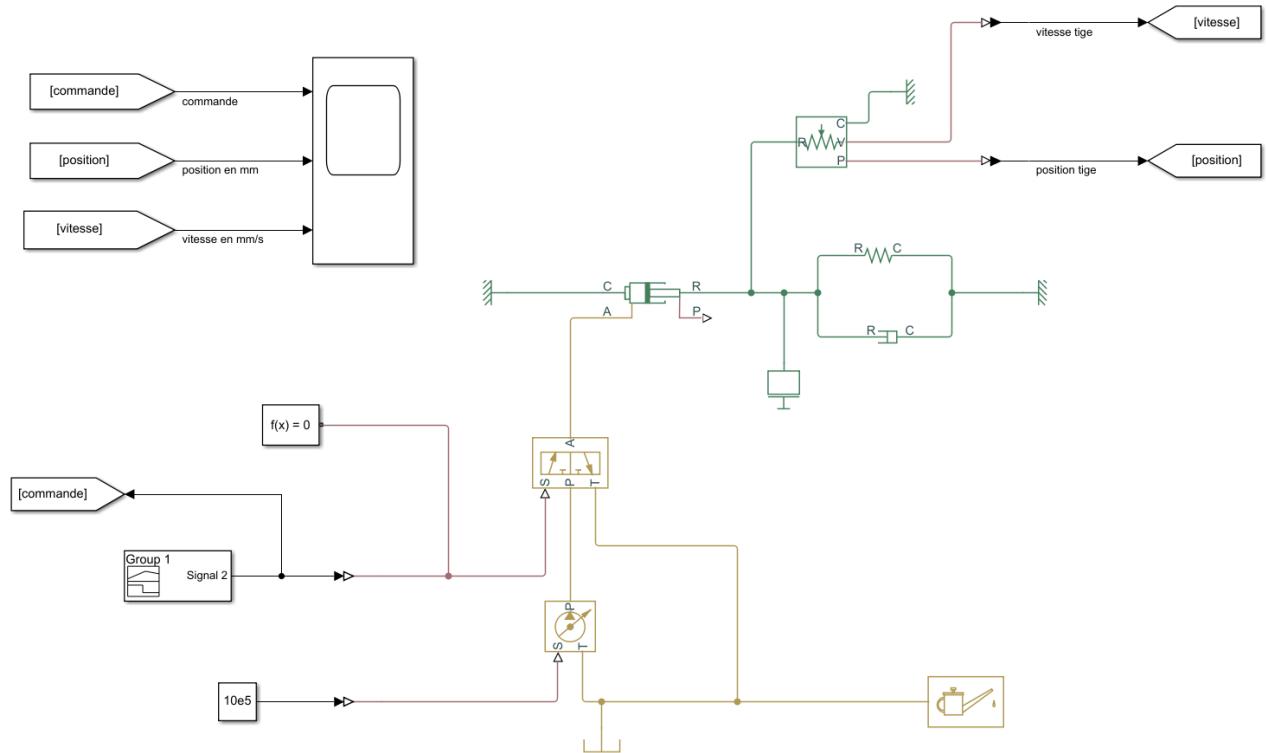
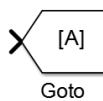


Figure 167 : modèle Simscape de la commande du vérin hydraulique avec routage des signaux

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**.

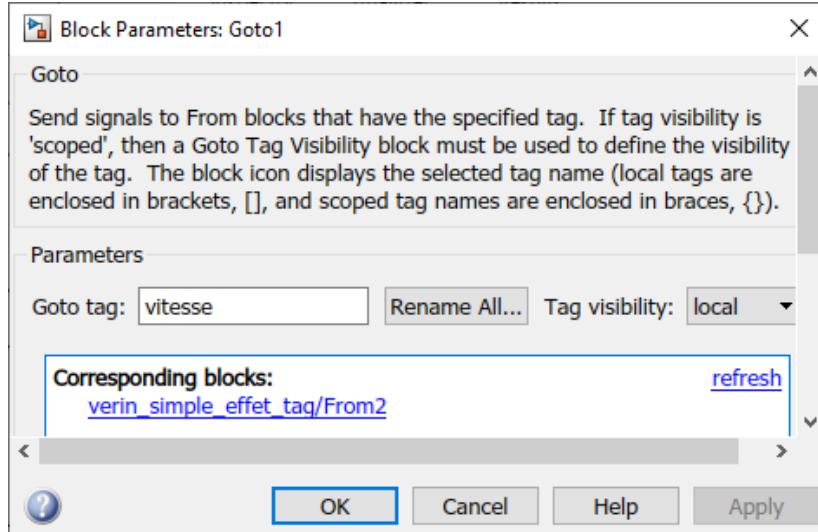
Paramétrage

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**.

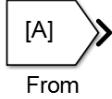


Goto tag : indique le nom que l'on donne au signal

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)

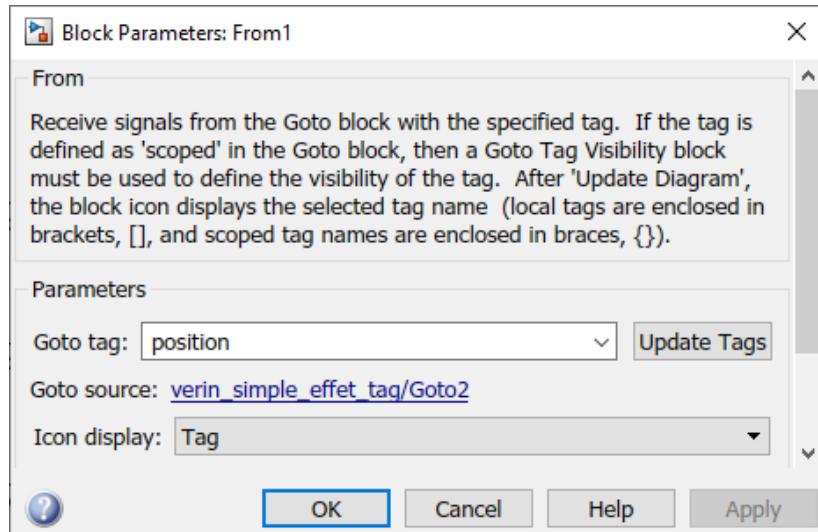
Paramétrage

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.

En double-cliquant sur le scope, il est possible de visualiser l'ensemble des signaux.

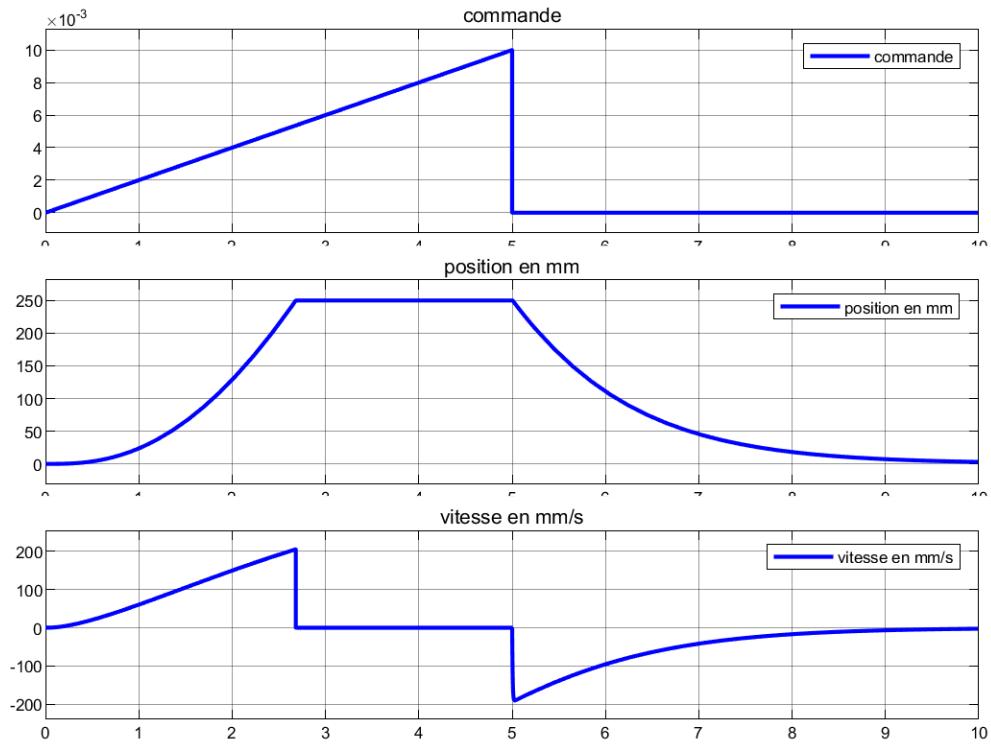


Figure 168 : visualisation des résultats de la simulation

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 169).

Ouvrir le fichier **verin_simple_effet_débit.slx**.

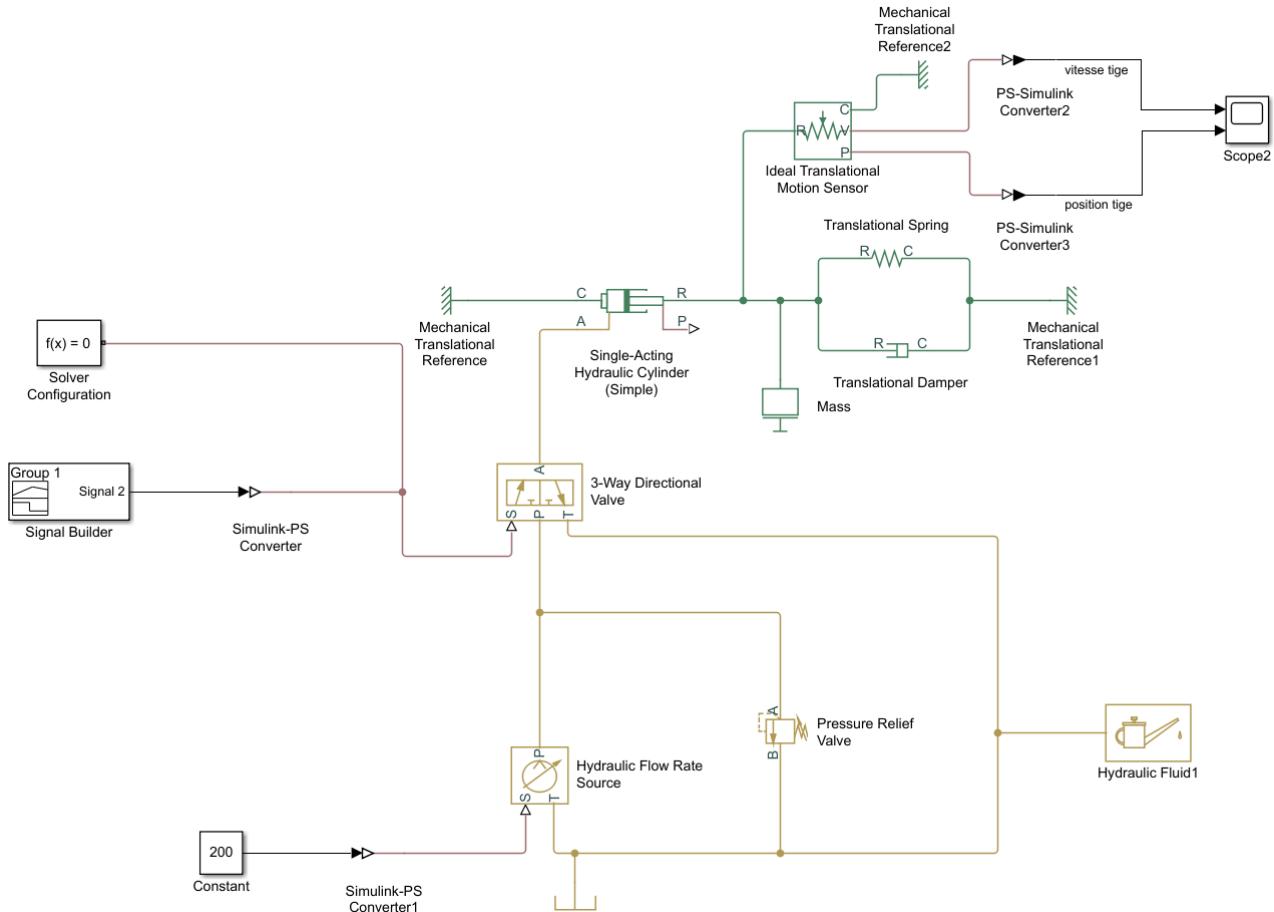


Figure 169 : modèle Simscape de la commande d'un vérin hydraulique avec source de débit

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



Hydraulic Flow Rate Source1

Simscape/Foundation 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**.

Paramétrage

Pressure Relief Valve



Simscape/Fluids/Hydraulics(Isothermal)/Valves/Pressure Control Valves

Ce composant modélise un limiteur de pression.

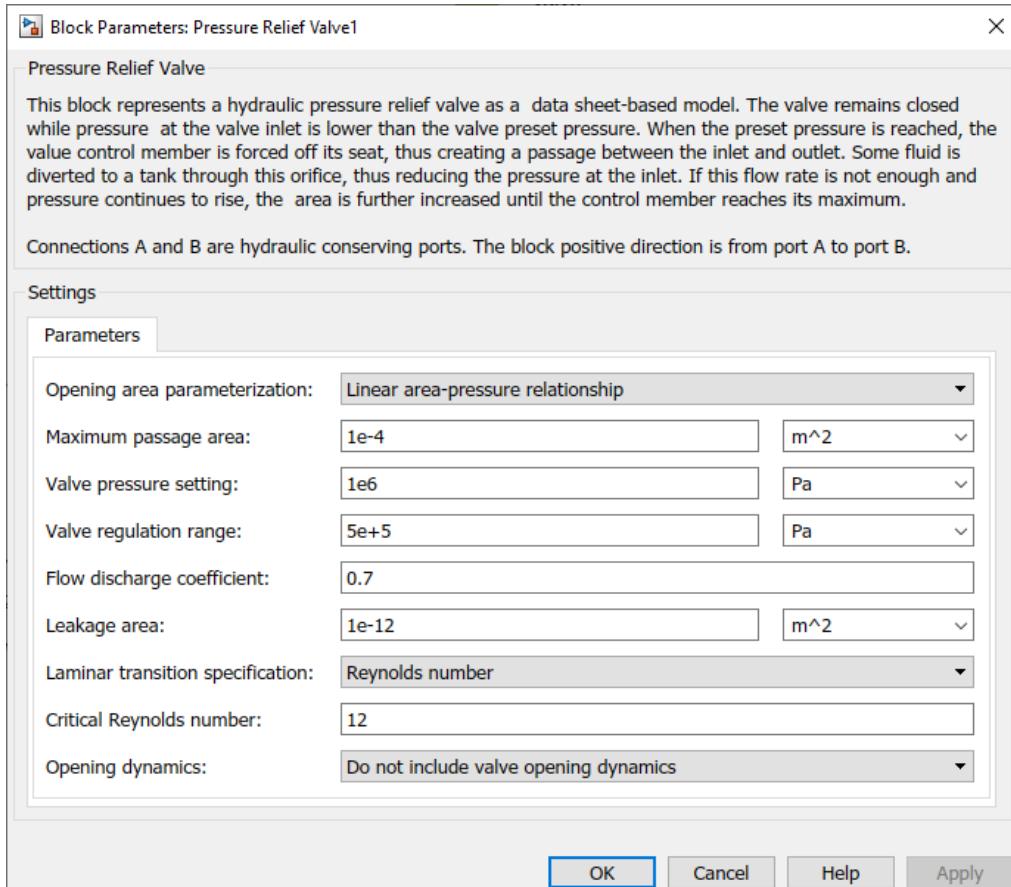


Figure 170 : paramétrage d'un bloc limiteur de pression

Maximum passage area : indique la surface maximale de passage du fluide

Valve pressure setting (p_{set}) : indique la valeur de la pression à partir de laquelle la valve commence à s'ouvrir.

Valve regulation range (p_{reg}) : indique la variation de pression dans la valve entre la position ouverte et la position fermée.

La courbe de la Figure 171 montre la variation de l'ouverture de la valve en fonction de la pression.

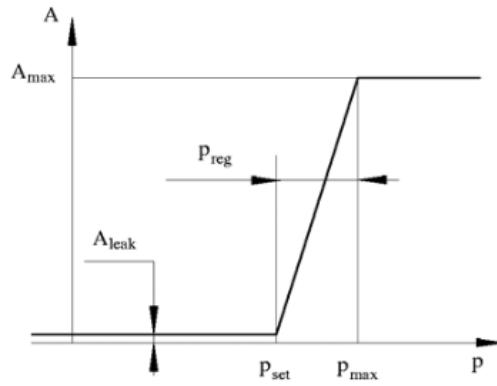


Figure 171 : variation de l'ouverture de soupape en fonction de pression

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

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

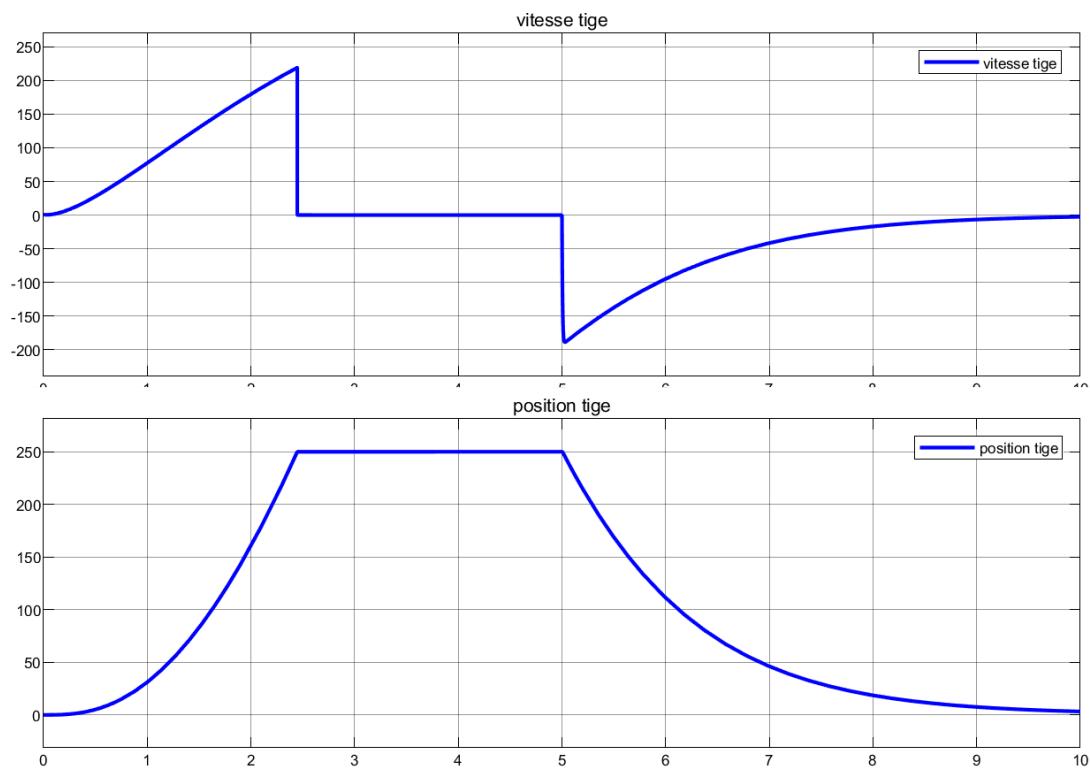


Figure 172 : évolution de la vitesse et de la position de la tige du vérin

On observe que la tige met environ 2.3 s pour atteindre la butée.

Augmenter la valeur de la **Valve pressure setting** dans le limiteur de pression **5e6 Pa** et relancer la simulation.

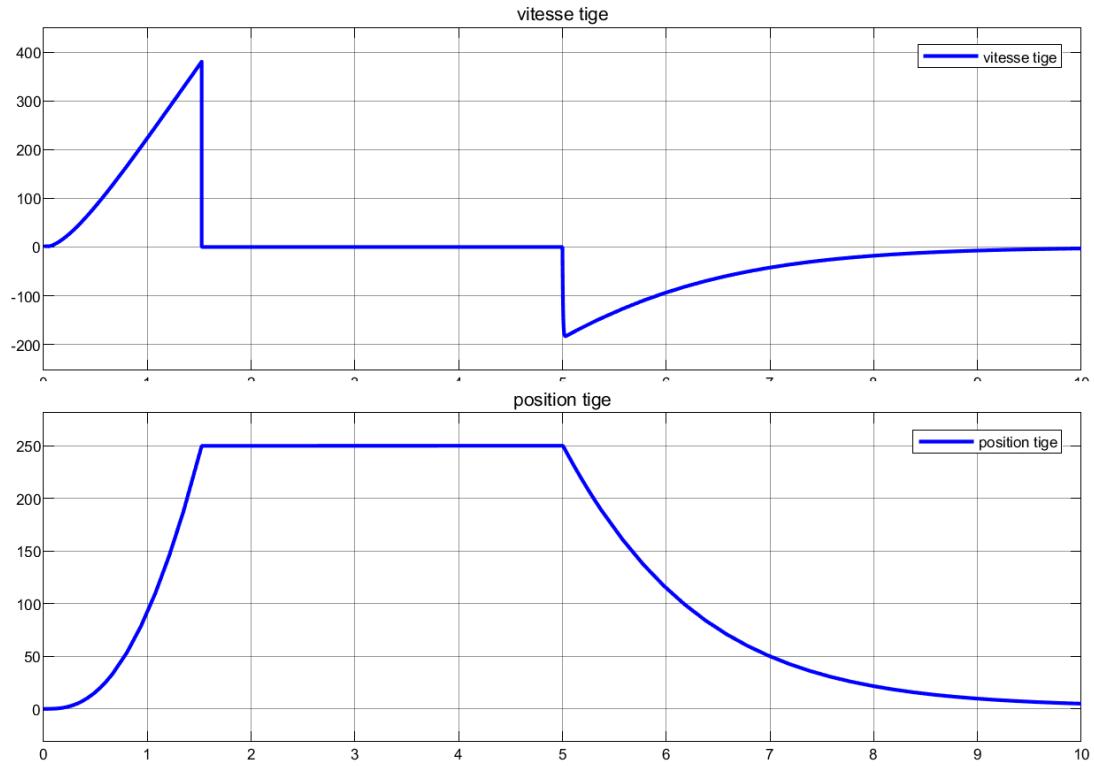


Figure 173 : évolution de la vitesse et de la position de la tige du vérin

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 1.5 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.

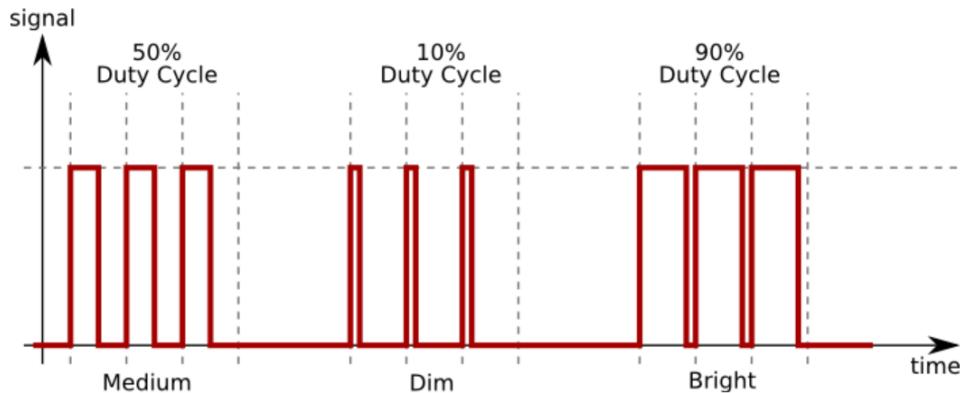


Figure 174 : principe de commande PWM

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

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 **Electrical** de Simscape :

Fonction du composant	Représentation	Bibliothèque
Génération d'une tension de commande de type PWM		Simscape/Electrical/Integrated Circuits
Pont en H		Simscape/Electrical/Semiconductors&Converters/Converters

1. Utilisation du composant « Controlled PWM Voltage »

Ouvrir le fichier **bloc_PWM.slx**.

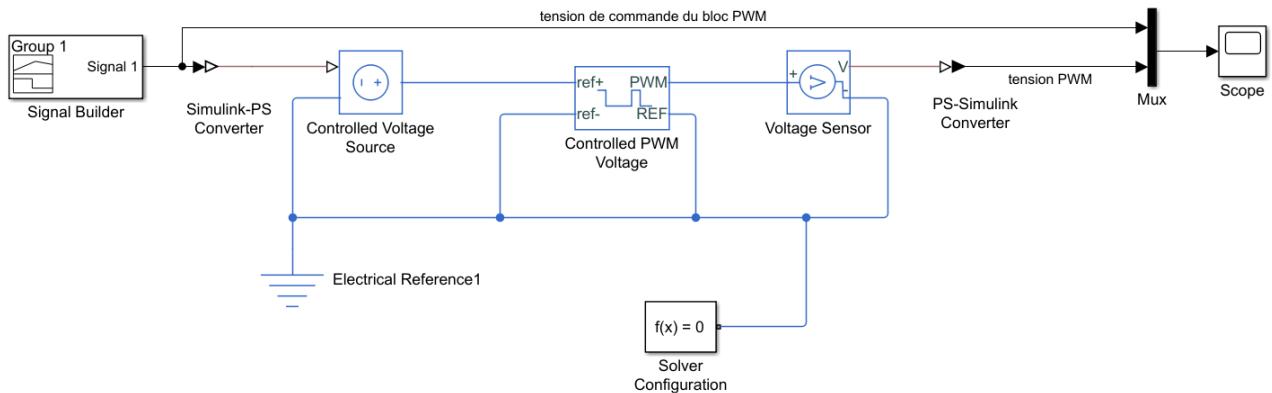


Figure 175 : Illustration du fonctionnement du bloc « Controlled PWM Voltage »

Double-cliquer sur le **Signal Builder** pour observer l'allure de la tension imposée au port +ref (Figure 176).

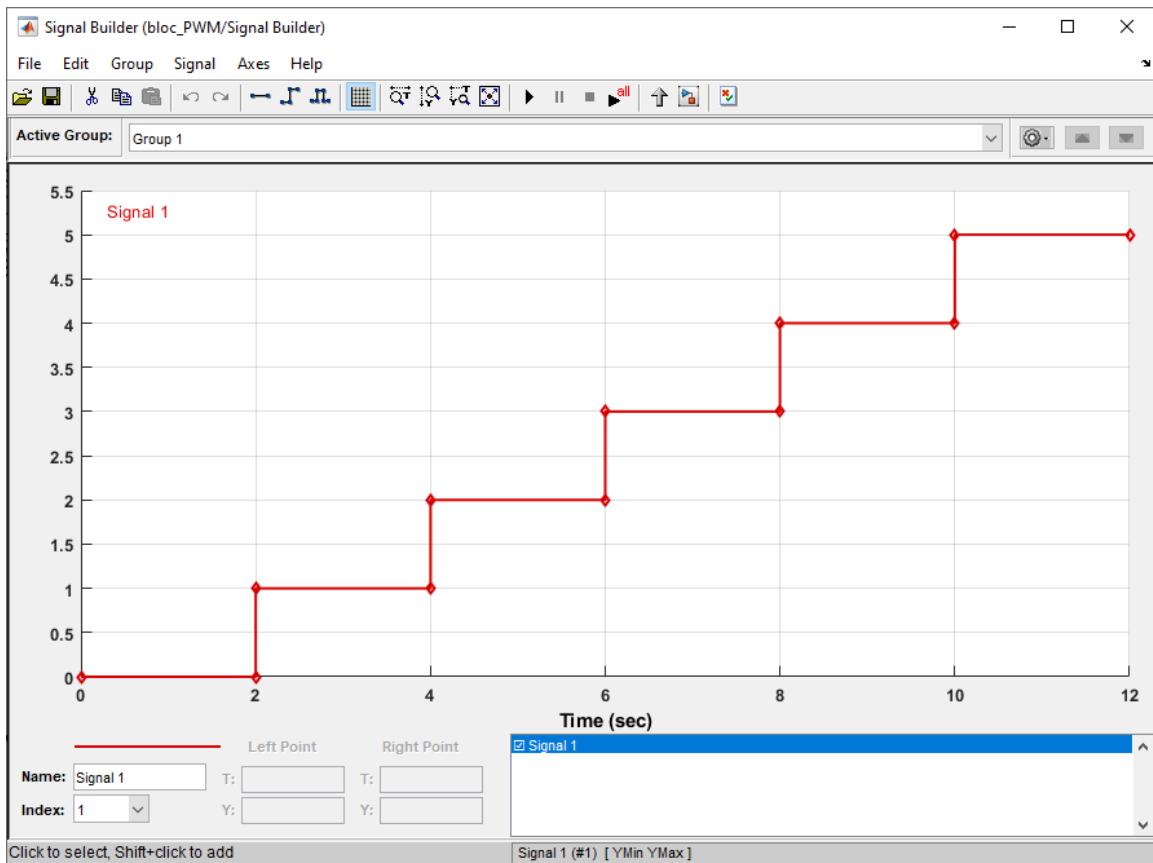


Figure 176 : tension de commande du bloc Controlled PWM Voltage

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.

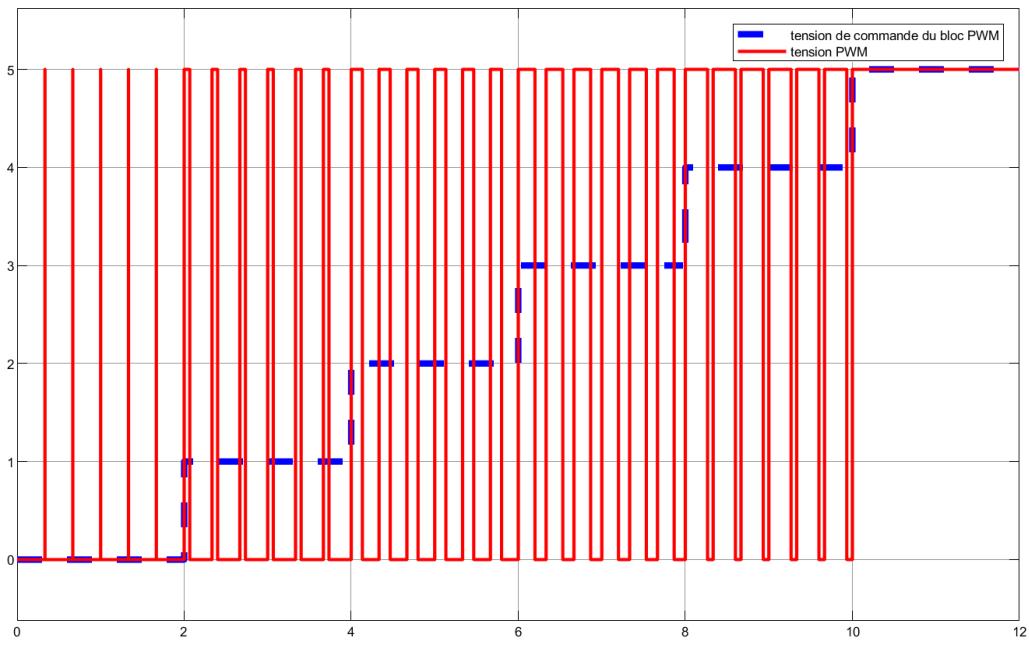


Figure 177 : signal PWM en fonction de la tension de commande

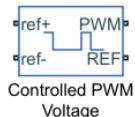
En bleu apparaît le signal de commande imposé sur le port **+ref**, en rouge la tension PWM qui sort sur le port **PWM**.

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



Simscape/Electronics/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** : borne + du bloc PWM
- **-ref** : borne - du bloc PWM
- **PWM** : signal de tension PWM dont le rapport cyclique est donné par $\alpha = 100 \times \left(\frac{V_{ref} - V_{min}}{V_{max} - V_{min}} \right)$ avec $V_{ref} = ref_+ - ref_-$. 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)

Le paramétrage de ce composant nécessite de renseigner plusieurs onglets :

- PWM
- Input Scaling
- Output Voltage

Paramétrage de l'onglet PWM :

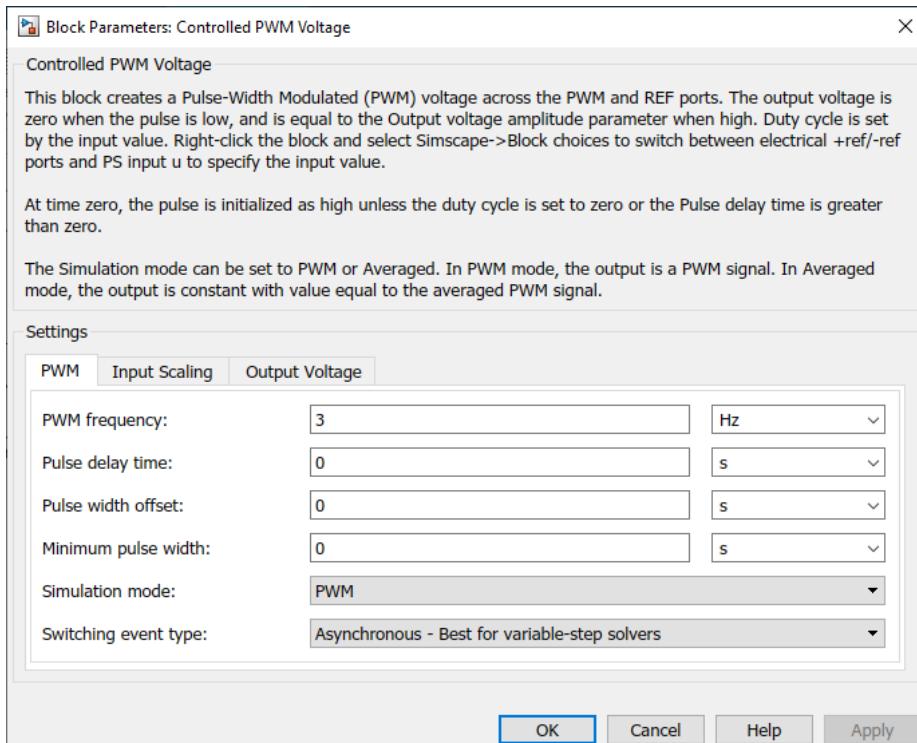


Figure 178 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage

PWM frequency : fréquence du signal 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

Minimum Pulse Width : largeur d'impulsion minimale

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.

Paramétrage de l'onglet Input Scaling :

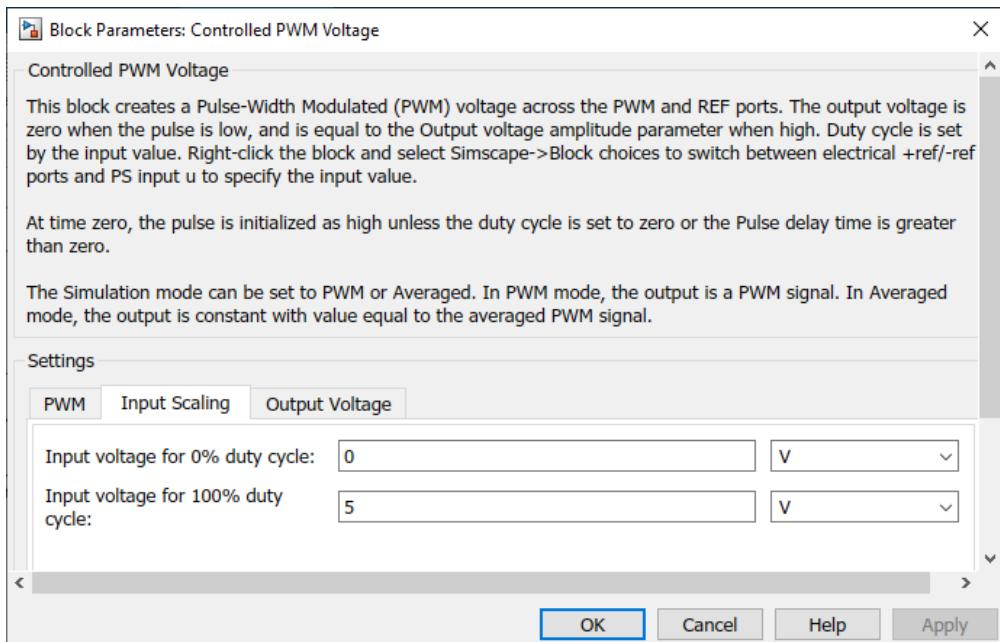


Figure 179 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage

Input value Vmin for 0% duty cycle : valeur de la tension V_{ref} qui donne un rapport cyclique de 0%.

Input value Vmax for 100% duty cycle : valeur de la tension V_{ref} qui donne un rapport cyclique de 100%.

Paramétrage de l'onglet Output Voltage

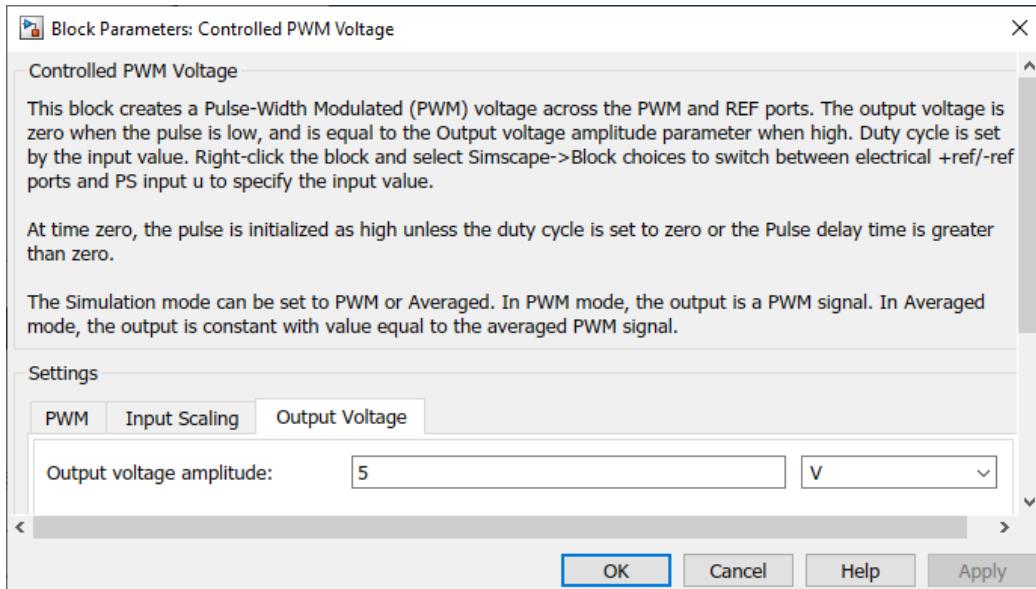


Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage

Output voltage amplitude : amplitude maximale de la tension de sortie

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**.

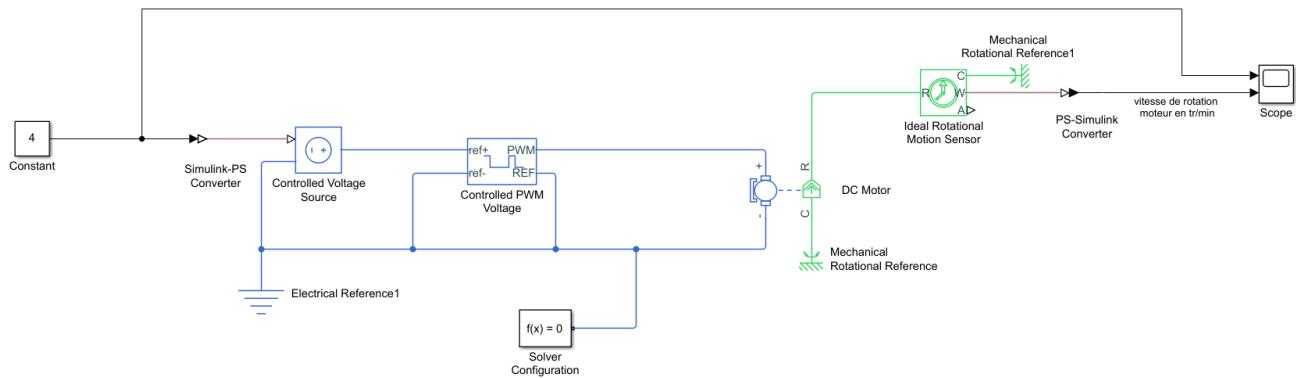


Figure 181 : commande PWM d'un moteur à courant continu

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 182).

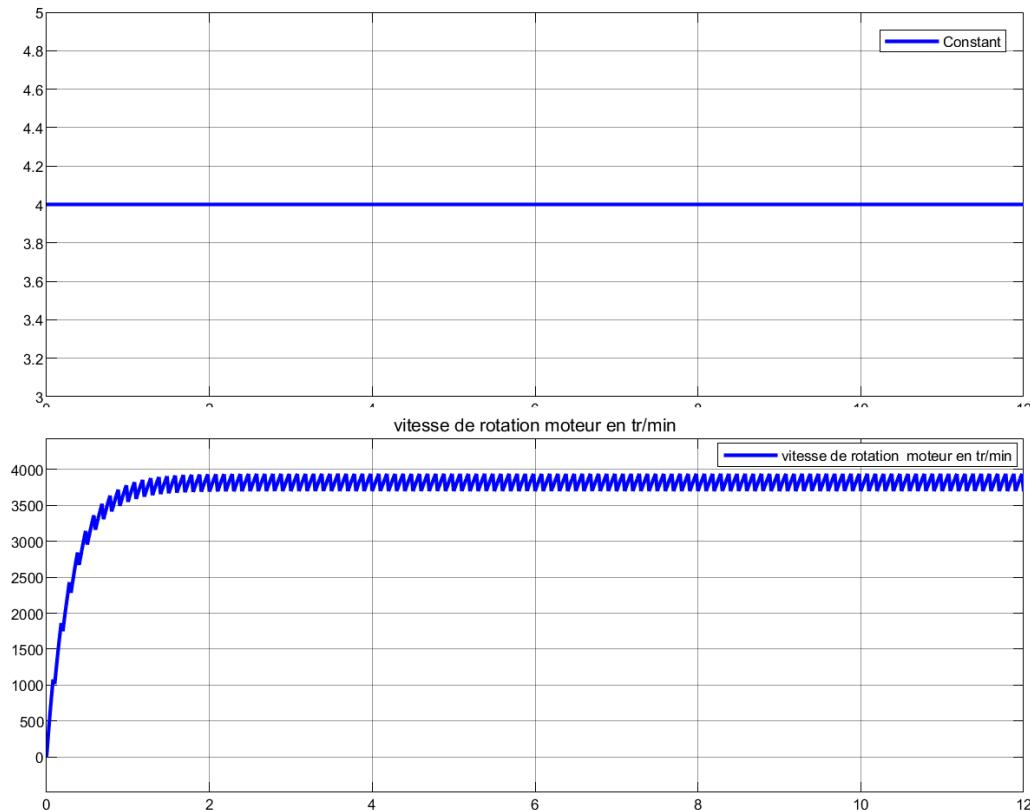


Figure 182 : vitesse de rotation du moteur

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. Le calcul va prendre plus de temps dans la mesure ou la fréquence élevée impose un petit pas de calcul pour le solveur.

Observer la vitesse de rotation du moteur (Figure 183).

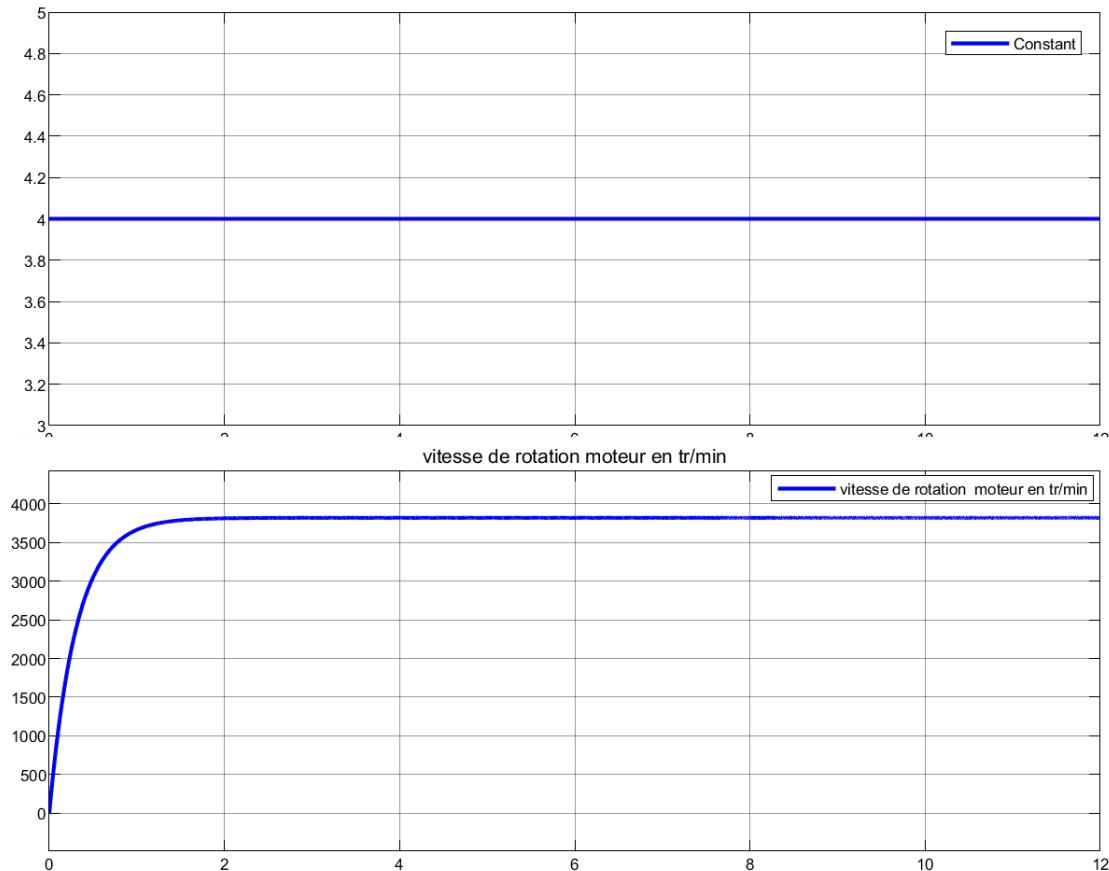


Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM

Nous remarquons sur la Figure 183 que la réponse en vitesse est maintenant plus lisse et l'influence de la commande PWM est moins visible.

Nous constatons 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 sera strictement identique à celui de la Figure 183. Nous avons pu constater le gain de temps dans la résolution du modèle par le solveur.

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**.

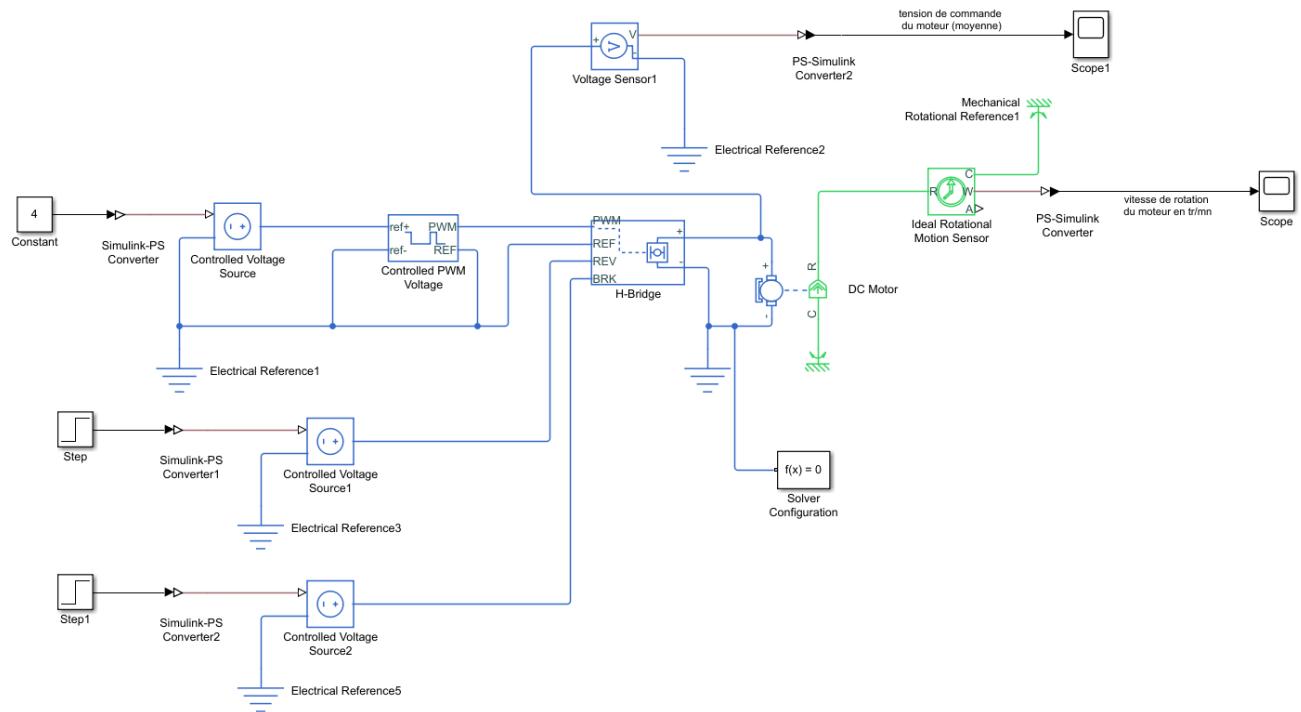


Figure 184 : commande PWM d'un moteur à courant continu avec pont en H

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 185).

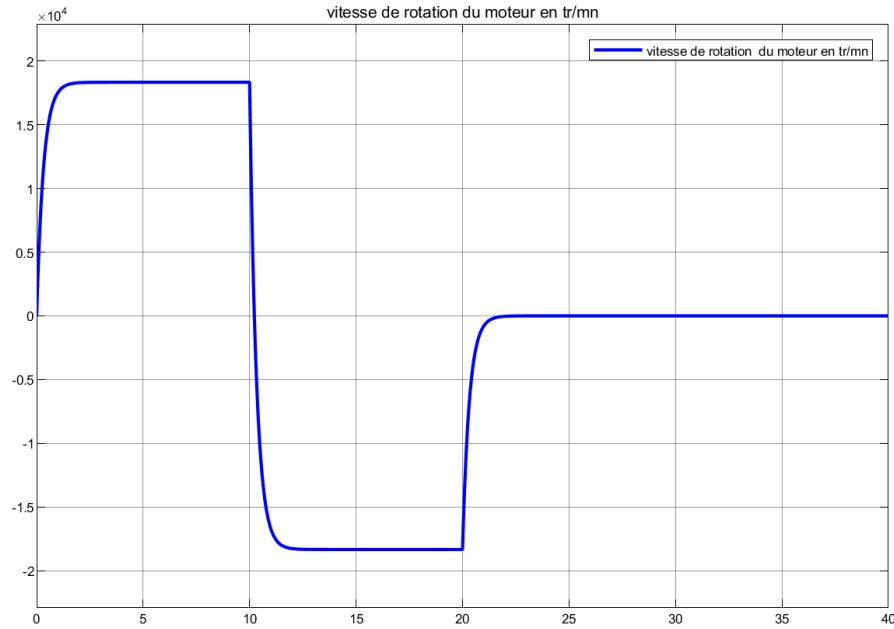


Figure 185 : vitesse de rotation du moteur commandé par le pont en H

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.

Ouvrir le scope correspondant à la tension de commande du moteur (Figure 186).

Nous remarquons 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.

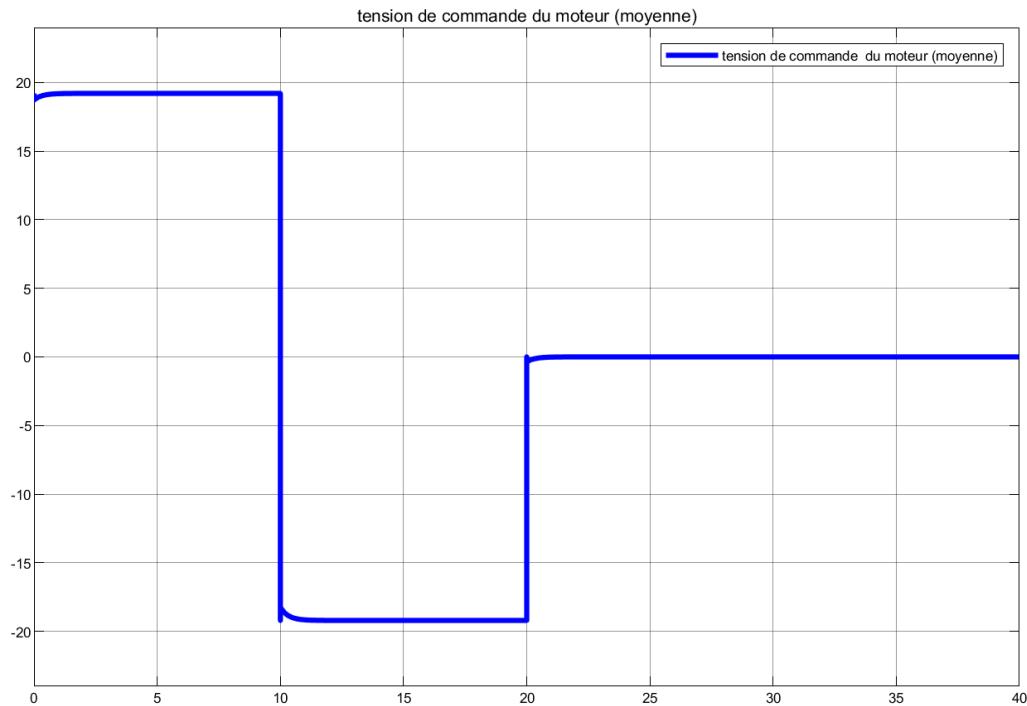
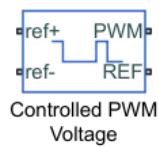


Figure 186 : tension moyenne d'alimentation du moteur

Paramétrage

Controlled PWM Voltage



Simscape/Electronics/Actuators and Drivers/Drivers

Paramétrage de l'onglet PWM :

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

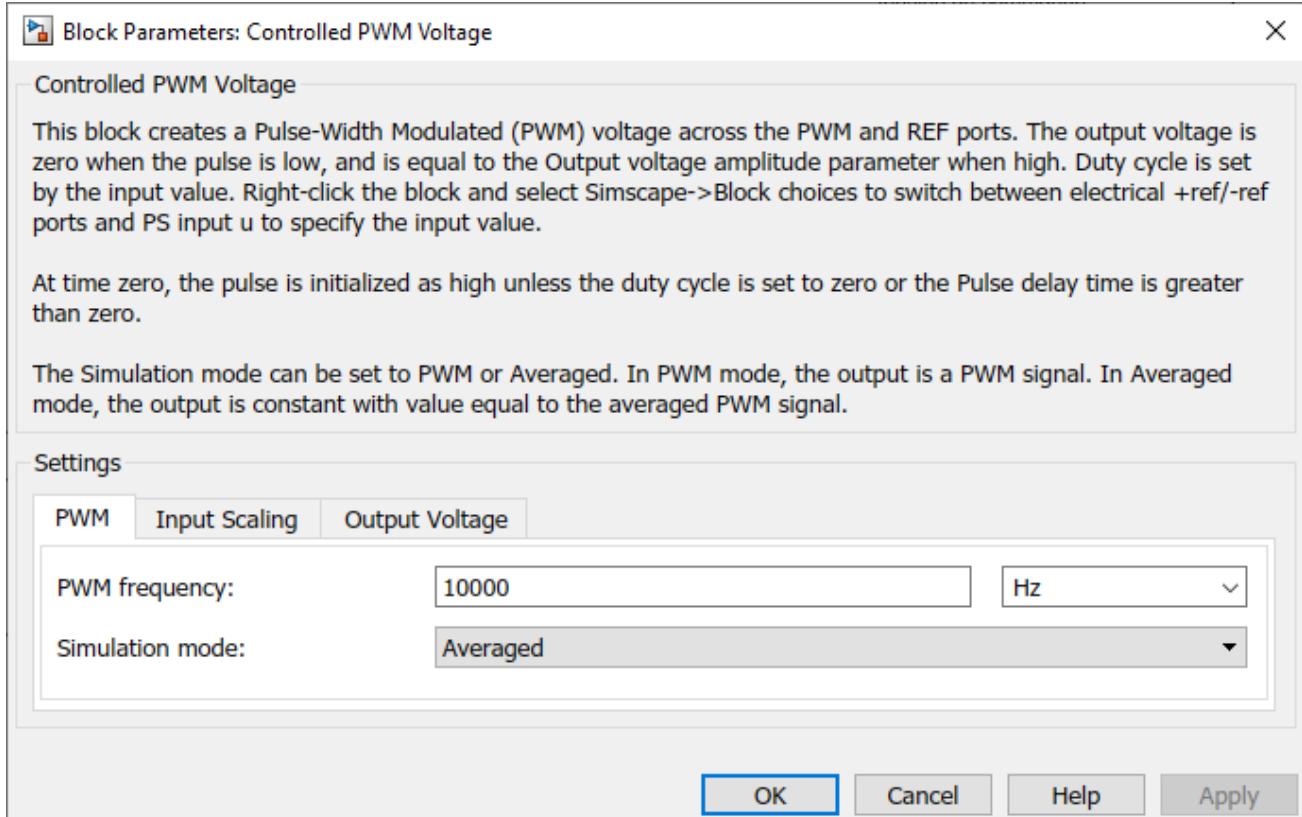


Figure 187 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage

Paramétrage de l'onglet Input Scaling :

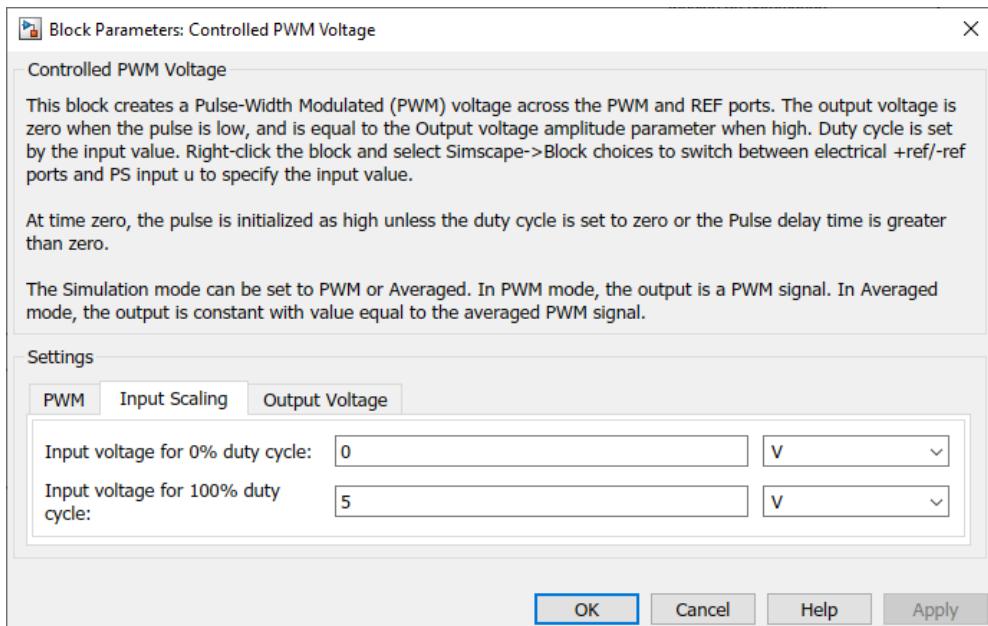


Figure 188 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage

Paramétrage de l'onglet Output Voltage

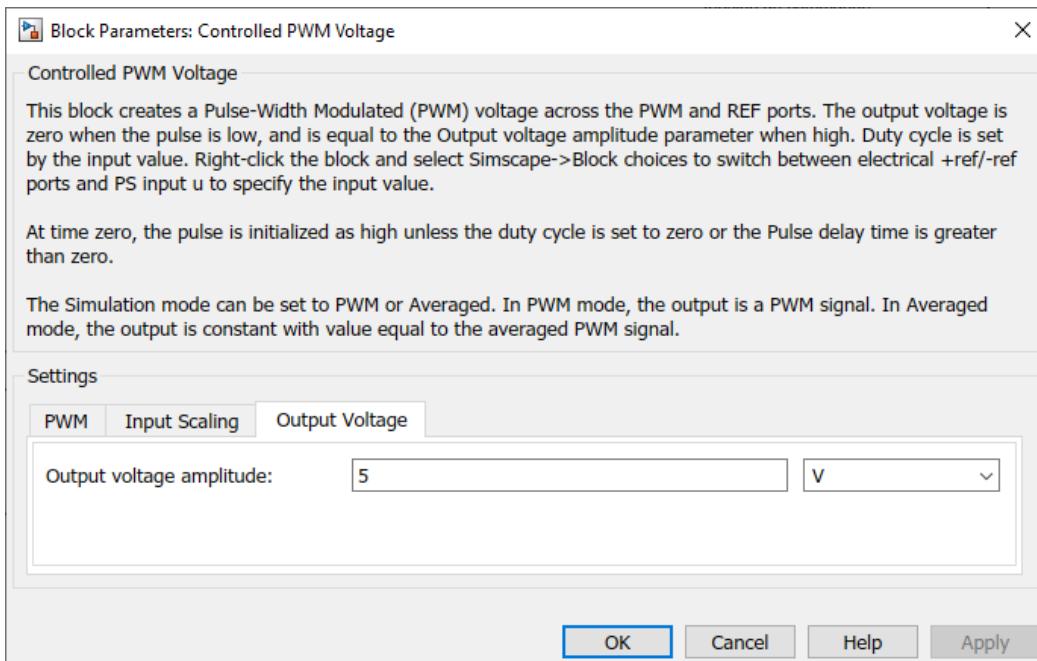
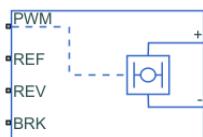


Figure 189 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage

Paramétrage

H-Bridge



Simscape/Electronics/Actuators and Drivers/Drivers

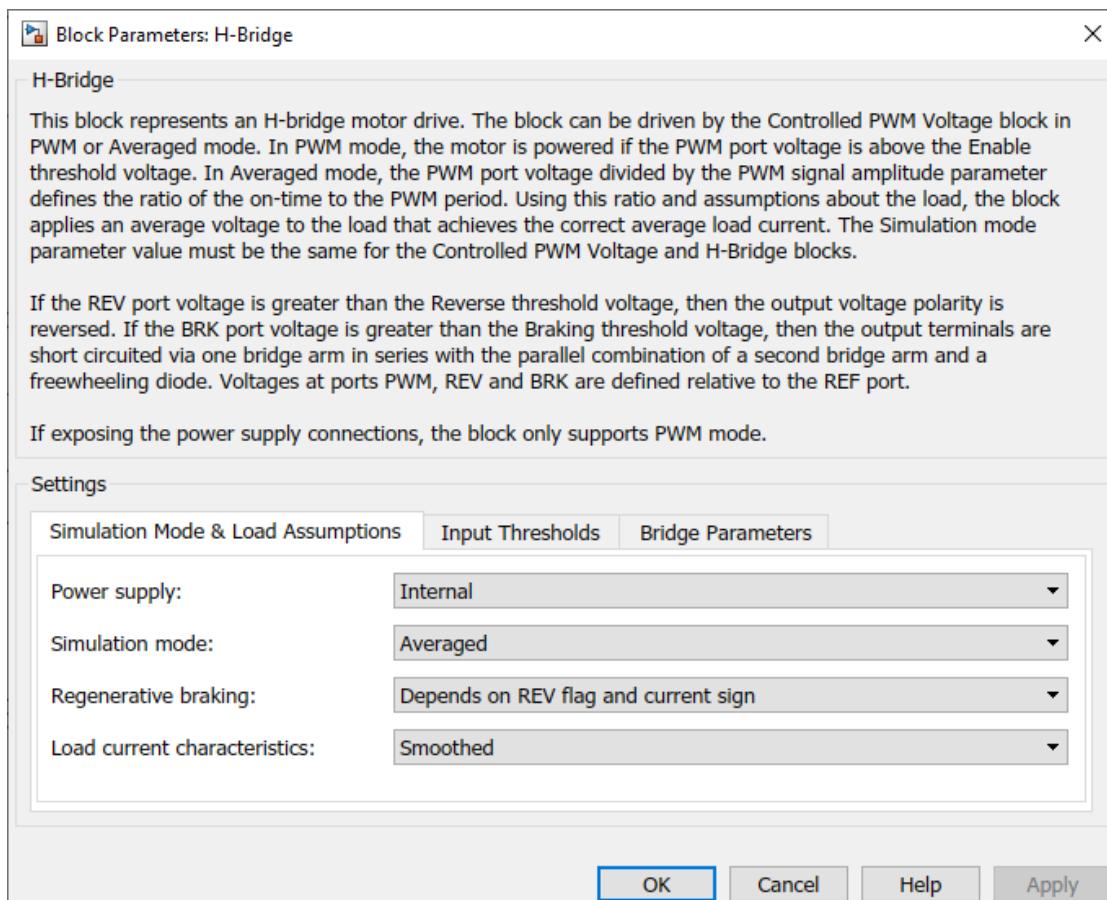
Ce composant modélise un pont en H. Il possède 6 ports de type PCP du domaine électrique.

- **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

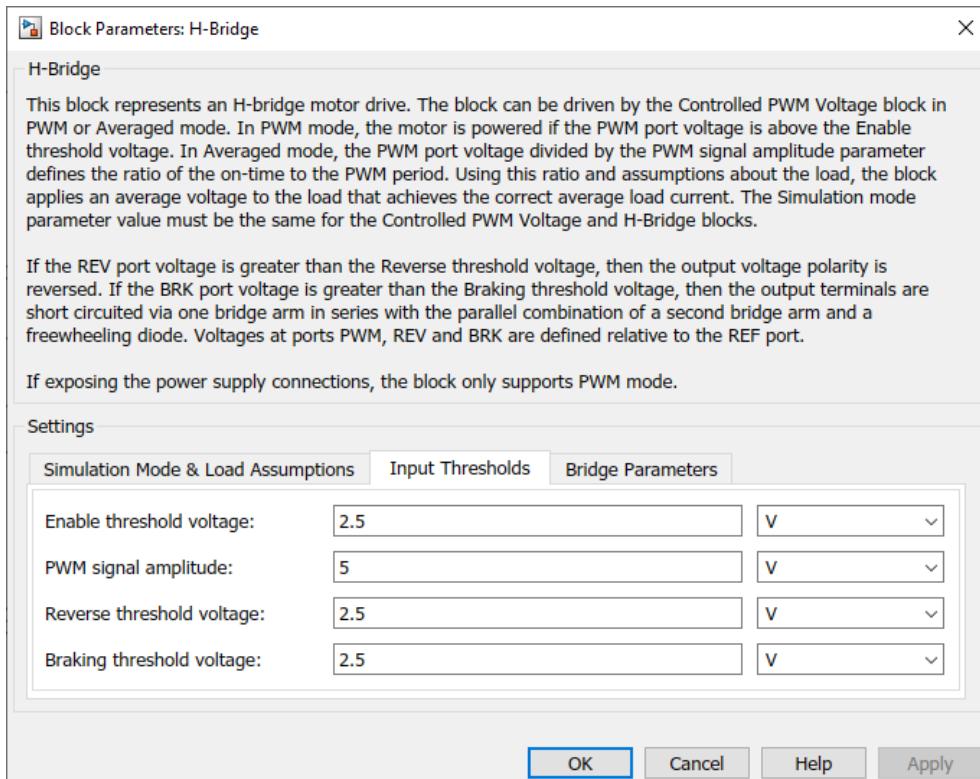
Le **Simulation Mode** est réglé sur **Average**.



Power supply: permet de spécifier si les ports d'alimentation du pont en H sont internes ou externes.

Simulation mode : choix entre **Average** et **PWM**

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

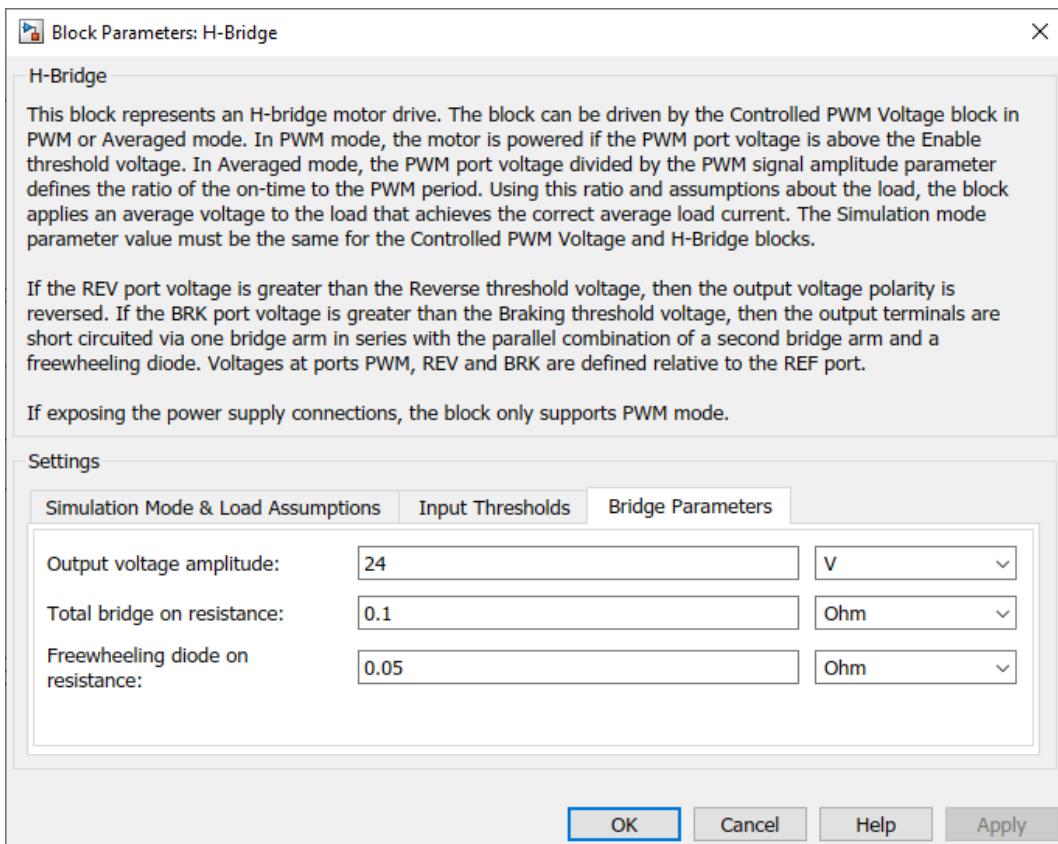


Figure 190 : paramétrage du bloc H-Bridge

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» et fixer sa valeur à **1** au lieu de **4** afin de modifier l'amplitude de la tension d'alimentation

Lancer la simulation et **visualiser** l'influence sur la tension d'alimentation et la vitesse de rotation du moteur (Figure 191 et Figure 192).

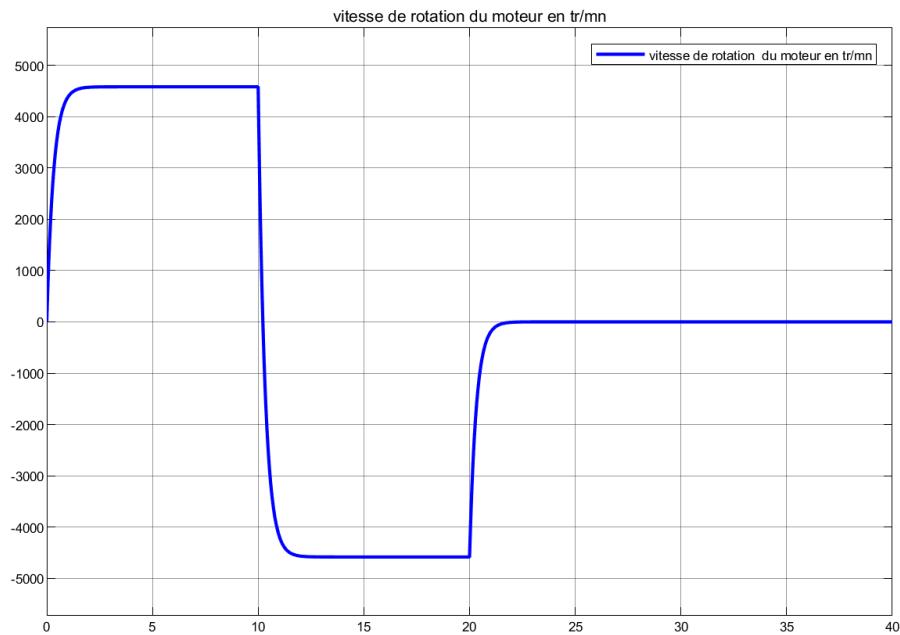


Figure 191 : vitesse de rotation du moteur commandé par le pont en H

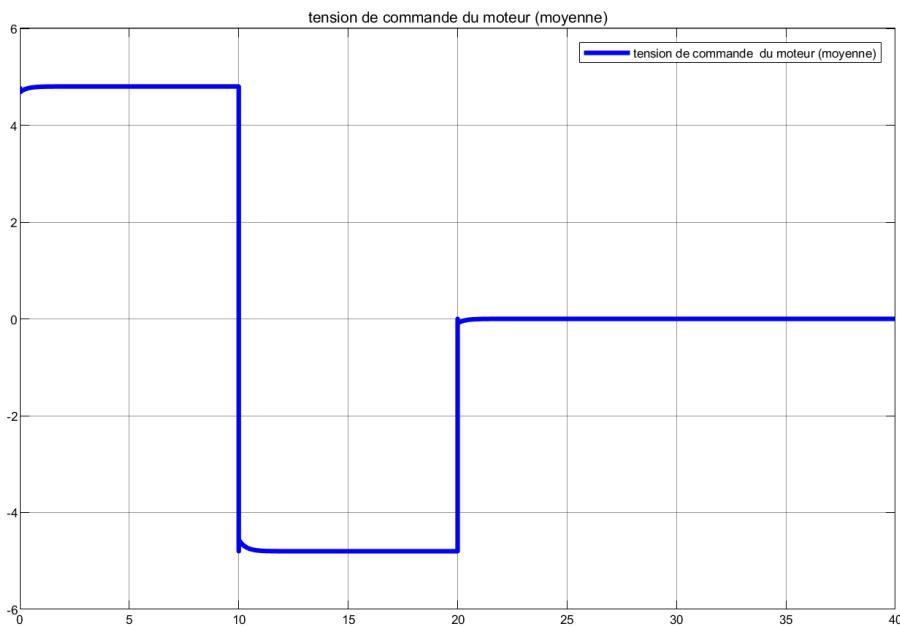


Figure 192 : tension moyenne d'alimentation du moteur

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 193).

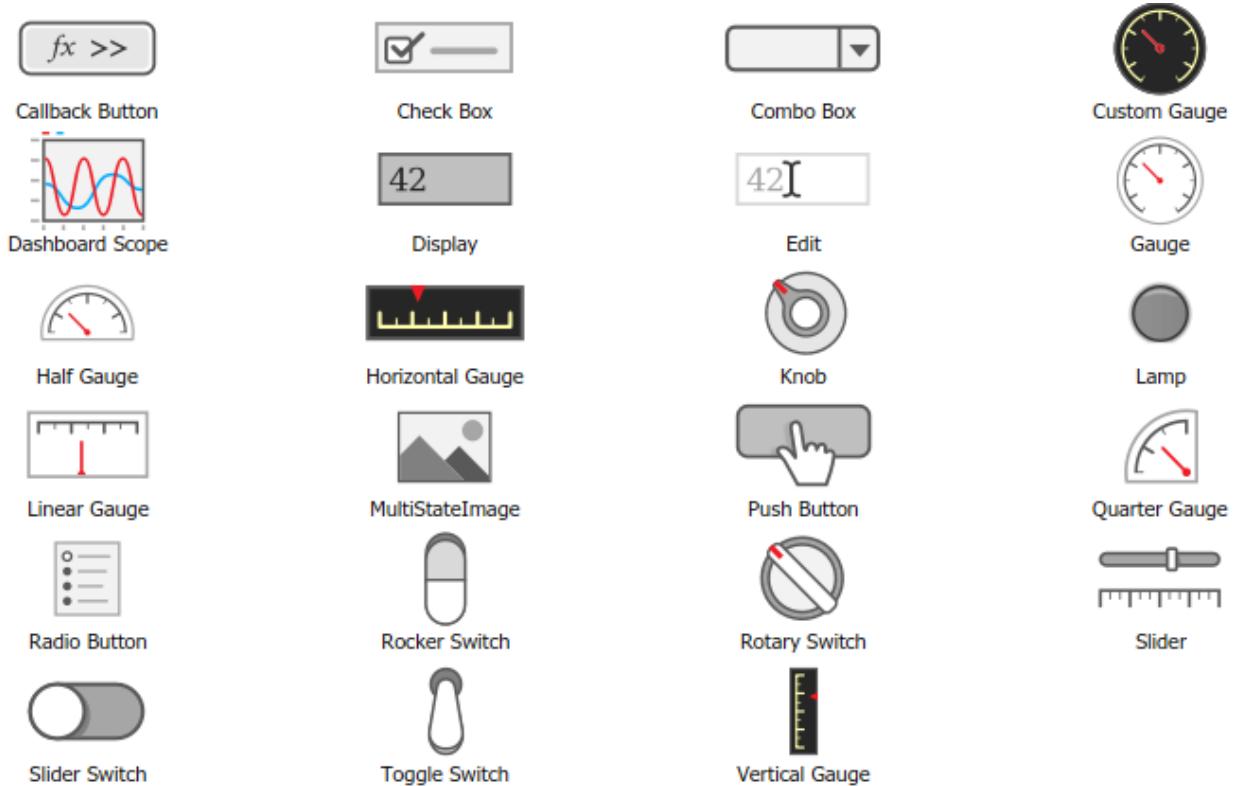


Figure 193 : les éléments de la bibliothèque de Simulink « Dashboard »

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.

1. Exemple de modèle interactif

Ouvrir le fichier « **axe_lineaire_dashboard.slx** » (Figure 194).

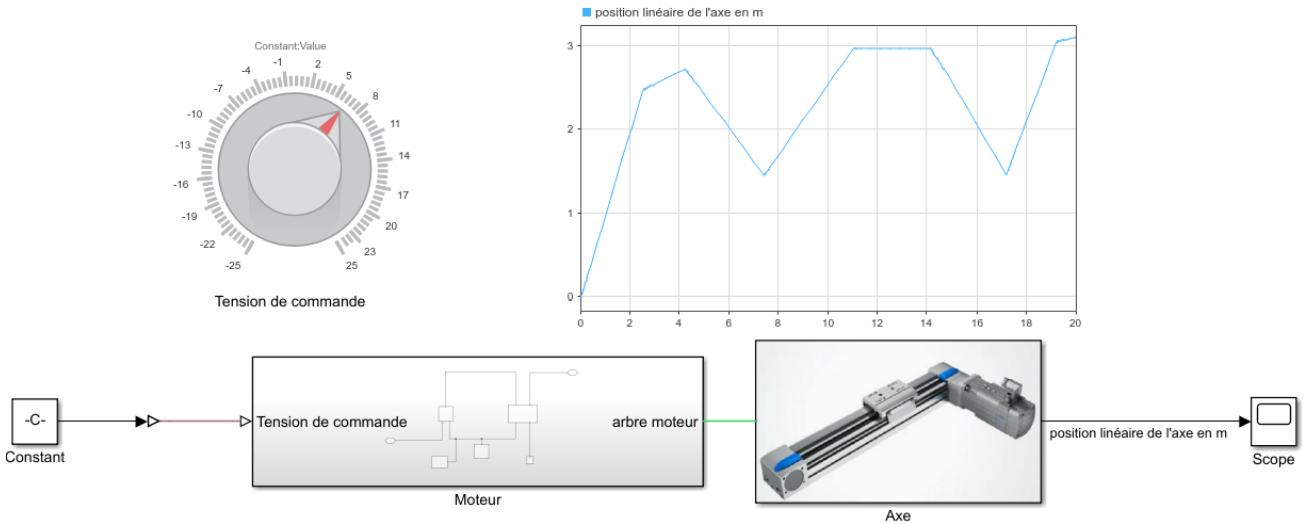


Figure 194 : pilotage interactif d'un modèle

Ce fichier contient le modèle de l'axe linéaire commandé en boucle ouverte. Deux composants de la bibliothèque Dashboard ont été ajoutés afin de rendre ce modèle interactif :

- Un **Knob**, bouton qui a été connecté avec le bloc constant qui permet de régler la tension de commande de l'axe.
- Un **Dashboard Scope** qui permet de visualiser de manière dynamique l'évolution d'un paramètre durant la simulation. Ce **Dashboard Scope** est relié au signal qui donne la position de l'axe linéaire en m.

Le **temps de simulation** est réglé sur **20 s** et la simulation a été paramétrée pour se dérouler en temps réel.

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 195).

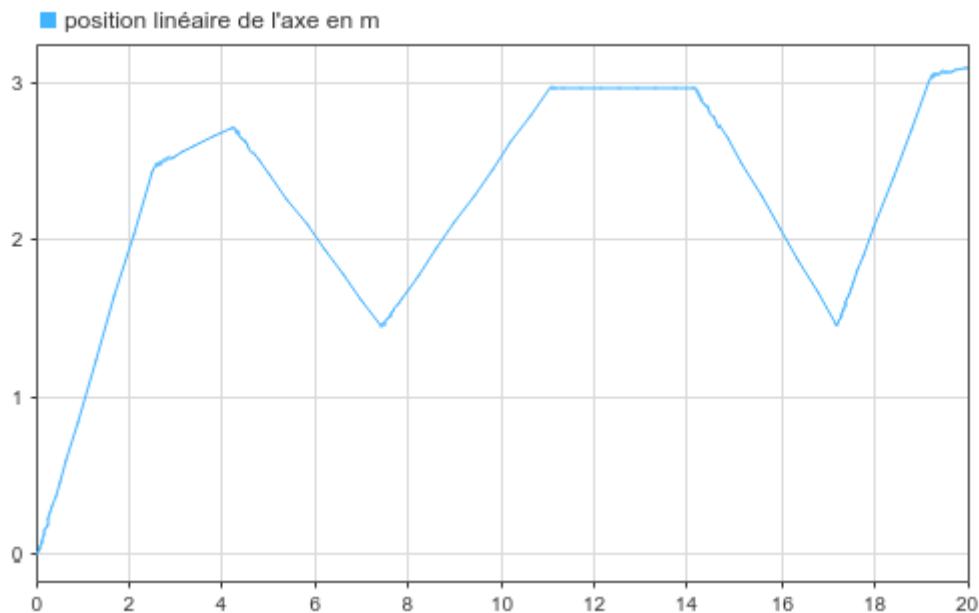


Figure 195 : visualisation en temps réel de la position de l'axe linéaire

2. Utilisation des blocs de la bibliothèque

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

Ouvrir le modèle « axe_lineaire_dashboard_start.slx ».

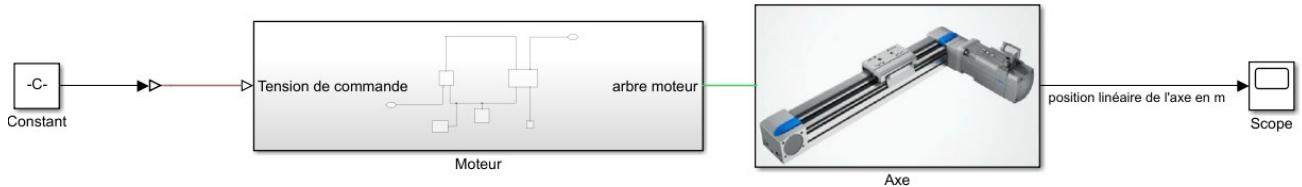


Figure 196 : modèle « axe_lineaire_dashboard_start.slx »

Ce fichier contient le modèle de l'axe linéaire précédemment réalisé. Nous allons ajouter des composants de la bibliothèque Dashboard afin de le rendre interactif.

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.

Pour cela cliquer sur la petite flèche sous le bouton afin d'ouvrir le menu déroulant, puis choisir **Simulation Pacing** (Figure 197).

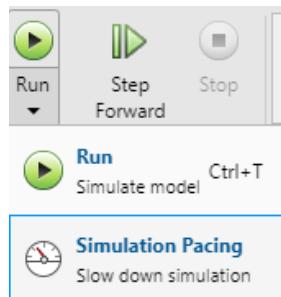


Figure 197 : sélection de la fonction Simulation Pacing

La fenêtre **Simulation Pacing Options** (Figure 198) permet d'indiquer le coefficient de déroulement temporel de la simulation.

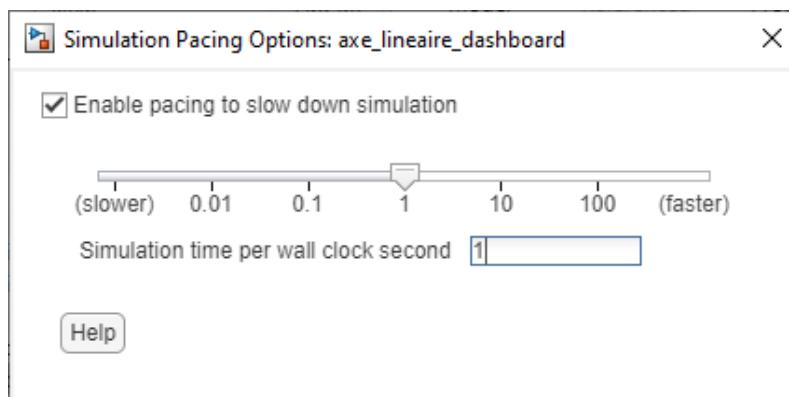


Figure 198 : paramétrage de la fenêtre du Simulation Pacing

Cocher la case **Enable pacing to slow down simulation**, puis choisir le coefficient de déroulement temporel de la simulation (**Simulation time per wall clock second**). Si ce coefficient est à 1, la simulation se déroulera en temps réel. Une seconde dans le temps de simulation durera exactement une seconde durant le processus de résolution du modèle. Cela permettra d'observer les phénomènes à la même vitesse que dans la réalité. Ce coefficient peut être modifié pour accélérer ou diminuer le temps de simulation du modèle.

Nous allons maintenant ajouter les composants de la bibliothèque Dashboard.

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

Faire **glisser, positionner et redimensionner** le bloc **knob** et le bloc **Dashboard Scope** pour obtenir la configuration de la Figure 199.

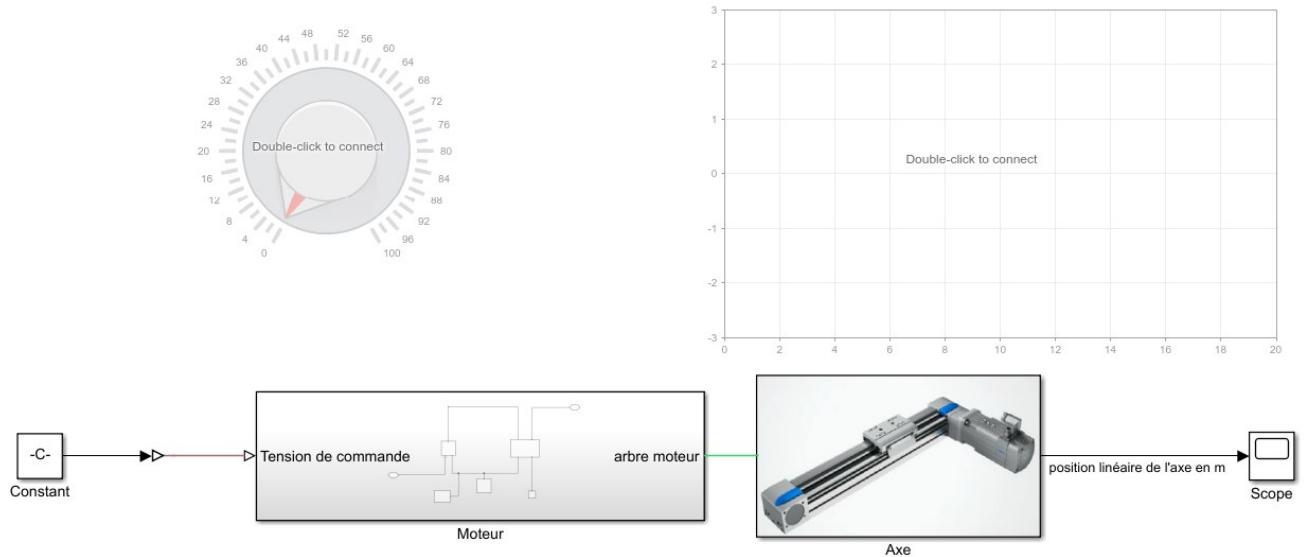


Figure 199 : insertion des blocs de la bibliothèque Dashboard

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.

Cliquez ensuite sur le bloc qui sera piloté par le bouton. Ici, il faudra sélectionner le bloc « **Constant** » qui impose la tension à l'entrée de l'axe linéaire puis valider la connexion en cliquant sur la case à cocher **connect** (Figure 200).

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 200.

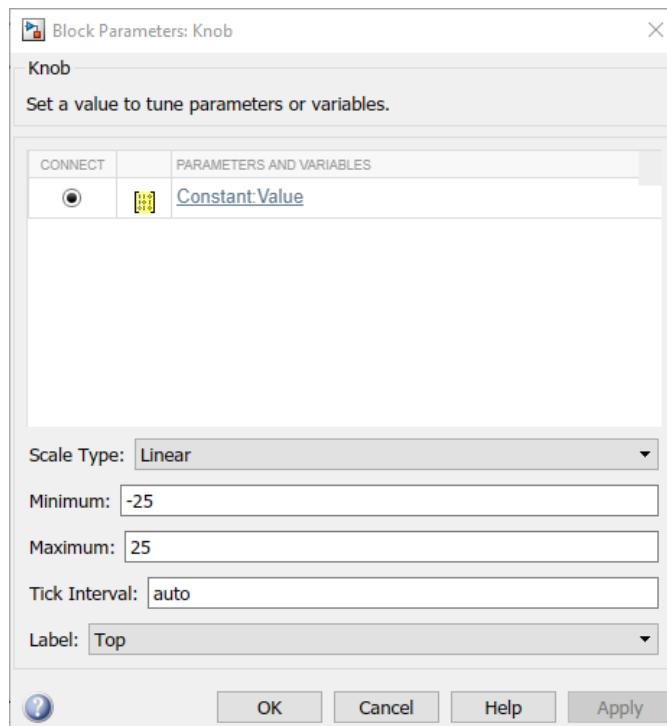


Figure 200 : fenêtre de paramétrage du bloc knob

Le bouton change alors d'aspect pour prendre la forme représentée sur la Figure 201.

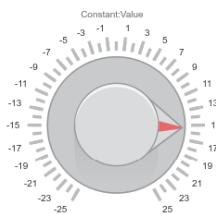


Figure 201 : allure du bouton knob après paramétrage et connexion avec le modèle

Paramétrage

Scope interactif



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 202).

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. Si ce paramètre est sur **auto**, le Time Span sera égal au temps de simulation.

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

La fenêtre de paramétrage doit être identique à la Figure 202.

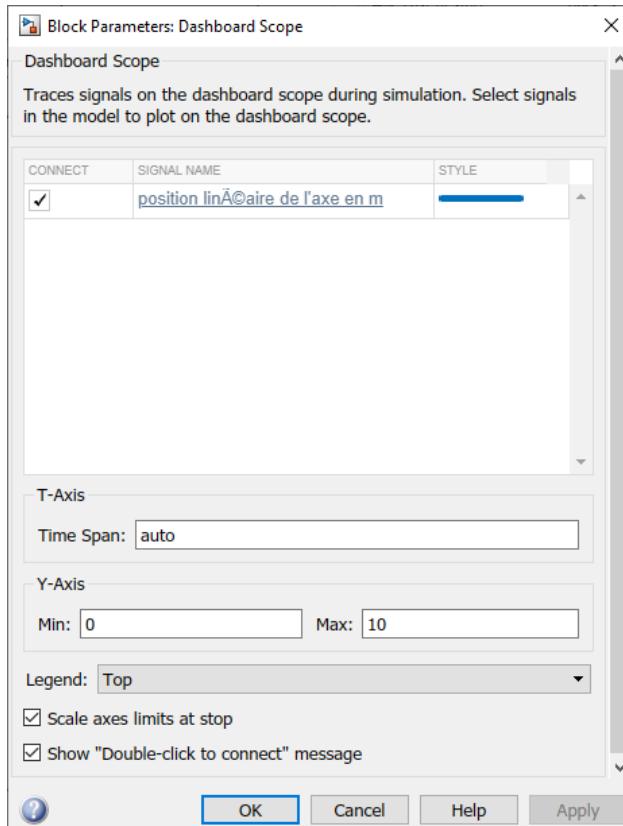


Figure 202 : fenêtre de paramétrage du bloc Dashboard Scope

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

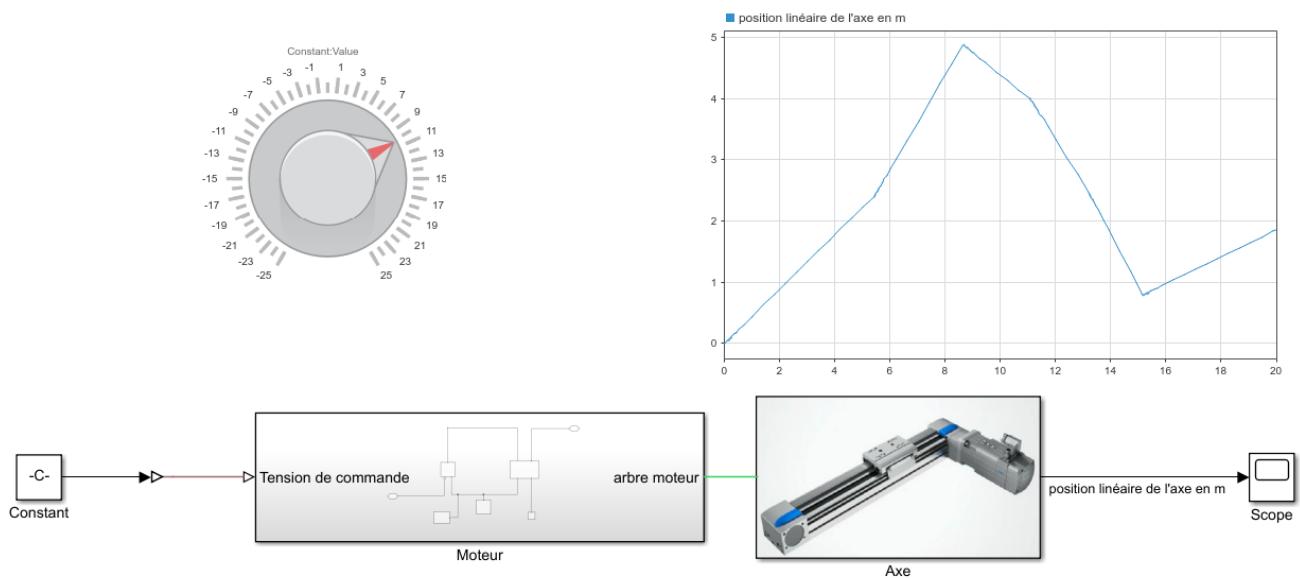


Figure 203 : simulation d'un modèle interactif avec les composants de la bibliothèque dashboard

E. Le domaine thermique

1. Rappels théoriques sur les échanges thermiques

Le domaine thermique de Simscape permet de modéliser les trois principaux phénomènes qui donnent lieu à des échanges thermiques :

- Le phénomène de conduction
- Le phénomène de convection
- Le phénomène de rayonnement

Comme pour chaque domaine physique de Simscape une connaissance théorique minimale du domaine physique est requise afin de pouvoir modéliser des systèmes.

2. Le phénomène de conduction

La conduction (Figure 204) est une transmission de chaleur provoquée par la différence de température entre deux régions d'un même matériau qui sont en contact physique. Un flux de chaleur s'établit naturellement de la partie chaude vers la partie froide du solide. Le flux thermique qui s'établit dépend des caractéristiques et de la géométrie du matériau. Plus le matériau est isolant et moins il y a de conduction.

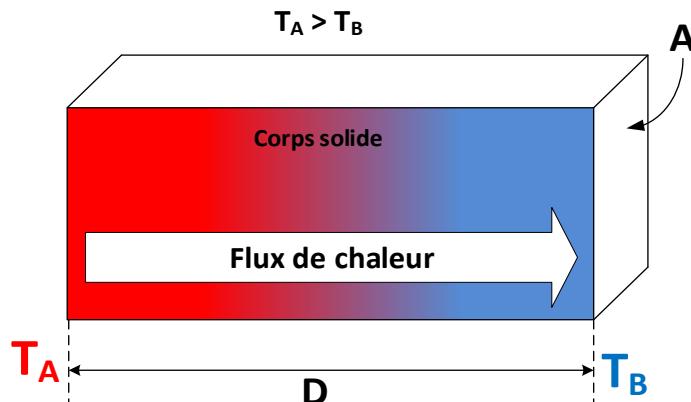


Figure 204 : transfert de chaleur par conduction

Ce type de transfert thermique peut être modélisé par la loi de Fourier définie de la manière suivante :

$$Q = \lambda \frac{A}{D} (T_A - T_B)$$

Q : flux thermique en W ou J.s

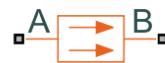
λ : conductivité thermique du matériau en W/(m.K)

A : aire de la surface d'échange, normale à la direction du flux thermique en m^2

D : distance entre les parois chaude et froide du matériau en m

T_A et T_B : températures des parois

Dans la bibliothèque thermique de Simscape ce phénomène est modélisé à l'aide du bloc **Conductive Heat Transfert** (Figure 205).



Conductive Heat Transfer

Figure 205 : bloc Conductive Heat Transfer de Simscape

Paramétrage du bloc Conductive Heat Transfert

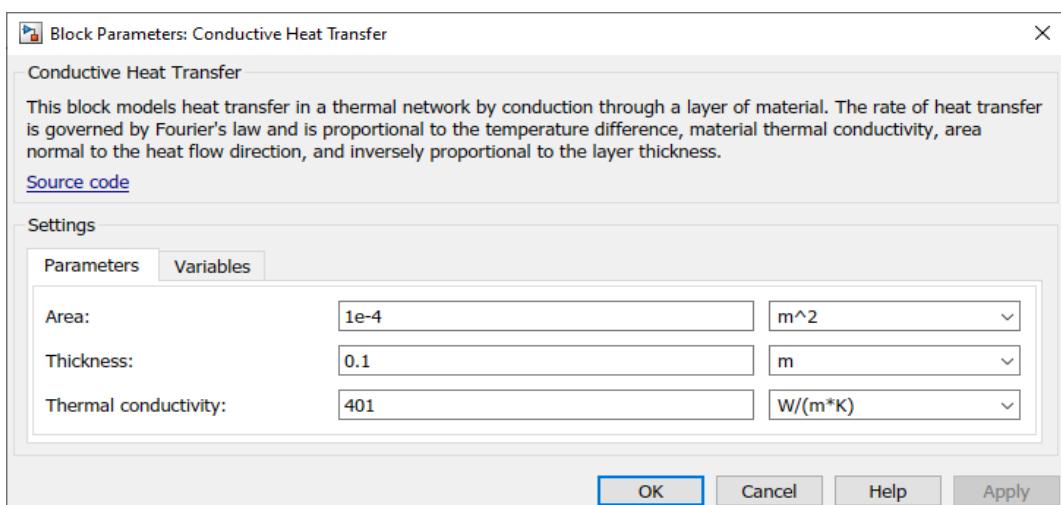


Figure 206 : Paramétrage du bloc Conductive Heat Transfer

Area : aire de la surface d'échange

Thickness : distance entre les parois chaude et froide du matériau

Thermal Conductivity : conductivité thermique du matériau au travers duquel s'effectue l'échange de chaleur par conduction.

3. Le phénomène de convection

La convection (Figure 207) est une transmission de chaleur au sein d'un fluide en mouvement ou entre un fluide en mouvement et une paroi solide. Plus le fluide est mobile et plus il y a de convection.

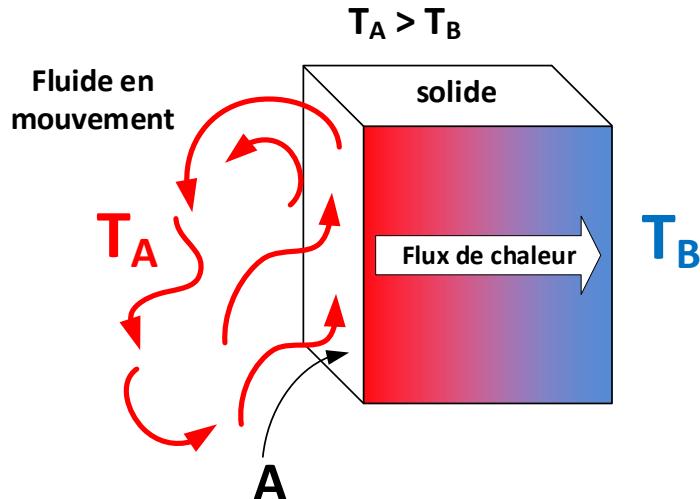


Figure 207 : transfert de chaleur par convection

Ce type de transfert thermique peut être modélisé par la loi de Newton définie de la manière suivante :

$$Q = k A (T_A - T_B)$$

Q : flux thermique en W ou J.s

k : coefficient de transfert de chaleur par convection W/(m²*K)

A : aire de la surface d'échange, normale à la direction du flux thermique m²

T_A et T_B : températures des deux corps

Dans la bibliothèque thermique de Simscape ce phénomène est modélisé à l'aide du bloc **Convective Heat Transfer** (Figure 208).



Figure 208 : bloc Convective Heat Transfer de Simscape

Paramétrage du bloc Convective Heat Transfer

Convective Heat
Transfer



Simscape/Foundation
Library/Thermal/Thermal Elements

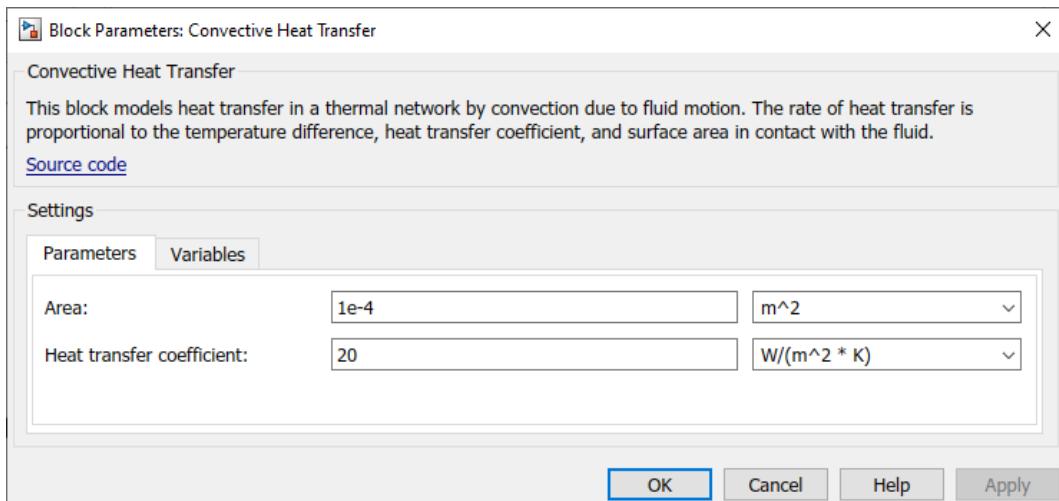


Figure 209 : Paramétrage du bloc Convective Heat Transfer

Area : aire de la surface d'échange

Heat Transfer coefficient : coefficient de transfert thermique par convection

4. Le phénomène de rayonnement

Le rayonnement thermique est un transfert de chaleur d'un corps à un autre par onde électromagnétique sans contact direct. Ce type de transfert peut se faire dans le vide.

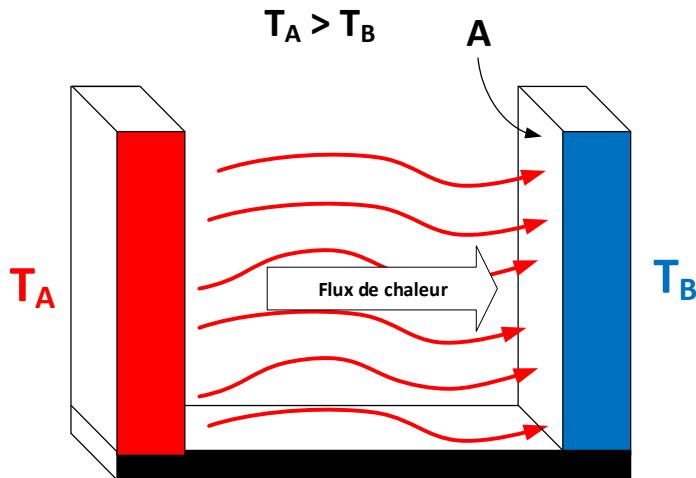


Figure 210 : transfert de chaleur par rayonnement

Ce type de transfert thermique peut être modélisé par la loi de Stefan-Boltzman définie de la manière suivante :

$$Q = kA(T_A^4 - T_B^4)$$

Q : flux thermique en W ou J.s

k : coefficient de radiation $W/(m^2 \cdot K^4)$

A : aire de la surface d'échange, normale à la direction du flux thermique en m^2

T_A et T_B : températures des deux corps

Dans la bibliothèque thermique de Simscape ce phénomène est modélisé à l'aide du bloc **Radiative Heat Transfert** (Figure 211).

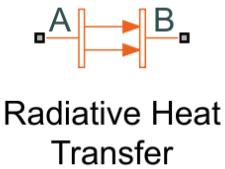


Figure 211 : bloc Radiative Heat Transfer de Simscape

Paramétrage du bloc Radiative Heat Transfer

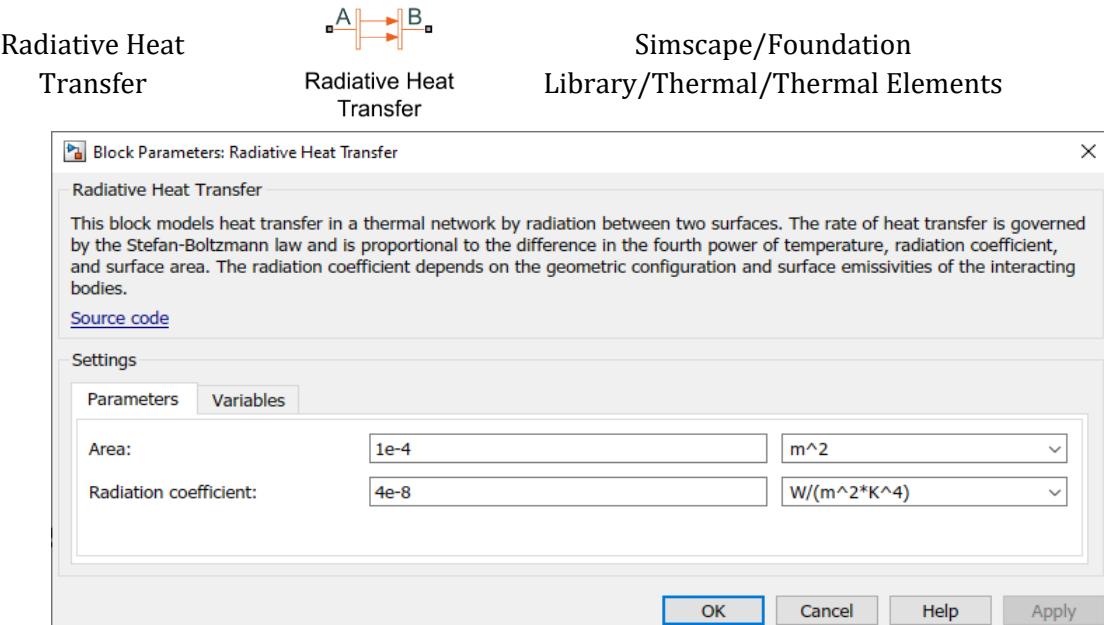


Figure 212 : Paramétrage du bloc Radiative Heat Transfer

Area : aire de la surface d'échange

Radiation coefficient : coefficient de transfert thermique par rayonnement

5. La masse thermique

La masse thermique permet de modéliser l'inertie thermique d'un élément et sa capacité à stocker et restituer de la chaleur. Plus le matériau aura une masse thermique importante et plus il sera long à monter en température. De plus cette montée en température dépendra de la nature du matériau. Un matériau métallique montera plus facilement en température que du béton ou que du bois. La masse thermique d'un matériau dépend donc de sa masse et de sa chaleur spécifique.

La masse thermique est caractérisée par l'équation suivante

$$Q = c \cdot m \frac{dT}{dt}$$

Q : flux thermique en W ou J.s

c : chaleur spécifique du matériau J/kg/K

m : masse du matériau en kg

$\frac{dT}{dt}$: variation de température de la masse en K/s

Dans la bibliothèque thermique de Simscape une masse thermique se modélise à l'aide du bloc **Thermal Mass** (Figure 213).



Figure 213 : bloc Thermal Mass de Simscape

Paramétrage du bloc Thermal Mass

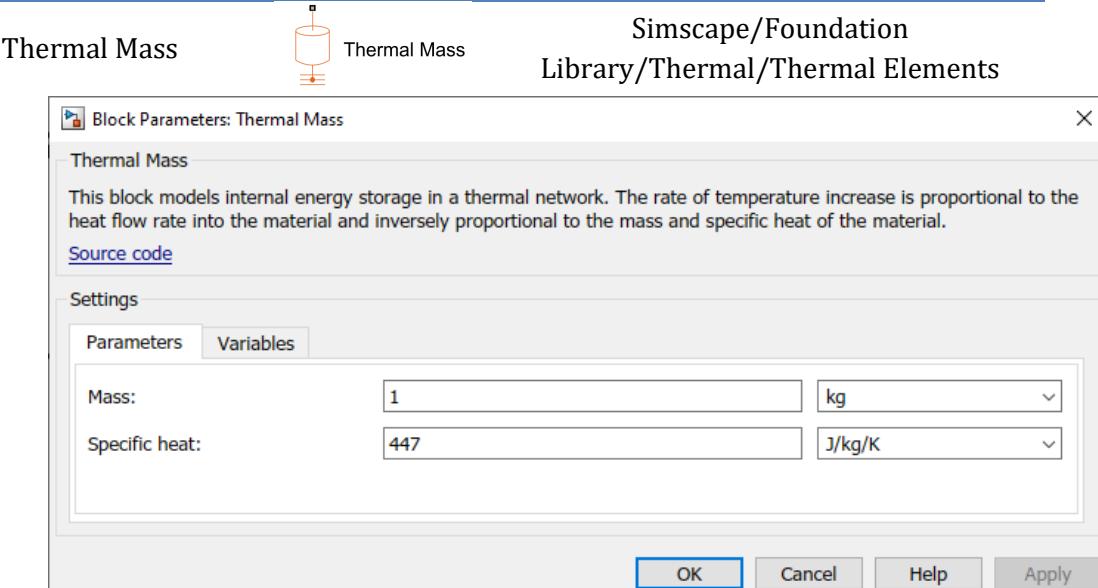


Figure 214 : Paramétrage du bloc Thermal Mass

Mass : masse du matériau

Specific Heat : chaleur spécifique

6. La résistance thermique

La résistance thermique est un concept permettant d'appréhender de manière globale le comportement thermique d'un élément. Ce comportement peut représenter des échanges thermiques par conduction, convection ou rayonnement. Cette représentation est très utilisée puisqu'elle permet de modéliser de manière globale la résistance que pourra présenter une paroi à un flux thermique qui s'établit entre deux isothermes. Plus la résistance thermique est élevée et plus le matériau sera isolant. En fonction du point de vue de modélisation choisi et des données à disposition pour la modélisation, les échanges thermiques pourront donc être modélisés soit à l'aide de résistances thermiques soit à l'aide des blocs, conduction, convection et rayonnement.

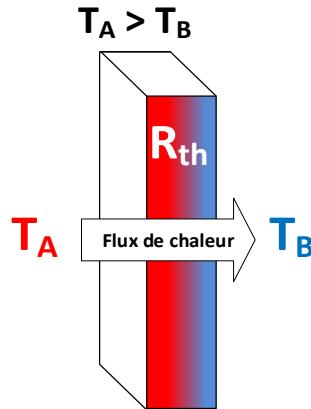


Figure 215 : résistance thermique R_{th}

Une résistance thermique est caractérisée par l'équation suivante qui s'apparente à la loi d'Ohms du domaine électrique :

$$(T_A - T_B) = R_{TH} Q$$

Q : flux thermique en W ou J.s

R_{TH} : résistance thermique en K/W

$T_A - T_B$: différence de température entre les parois

Il est possible de définir la résistance thermique de conduction ou de convection par identification en utilisant les lois de Fourier ou de Newton et la relation $Q = \frac{(T_A - T_B)}{R_{TH}}$

Résistance thermique de conduction

$$Q = \lambda \frac{A}{D} (T_A - T_B) = \frac{1}{R_{cond}} (T_A - T_B) \Rightarrow R_{cond} = \frac{D}{\lambda \cdot A}$$

D : distance entre les parois chaude et froide du matériau en m

λ : conductivité thermique du matériau en W/(m.K)

A : aire de la surface d'échange, normale à la direction du flux thermique m²

R_{cond} : résistance thermique de conduction en K/W

On notera qu'une augmentation de la surface d'échange A diminue la résistance thermique de conduction. Une augmentation de l'épaisseur du matériau augmente la résistance thermique de conduction.

Résistance thermique de convection

$$Q = kA(T_A - T_B) = \frac{1}{R_{conv}}(T_A - T_B) \Rightarrow R_{conv} = \frac{1}{kA}$$

k : coefficient de transfert de chaleur par convection W/(m²*K)

A : aire de la surface d'échange, normale à la direction du flux thermique m²

R_{conv} : résistance thermique de convection en K/W

On notera qu'une augmentation de la surface d'échange A diminue la résistance thermique de convection. L'épaisseur du matériau n'intervient pas dans la résistance thermique de convection puisque les échanges de chaleur se produisent à la surface du matériau.

Résistance thermique surfacique

Dans la documentation qui donne les caractéristiques des matériaux, le terme de résistance thermique est souvent évoqué. La résistance thermique permet de comparer la performance énergétique des matériaux. Plus la résistance thermique est élevée et plus le matériau est isolant.

Il faut être très vigilant dans la mesure ou la résistance thermique est souvent donnée en m².K/W. Il s'agit en fait de la **résistance thermique surfacique** qui représente la résistance thermique d'un mètre carré de matériau. La résistance thermique surfacique est donc une grandeur usuelle qui diffère de la résistance thermique définie théoriquement.

Il existe une relation très simple entre ces deux grandeurs :

$$R_{thermique}(K/W) = \frac{R_{thermique\ surfacique}(m^2.K/W)}{A(m^2)}$$

La résistance thermique surfacique de conduction est donnée par :

$$R_{thermique\ surfacique\ conduction} = \frac{D}{\lambda} (m^2.K/W)$$

La résistance thermique surfacique de convection est donnée par :

$$R_{thermique\ surfacique\ convection} = \frac{1}{k} (m^2.K/W)$$

Dans la bibliothèque thermique de Simscape une résistance thermique se modélise à l'aide du bloc **Thermal Resistance** (Figure 216).



Thermal Resistance

Figure 216 : bloc Thermal Resistance de Simscape

Paramétrage du bloc Thermal Resistance

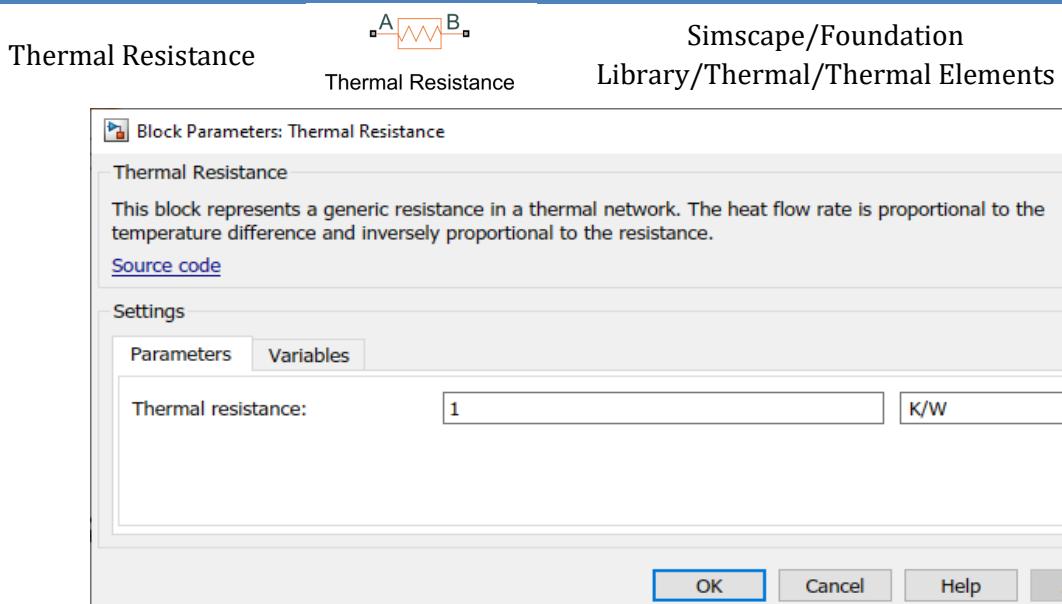


Figure 217 : Paramétrage du bloc Thermal Resistance

Thermal Resistance : résistance thermique de l'élément en K/W

7. Présentation de la bibliothèque Thermal de Simscape

La bibliothèque Thermal de Simscape comprend des éléments, des capteurs (Sensors) et des sources.

- Les éléments (Figure 218) permettent de modéliser les échanges thermiques comme décrit dans le paragraphe précédent
- Les sources (Figure 219) permettent de modéliser soit un flux thermique constant « **Heat Flow Rate Source** » soit une zone du modèle qui restera à température constante « **Temperature Source** ». Ces sources peuvent également être pilotées par des grandeurs variables « **Controlled Heat Flow Rate Source** » et « **Controlled Temperature Source** »
- Les capteurs (Figure 220) permettent de prélever les grandeurs physiques sur le modèle. Il sera possible de prélever la température en un point de notre modèle « **Temperature Sensor** » (Across variable) ou le flux thermique à travers une connexion « **Heat Flow Rate Sensor** » (Through variable)

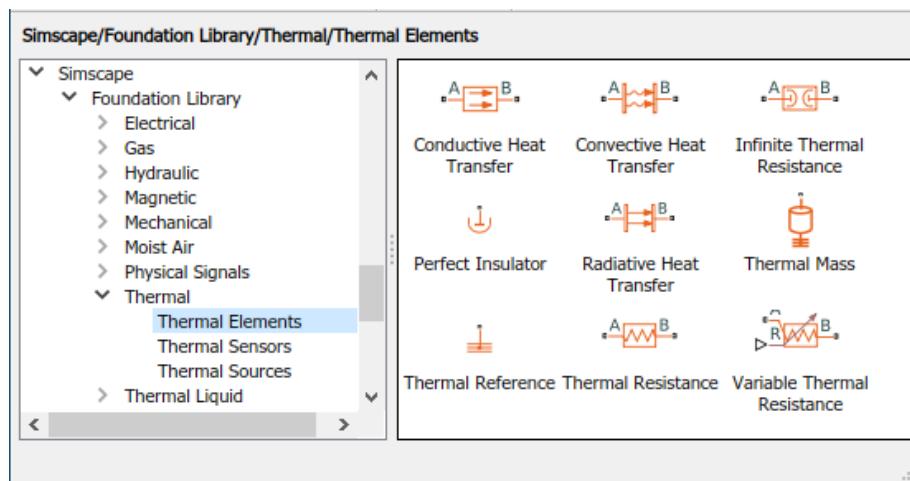


Figure 218 : les éléments de la bibliothèque Simscape Thermal

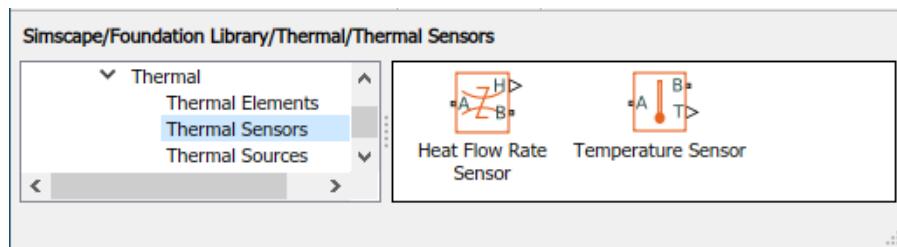


Figure 219 : les capteurs de la bibliothèque Simscape Thermal

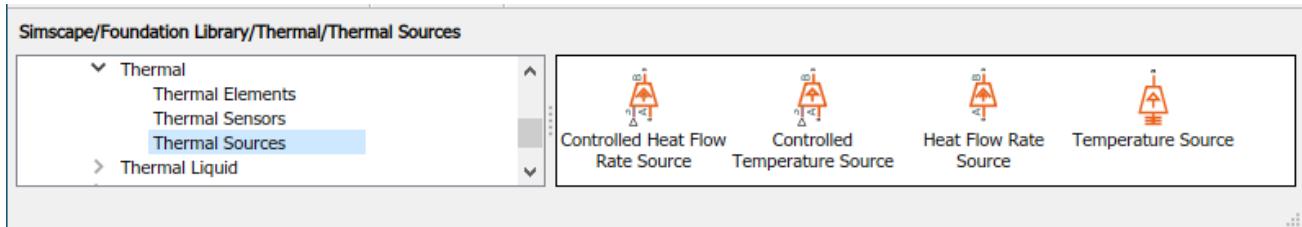


Figure 220 : les sources de la bibliothèque Simscape Thermal

8. Chauffage d'une barre métallique par conduction

La Figure 221 représente une barre d'acier à une température initiale de $T_B=20^\circ\text{C}$ placée dans un milieu parfaitement isolant. Ses deux extrémités sont reliées à un solide chaud à une température $T_H=300^\circ\text{C}$ et un solide froid à une température de $T_C=0^\circ\text{C}$.

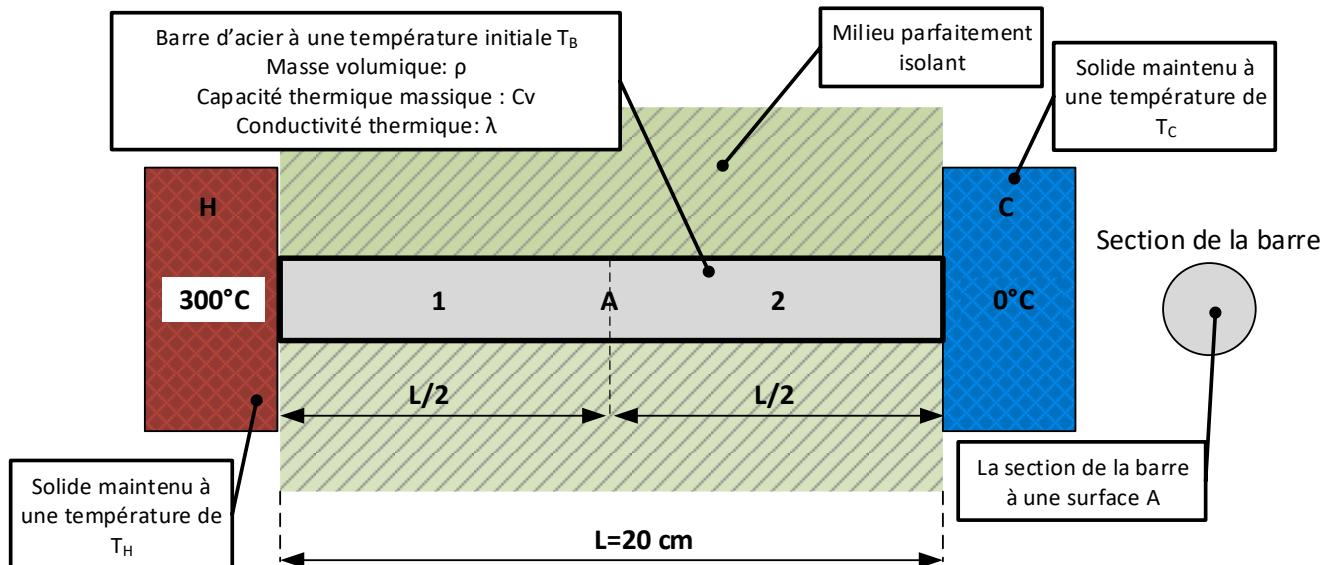


Figure 221 : chauffage d'une barre par conduction

Données pour la modélisation :

Longueur de la barre: $L = 20 \text{ cm}$

Diamètre de la barre : $D = 3.2 \text{ cm}$

Section de la barre : $A = 8 \text{ cm}^2$

Masse volumique de l'acier: $\rho = 7800 \text{ kg/m}^3$

Capacité thermique massique de l'acier : $C_v = 444 \text{ J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$

Conductivité thermique de l'acier: $\lambda = 50 \text{ W}\cdot\text{m}^{-1}\cdot\text{K}^{-1}$

L'objectif de la modélisation est de déterminer :

- le temps nécessaire pour atteindre le régime permanent
- la température au milieu de la barre (point A, distance $L/2$ des sources de température)
- le flux thermique qui traverse la barre en régime permanent

9. Choix des composants du domaine thermique

Ouvrir un « Blank Model » à partir de l'environnement Simulink.

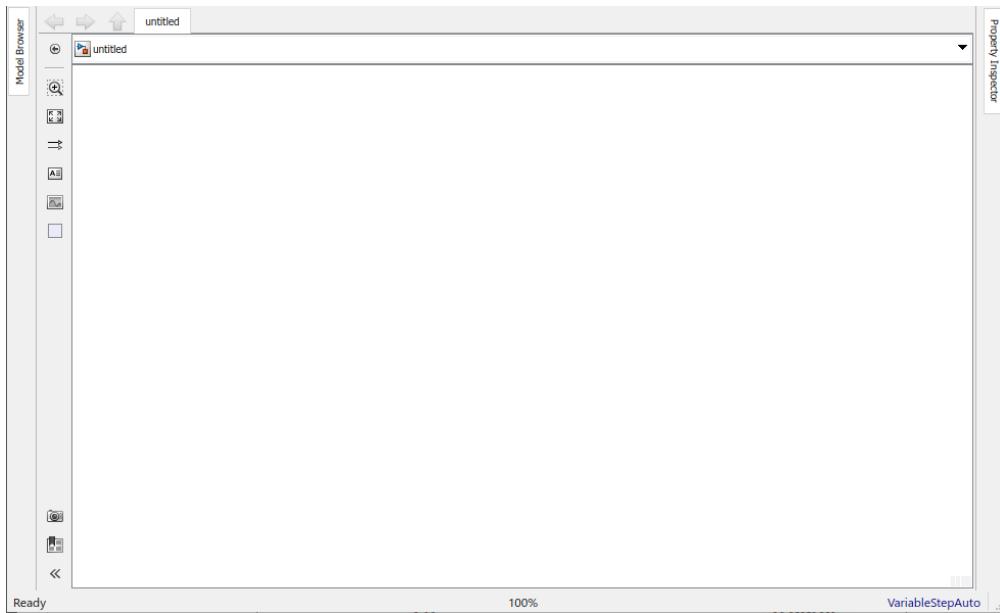


Figure 222 : ouverture d'un «Blank Model » Simulink



Cliquer sur pour ouvrir la bibliothèque de Simulink.

Pour modéliser l'ensemble nous aurons besoin de la bibliothèque de composants du domaine thermique.

Développer la bibliothèque Simscape pour faire apparaître la bibliothèque **Foundation Library/Thermal** (Figure 223)

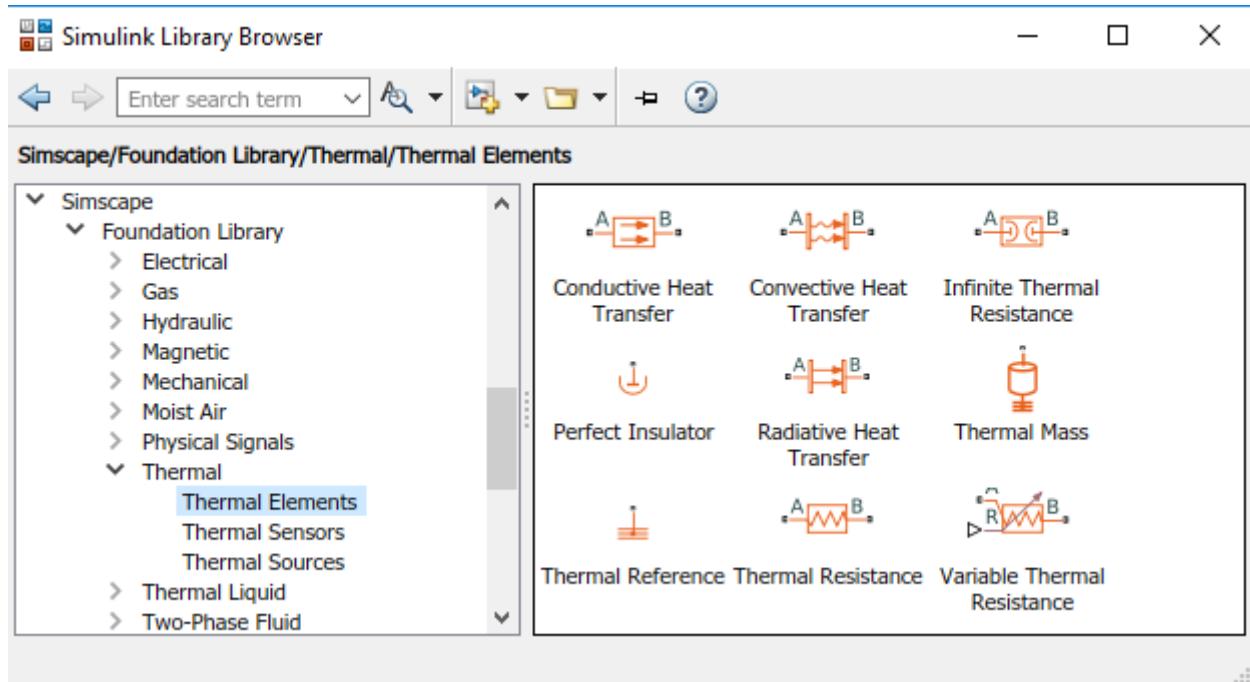


Figure 223 : bibliothèque Simscape Thermal

Pour commencer la démarche de modélisation il faut choisir les composants permettant de modéliser la barre.

La modélisation du chauffage d'une barre par conduction est obtenue en utilisant les composants présentés sur la Figure 224

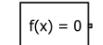
Fonction du composant	Icône	Emplacement dans la bibliothèque
Source de température		Simscape/Foundation Library/Thermal/Thermal Sources
Transfert de chaleur par conduction	 Conductive Heat Transfer	Simscape/Foundation Library/Thermal/Thermal Elements
Masse thermique	 Thermal Mass	Simscape/Foundation Library/Thermal/Thermal Elements
Référence du domaine thermique	 Thermal Reference	Simscape/Foundation Library/Thermal/Thermal Elements
Bloc solveur	 f(x) = 0 Solver Configuration	Simscape/Utilities

Figure 224 : les composants nécessaires à la modélisation du chauffage d'une barre par conduction

Ouvrir le fichier « **chauffage_barre_0.slx** » ou **Glisser/Déposer** les différents composants à partir des bibliothèques et les disposer dans la fenêtre de travail comme indiqué sur la Figure 225. Vous pouvez vous reporter à la Figure 226 pour prendre connaissance des commandes utiles à l'orientation des composants. Un simple double-clic sur le nom du composant permet de modifier son nom en ajoutant des informations qui améliorent la lisibilité.

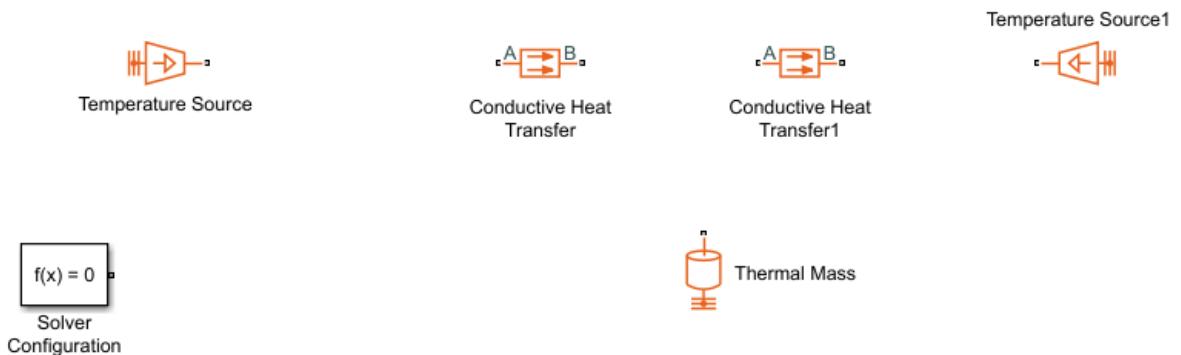


Figure 225 : positionnement des composants dans la fenêtre de travail

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 sélectionner Delete	Suppr
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 composant	

Figure 226 : les commandes de rotation et d'inversion des composants

Relier les composants pour obtenir le modèle de la Figure 227 ou ouvrir le fichier « chauffage_barre_1.slx ».

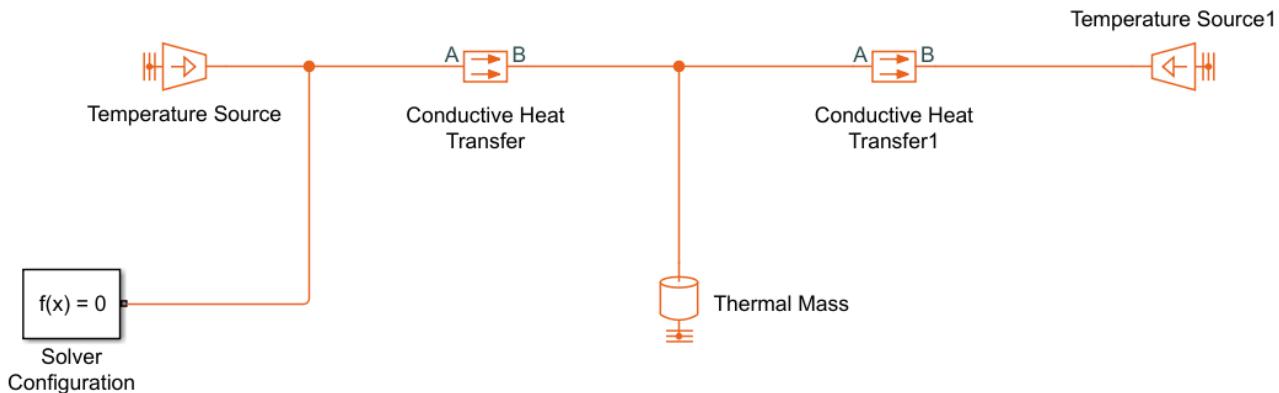


Figure 227 : modèle Simscape du chauffage d'une barre par conduction

10. Les ports transmetteurs de puissance (PCP) du domaine thermique

Chaque composant du domaine thermique possède des ports afin de pouvoir se connecter avec les autres composants. Ces ports qui permettent la transmission de puissance entre les composants d'un même domaine physique sont appelés « **Physical Conserving Port** » (PCP) (Figure 228).

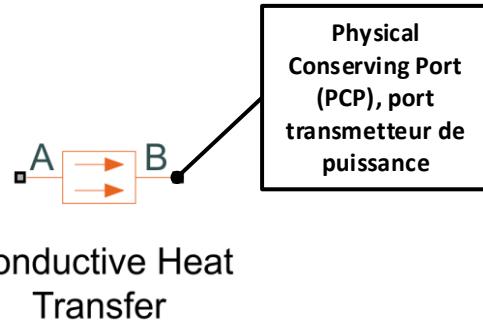


Figure 228 : les ports transmetteurs de puissance (PCP) du domaine thermique

Les connexions que nous venons de réaliser permettent de relier les ports de type PCP de notre modèle. Elles ne sont pas orientées et sont analogues aux liens qui relient les composants dans la réalité. Dans le cas de notre modèle qui ne fait intervenir que le domaine thermique, ces connexions peuvent être assimilées à des conducteurs de flux thermique parfaits, sans résistance thermique. Elles peuvent donc conduire un flux thermique et il est possible de mesurer la température d'un nœud d'une connexion par rapport à un nœud de référence.

11. Paramétrage des composants du modèle

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

Paramétrage du bloc Ideal Temperature Source

Ideal Temperature
Source



Simscape/Foundation
Library/Thermal/Thermal Elements

Le modèle possède 2 blocs de ce type. Ces deux blocs permettent de modéliser le fait que les extrémités de la barre sont maintenues à des températures constantes de 300°C et 0°C.

Le paramétrage de ce bloc consiste à indiquer la température de la source qui restera constante tout au long de la modélisation. Il faudra donc indiquer 300 degrés Celsius pour la source de température de gauche et 0 degrés Celsius pour la source de température de droite. Le menu déroulant propose un choix d'unités, il est possible de taper l'unité de son choix si elle n'est pas proposée par défaut (ici degC pour degrés Celsius)

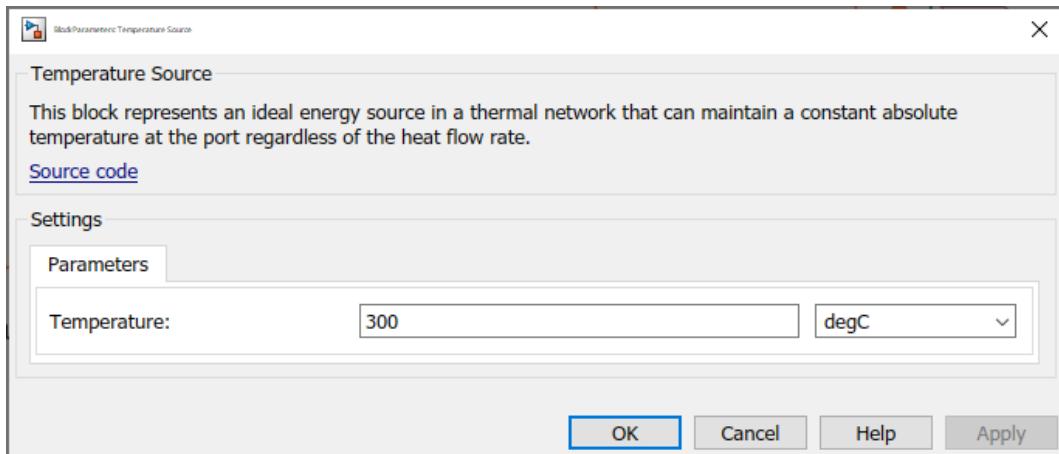


Figure 229 : paramétrage du bloc Temperature Source

Paramétrage du bloc Conductive Heat Transfert



Le modèle contient 2 blocs identiques de cette nature représentant chacun une moitié de la barre en acier.

Ce bloc permet de définir les paramètres de l'échange de chaleur par conduction. Les paramètres à renseigner sont :

Area : la surface correspondant à l'échange thermique. Dans notre exemple il s'agit de l'aire de la section de la barre soit $A=8\text{cm}^2$

Thickness : la longueur de l'éléments traversé par le flux thermique. Dans notre exemple, il s'agit de la moitié de la longueur totale de la barre soit 10 cm.

Thermal Conductivity : il faut renseigner ici la conductivité thermique du matériau au travers duquel s'effectue l'échange de chaleur par conduction. Dans notre exemple, la conductivité thermique de l'acier est de $50 \text{ W.m}^{-1}\text{.K}^{-1}$

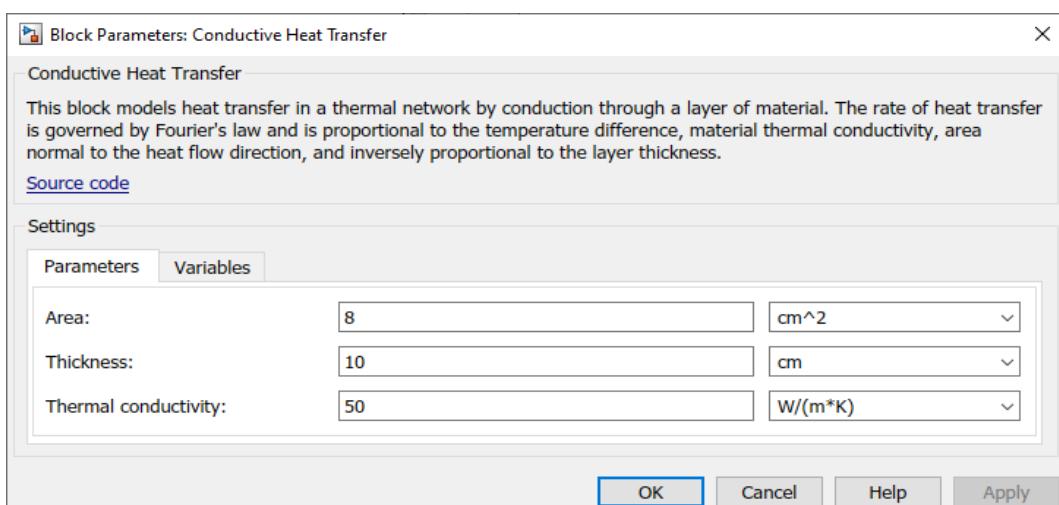
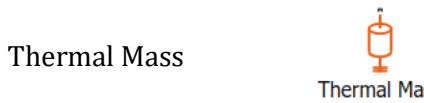


Figure 230 : paramétrage du bloc Conductive Heat Transfer

Paramétrage du bloc Thermal Mass



Simscape/Foundation
Library/Thermal/Thermal Elements

Ce bloc permet de modéliser le comportement d'une masse thermique. Les paramètres à renseigner sont :

Mass : la masse de l'élément considéré. Dans notre exemple la masse de la barre en acier est égale à son volume ($A \times L$) multiplié par sa masse volumique (ρ), soit 1,248 kg.

Specific Heat : il faut renseigner ici la capacité thermique massique du matériau considéré. Dans notre exemple, la capacité thermique massique de l'acier est de 444 J/Kg/K.

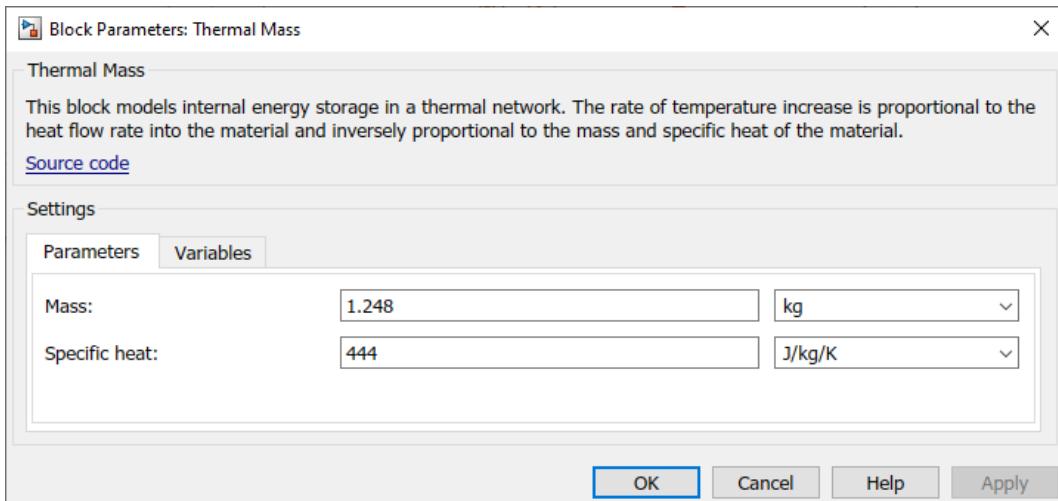
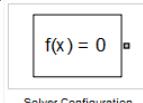


Figure 231 : paramétrage du bloc Thermal Mass

Paramétrage du bloc Solver

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 résoudre le modèle. L'absence de ce bloc dans un modèle Simscape provoque un message d'erreur au lancement de la simulation.

12. Mise en place des capteurs du domaine thermique

Les capteurs du domaine thermique permettent de prélever :

- Une température
- Un flux thermique

Les grandeurs physiques à prélever sur le modèle sont les suivantes :

- T_A : température au milieu de la barre
- Le flux thermique à l'entrée et à la sortie de la barre
- Le flux thermique absorbé ou restitué par la masse thermique

Afin de pouvoir relever ces grandeurs physiques, des capteurs doivent être placés sur le modèle. Nous allons planter un capteur de température au milieu de la barre et 3 capteurs de flux thermique.

L'implantation des capteurs doit respecter des règles strictes correspondant au caractère Across ou Through de la variable à prélever.

Le capteur de température

La température est une grandeur physique de type « Across » qui se mesure en se plaçant en parallèle dans le modèle.

Les ports de communication du capteur de température du domaine thermique sont décrits sur la Figure 232.

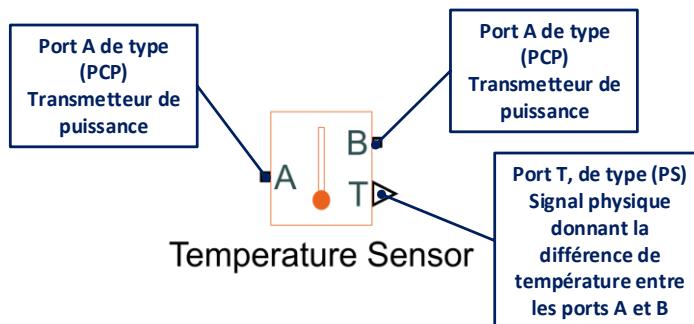


Figure 232 : les ports du capteur de température du domaine thermique

Port A : port de type (PCP) qui doit être relié au point du domaine physique dont on veut connaître la température par rapport au nœud relié au port B du capteur.

Port B : port de type (PCP) qui doit être relié au point du domaine physique qui sert de référence pour mesurer la température. Ce point peut être relié à un composant de type **Thermal Reference**.

Port T : port de type **Physical Signal (PS)** donnant la différence de température entre les ports A et B du capteur.

Les signaux de type Physical Signal (PS) contiennent une grandeur physique qui possède la dimension de la grandeur physique mesurée par le capteur. Elle pourra être affichée dans un scope en ayant préalablement fixé son unité à l'aide d'un bloc adapté. L'unité d'affichage devra obligatoirement être homogène à la grandeur physique mesurée.

Le capteur de température et de position **est orienté positivement de A vers B**. La Figure 233 montre l'implantation du capteur de température.

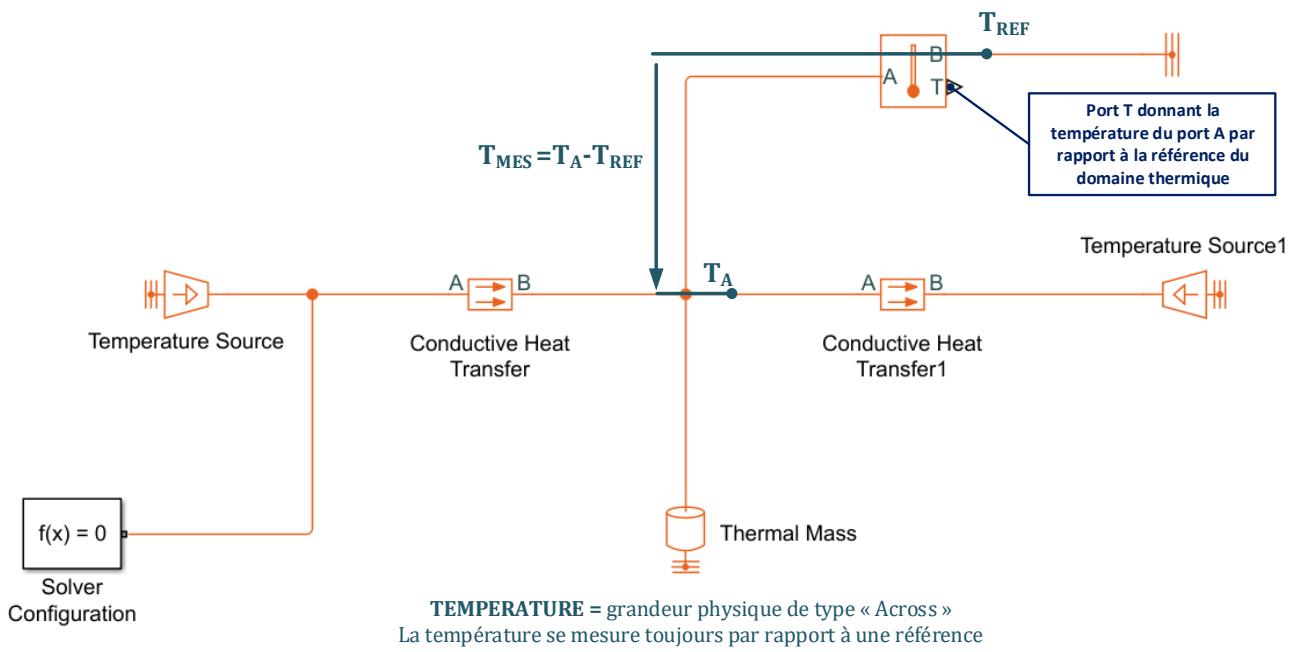
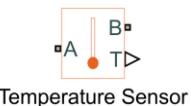


Figure 233 : implantation du capteur de température

Paramétrage du bloc Ideal Temperature Sensor

Ideal Temperature Sensor



Simscape/Foundation Library/Thermal/Thermal Sensors

Ce composant permet de mesurer la température d'un nœud d'une connexion par rapport à une référence. Aucun paramétrage n'est nécessaire pour ce bloc.

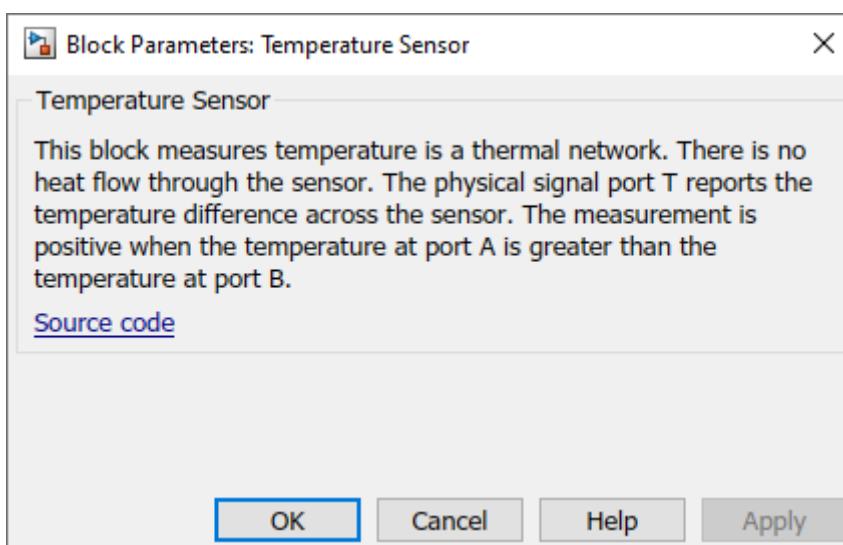


Figure 234 : paramétrage du bloc Ideal Temperature Source

Afin de pouvoir relever la température il faut ajouter un scope à notre modèle et convertir le signal physique dans la bonne unité pour l'affichage dans le scope.

Glisser/Déposer les composants de la Figure 235 dans le modèle du chauffage d'une barre par conduction et les relier afin d'obtenir la configuration de la Figure 236.

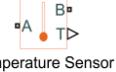
Capteur de température	 Temperature Sensor	Simscape/Foundation Library/ Thermal/Thermal Sensors
Conversion d'un signal physique en signal numérique	 PS-Simulink Converter	Simscape/Utilities
Afficheur de signaux numériques	 Scope	Simulink/Commonly Used Blocks

Figure 235 : les composants associés au capteur de température

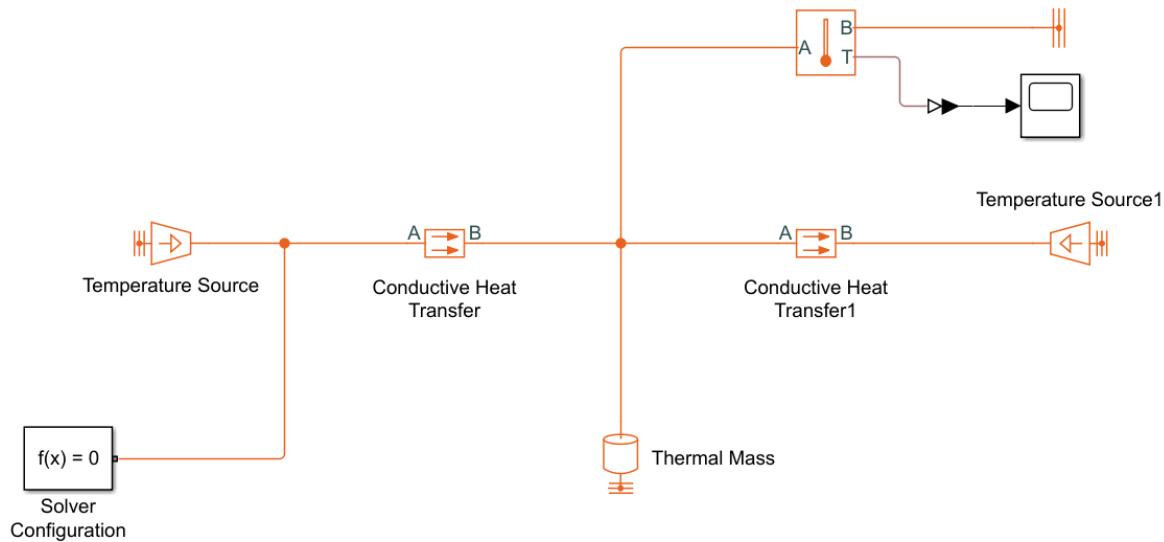


Figure 236 : implantation du capteur de température

Paramétrage du bloc PS-Simulink Converter

PS – Simulink Converter	 PS-Simulink Converter	Simscape/Utilities
----------------------------	---	--------------------

Ce composant permet de convertir le signal physique (PS) issu d'une grandeur mesurée en signal **Simulink** (S) 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. Si l'unité n'est pas spécifiée par l'utilisateur (valeur par défaut 1 ou inherit), la grandeur sera affichée dans le scope en unités SI.

Le menu déroulant propose un choix d'unités, il est possible de taper l'unité de son choix si elle n'est pas proposée par défaut.

Ce bloc permet de spécifier l'unité de la grandeur physique qui sera affichée dans le scope. Ici, nous utilisons les degrés Celsius. Cela nécessite deux paramétrages :

Output signal unit : degC

Cocher la case **Apply affine conversion** pour assurer la conversion des Kelvin en degrés Celsius.

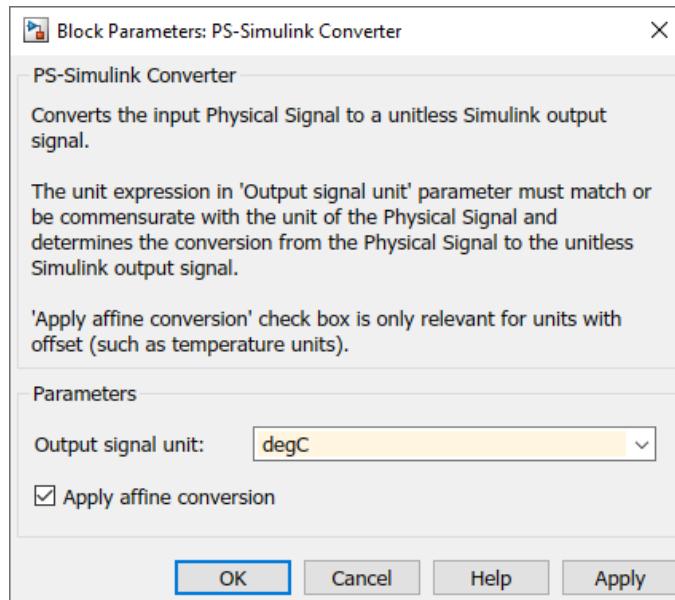


Figure 237 : paramétrage du bloc PS – Simulink Converter

Le capteur de flux thermique

Le flux thermique est une grandeur physique traversante de type « Through » qui se mesure en se plaçant en série dans une connexion.

Les ports de communication de capteur de flux thermique sont décrits sur la Figure 238.

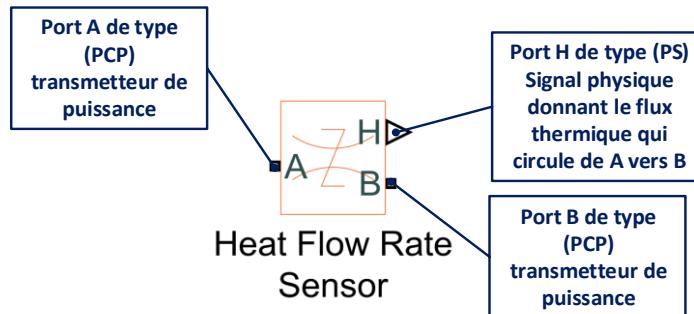


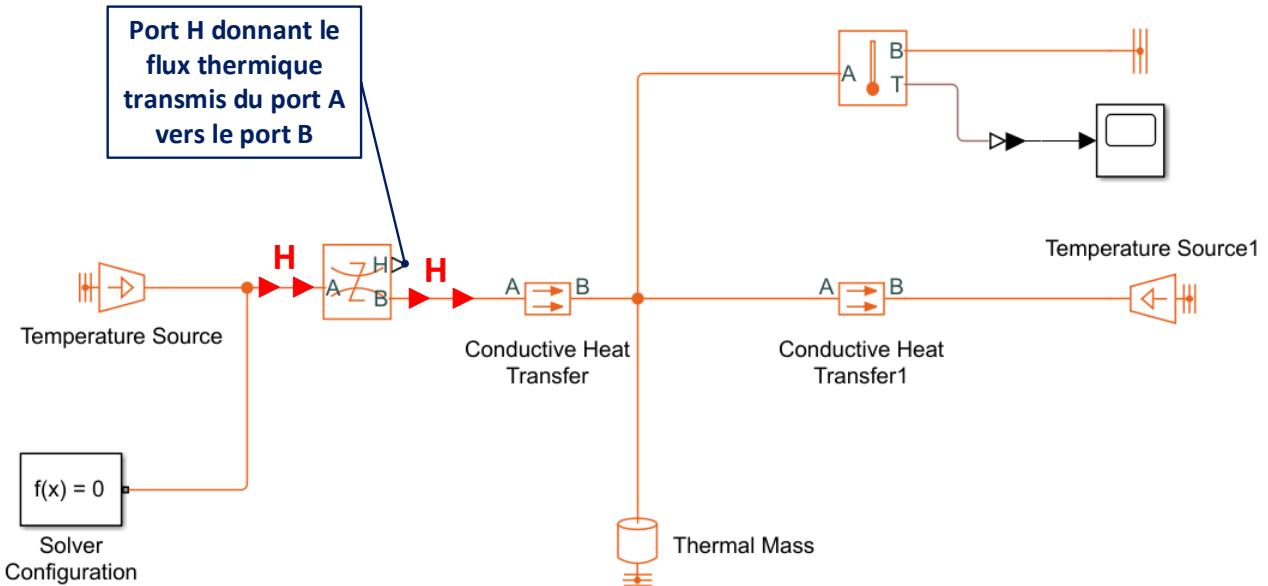
Figure 238 : les ports du capteur de flux thermique

Pour mettre en place le capteur de flux thermique, il faut donc ouvrir une connexion et placer le capteur en série.

Port A : port de type (PCP) qui doit être relié à l'un des points correspondant à l'ouverture de la connexion.

Port B : port de type (PCP) qui doit être relié à l'autre point correspondant à l'ouverture de la connexion.

Port H : port de type **Physical Signal (PS)** donnant le flux thermique qui traverse la connexion.

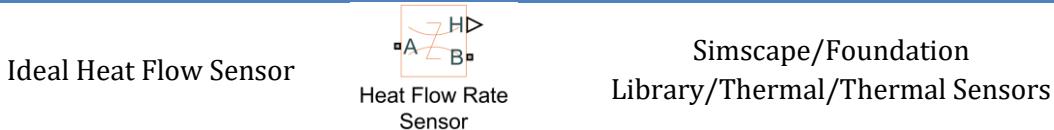


FLUX THERMIQUE = grandeur de type « Through »
Le flux thermique se mesure toujours en série dans une connexion

Figure 239 : mesure d'une grandeur physique de type flux thermique

Si le flux thermique est transmis du port A vers le port B, le capteur mesure un flux thermique positif. En cas d'inversion des ports A et B du capteur de flux thermique, le signe de la mesure est inversé.

Paramétrage du bloc Ideal Heat Flow Sensor



Ce composant permet de mesurer le flux thermique qui traverse une connexion du domaine thermique.
Ce bloc ne nécessite aucun paramétrage.

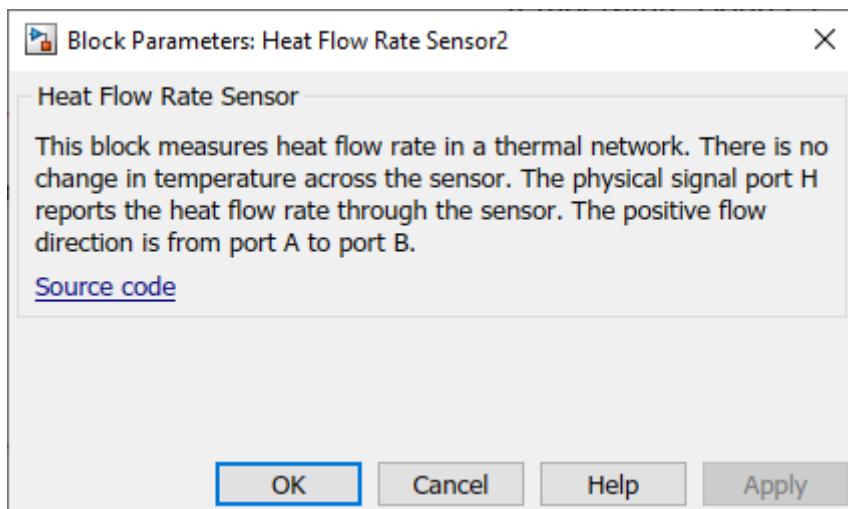


Figure 240 : paramétrage du bloc Ideal Heat Flow Sensor

Nous allons planter trois capteurs de flux thermique dans le modèle.

Ouvrir le fichier «chauffage_barre_3.slx» ou **Glisser/Déposer** les composants de la Figure 241 dans le modèle afin obtenir la configuration de la Figure 242.

Capteur de température		Simscape/Foundation Library/ Thermal/Thermal Sensors
Conversion d'un signal physique en signal numérique		Simscape/Utilities
Afficheur de signaux numériques		Simulink/Commonly Used Blocks

Figure 241 : les composants associés au capteur de flux thermique

Dans notre exemple, nous souhaitons mesurer le flux thermique qui circule à chaque extrémité de la barre ainsi que celui absorbé ou restitué par la masse thermique. Les connexions ont été ouvertes afin de pouvoir planter les trois capteurs de flux en série.

Le port **H** permet de transmettre le signal physique contenant le flux thermique qui traverse la connexion. Une inversion des ports A et B du capteur entraîne une inversion du signe de la grandeur physique mesurée.

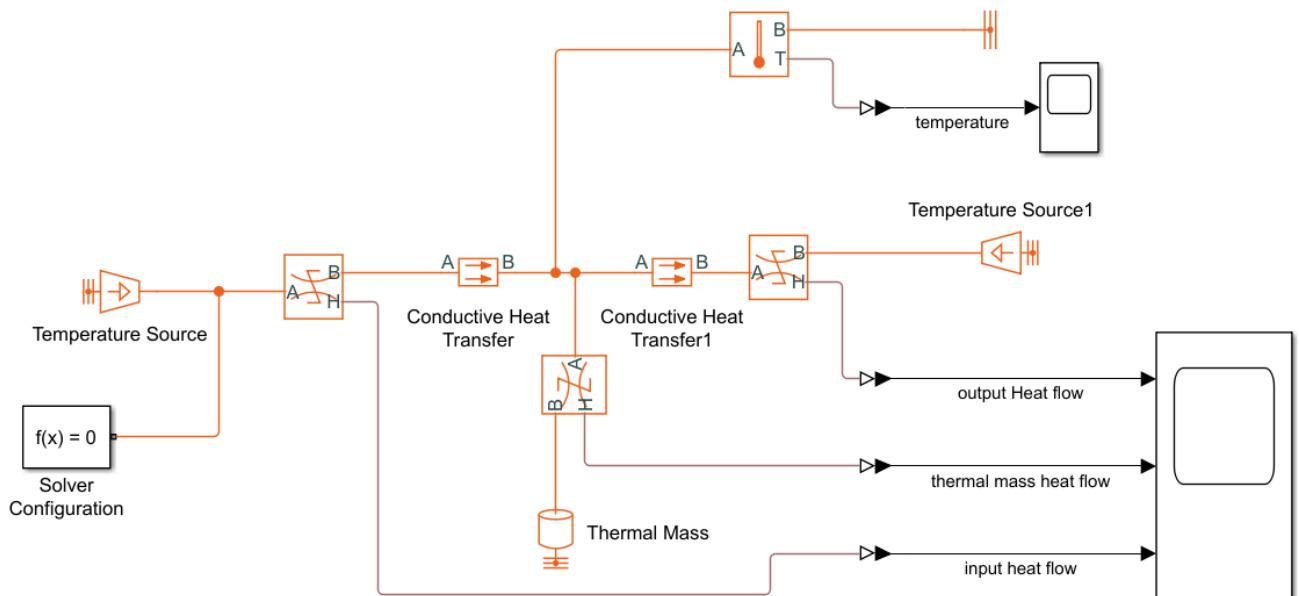


Figure 242 : implantation des capteurs de flux thermique dans le modèle

Paramétrage du bloc PS-Simulink Converter

PS – Simulink
Converter

PS-Simulink
Converter

Simscape/Utilities

Ce composant permet de convertir le signal physique (PS) issu d'une grandeur mesurée en signal **Simulink** (S) 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. Si l'unité n'est pas spécifiée par l'utilisateur (valeur par défaut 1 ou inherit), la grandeur sera affichée dans le scope en unités SI.

Le menu déroulant propose un choix d'unités, il est possible de taper l'unités de son choix si elle n'est pas proposée par défaut.

Pour les blocs **PS-Simulink Converter** qui mesure des flux thermiques, choisir **J/s ou W** (Joule par seconde ou Watt) dans la case « Output Signal Unit »

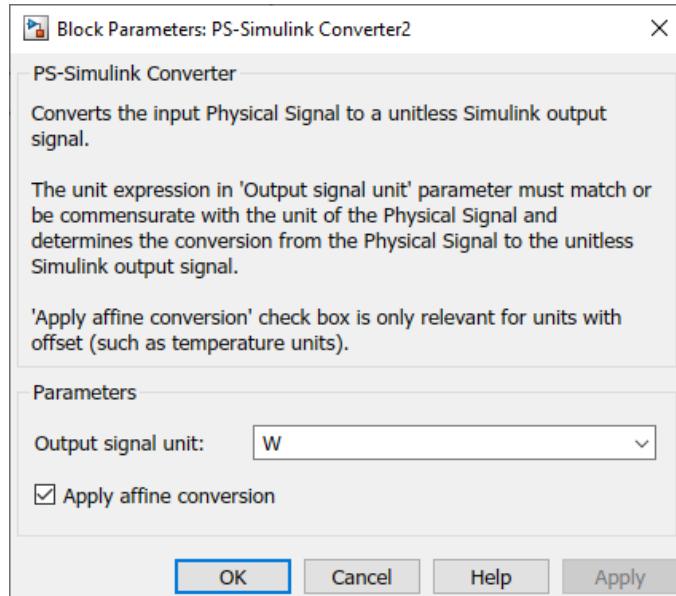


Figure 243 : paramétrage du bloc PS – Simulink Converter

13. Définition des conditions initiales du modèle

La phase de modélisation étant maintenant terminé, MATLAB doit résoudre le modèle. Afin de pouvoir mener les calculs, il faut préciser les conditions initiales du modèle, c'est-à-dire les valeurs que prendront les variables de notre modèle à l'instant $t=0$, instant initial de la simulation. Si rien n'est précisé par l'utilisateur, toutes les variables auront, par défaut, une valeur nulle en début de simulation. Si cette configuration correspond à la réalité de ce que l'on souhaite simuler, aucune opération n'est requise à ce niveau. Par contre si des variables ont des valeurs différentes de 0 en début de simulation, leurs valeurs devront être précisées.

Dans notre exemple, à $t=0$, la température de la barre est de 20°C. Cette information doit être indiquée dans les condition initiales de la simulation.

Pour cela double cliquer sur le bloc **Thermal Mass**, et sélectionner l'onglet **Variables**.

Cocher la case température et indiquer une température initiale de **20 degC** comme indiqué sur la Figure 244.

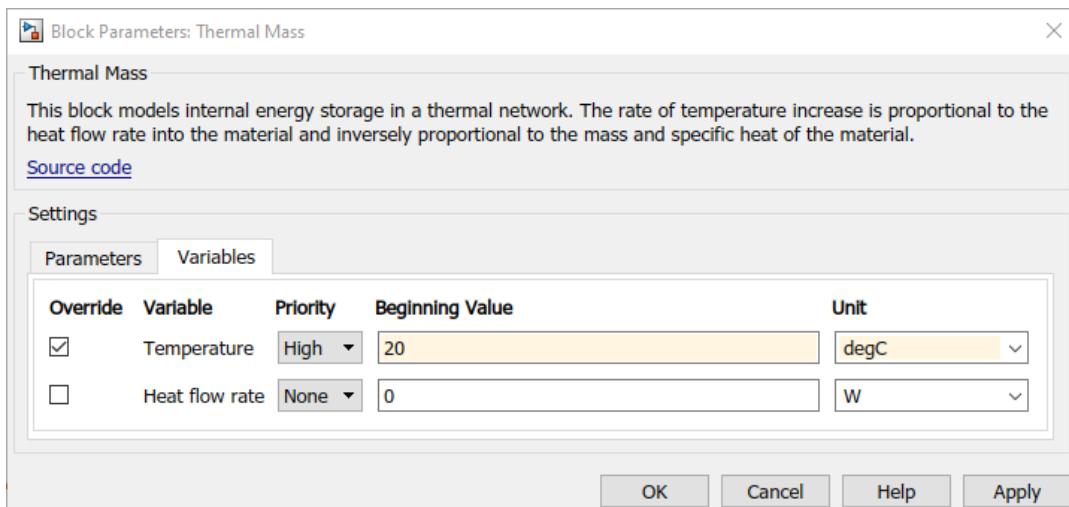


Figure 244 : spécification de la température initiale de la barre

14. Réglage du temps de simulation

La phase de modélisation étant terminée et les conditions initiales définies, MATLAB doit résoudre le modèle et une méthode de résolution numérique doit être mise en œuvre. L'utilisateur doit définir le temps de simulation et lancer le calcul (Figure 245).

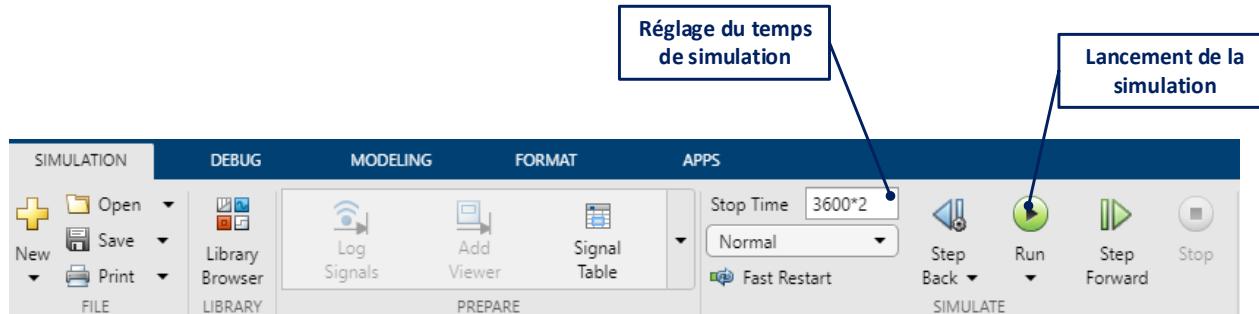


Figure 245 : configuration des paramètres du solveur

Nous choisirons un temps de simulation de **2 heures** soit **3600*2 s**. Le temps de simulation doit être spécifié directement dans la barre d'outil (Figure 245).

15. Lancement de la simulation et exportation des résultats

Le fichier « **chauffage_barre_3.slx** » contient le modèle entièrement paramétré et prêt à être exécuté.

Le temps de simulation étant réglé sur 3600*2 secondes, **lancer** la simulation en appuyant sur dans la barre de commande Figure 245.

Ouvrir les scopes donnant la température au milieu de la barre et les flux thermiques. Pour mettre en forme les courbes dans les scopes, vous pouvez vous rendre à l'annexe paramétrage des scopes en fin d'ouvrage.

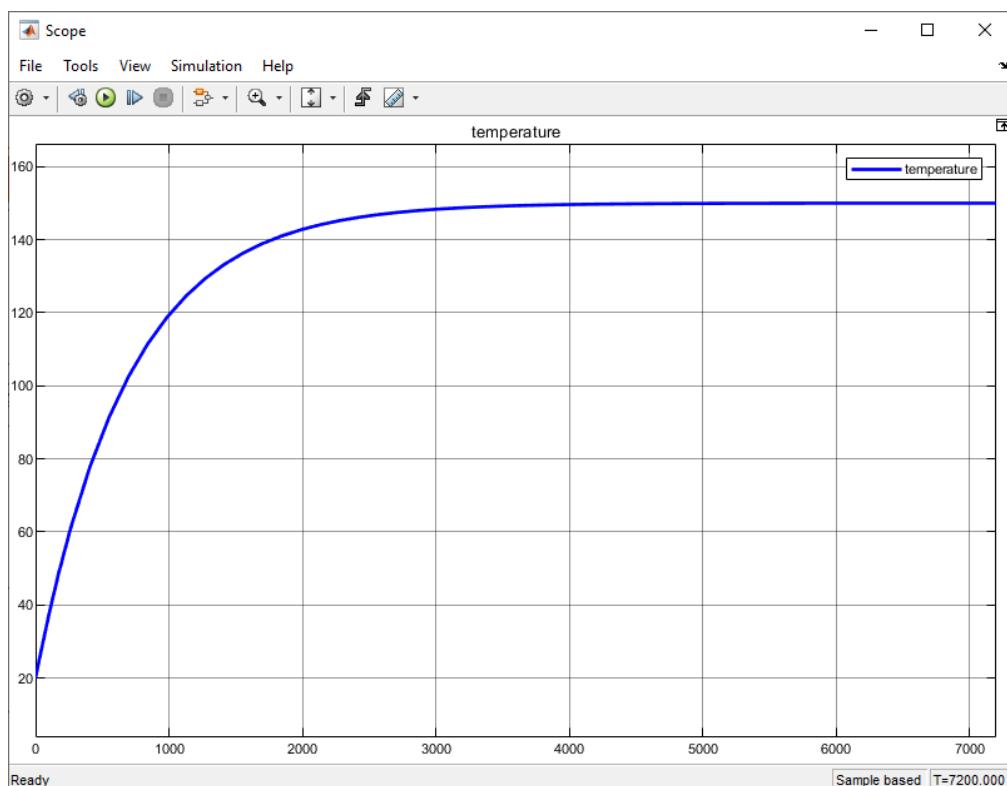


Figure 246 : température au milieu de la barre en fonction du temps

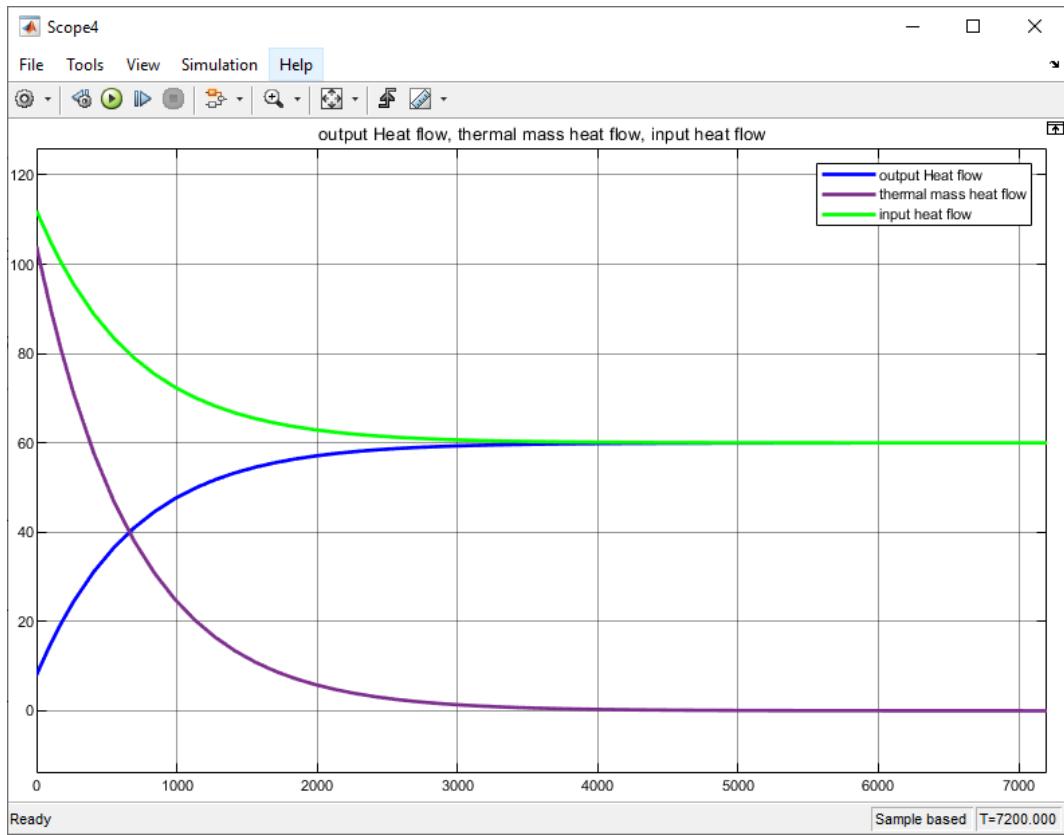


Figure 247 : flux thermiques mesurés sur le modèle

16. Interprétation des résultats de la simulation

La Figure 246 nous permet de voir que le régime permanent s'établit au bout de $t_{5\%}=2016 \text{ s}$ avec une température au centre de la barre $T_A=150^\circ\text{C}$.

Analysons maintenant les mesures de flux thermique que l'on peut observer sur la Figure 247.

Le flux thermique H_i à l'extrémité chaude de la barre se divise en deux flux thermiques, H_{tm} vers la masse thermique et H_o vers l'extrémité froide de la barre. Il est aisément de vérifier la relation $H_i=H_{tm}+H_o$.

Le flux thermique H_{tm} tend logiquement vers 0 en régime permanent et l'inertie thermique de la masse n'a alors plus d'influence.

Le flux thermique H qui traverse la barre en régime permanent s'établit à $H=60 \text{ W}$.

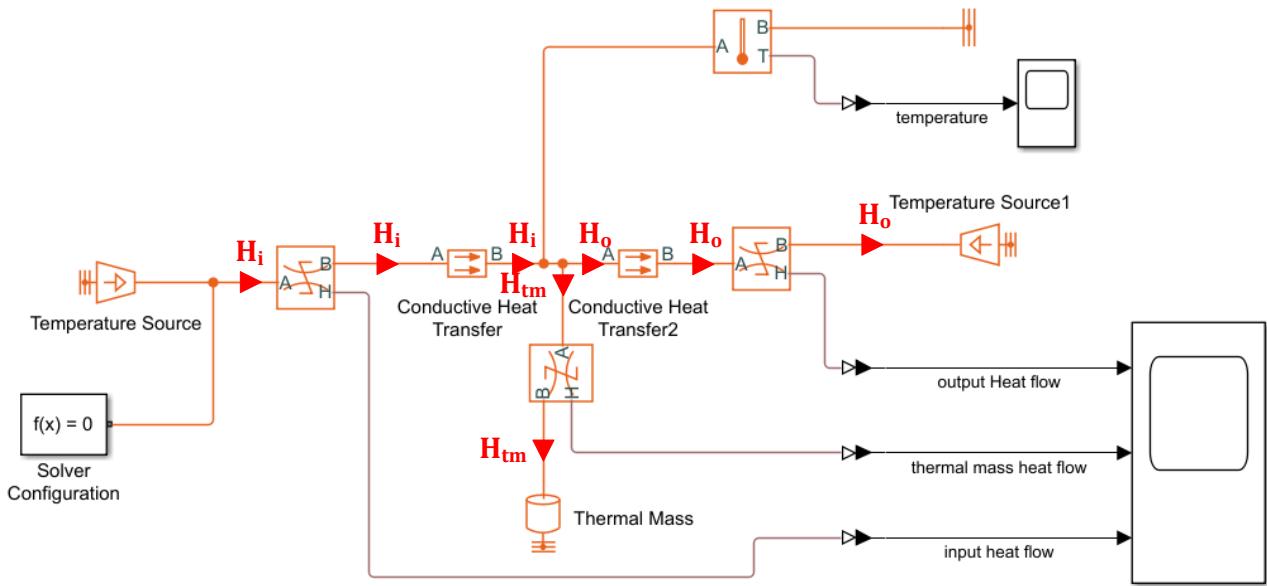


Figure 248 : circulation des flux thermiques dans les différentes parties du modèle

Il est très simple de modifier le modèle pour vérifier la loi des nœuds pour le flux thermique (variable Through) : $H_i = H_{tm} + H_o$.

Le fichier « **chauffage_barre_4.slx** » contient le modèle de la Figure 249.

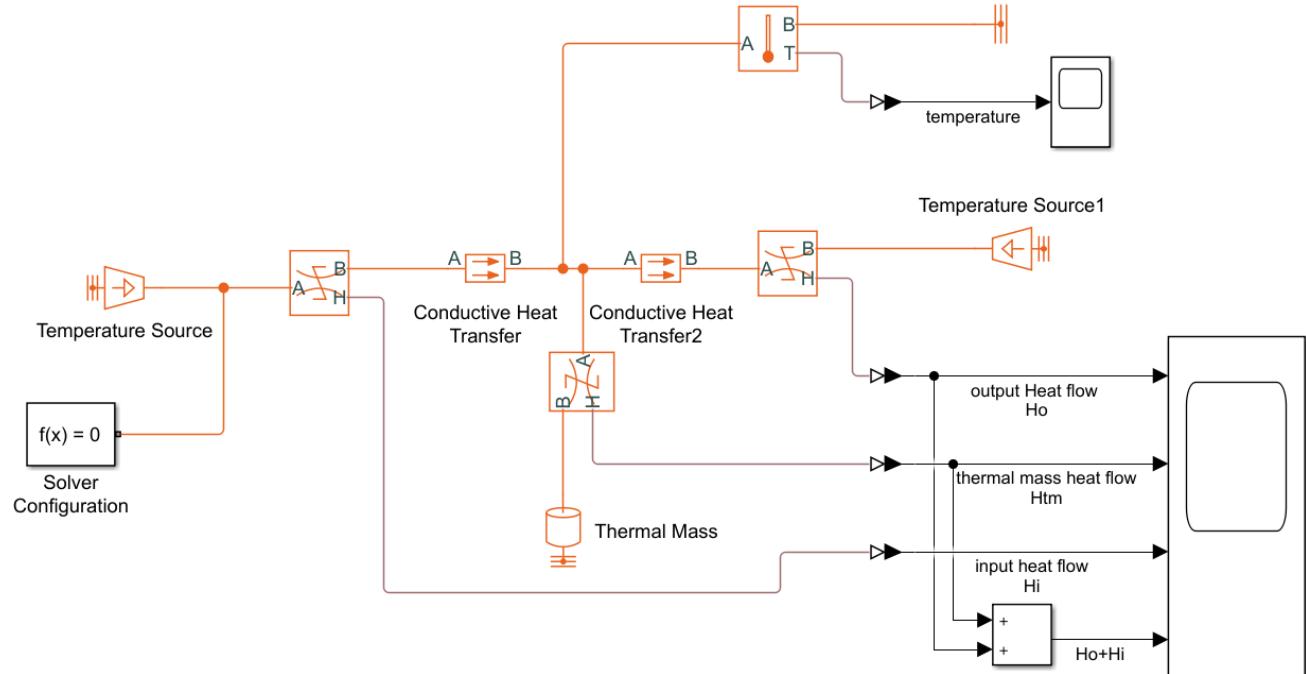


Figure 249 : vérification de la conservation du flux thermique dans le modèle

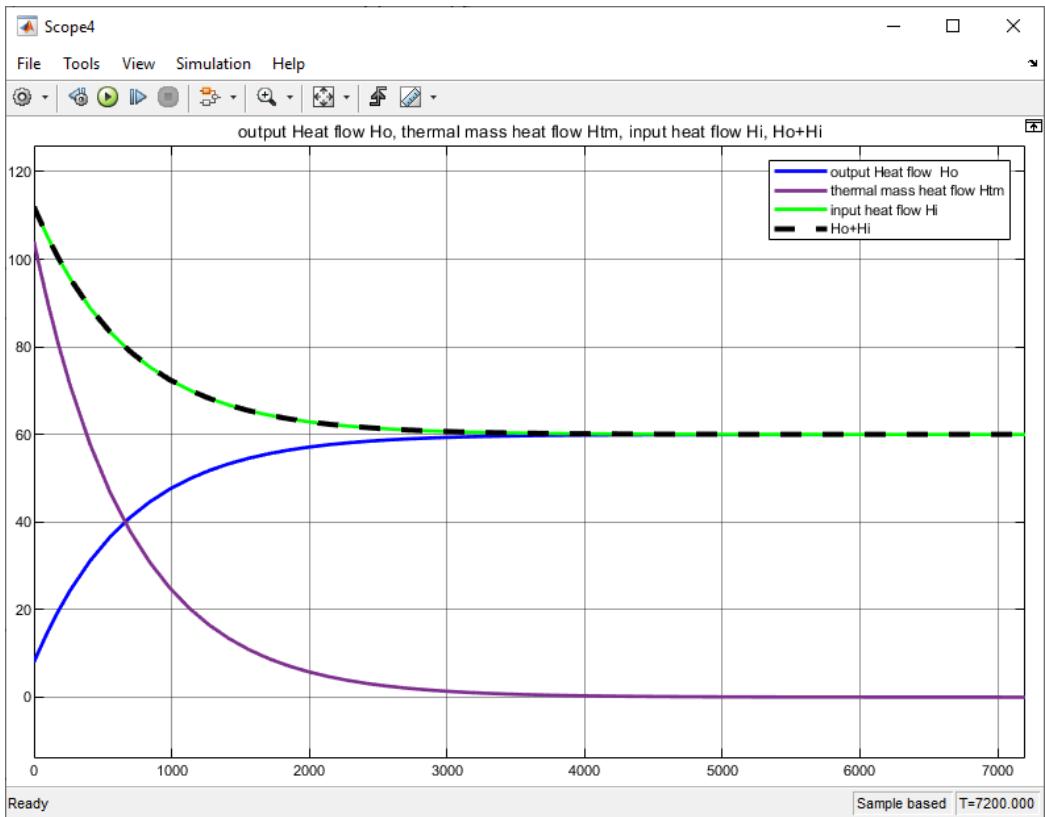


Figure 250 : visualisation de la conservation du flux thermique dans le scope

Prise en compte des pertes

On considère maintenant que la barre n'est plus isolée mais laissée à l'air libre. Il se produit alors des échanges de chaleur par convection sur toute la surface extérieure de la barre. Le coefficient d'échange de chaleur par convection entre l'air et la barre d'acier vaut $k=32 \text{ W}/(\text{m}^2\cdot\text{K})$ pour les conditions de l'expérience. La surface d'échange correspond à la surface extérieure du cylindre $S_{ext} = \pi DL = 201 \text{ cm}^2$

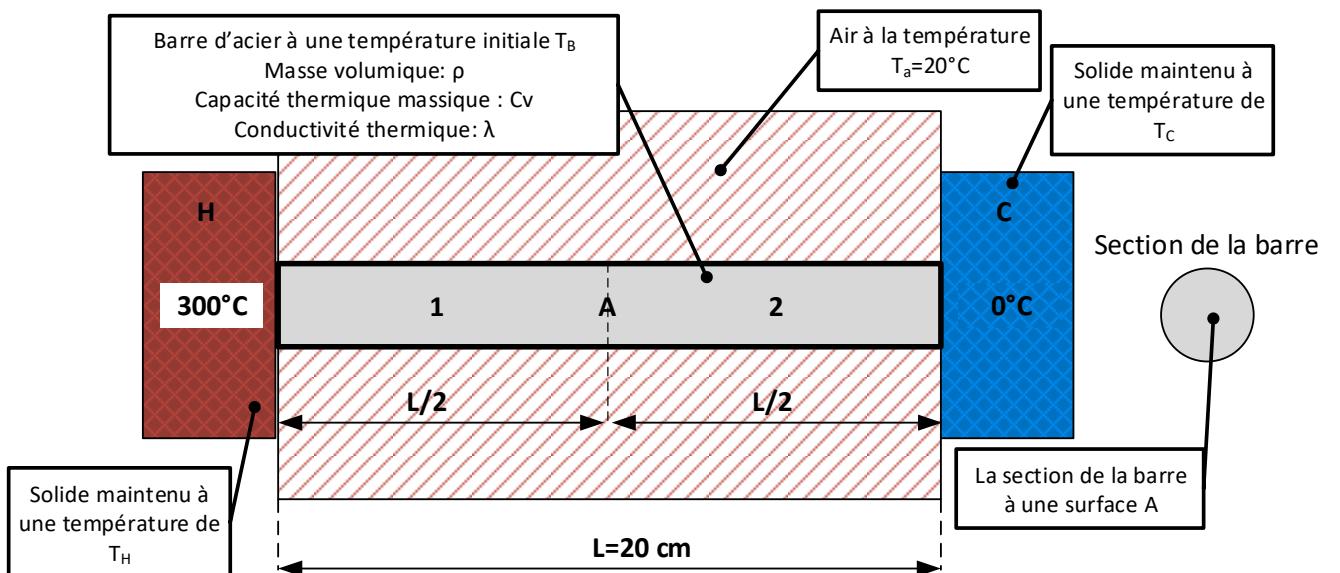


Figure 251 : chauffage d'une barre par conduction non isolée

Afin de modéliser l'échange de chaleur par convection entre la surface extérieure de la barre et l'air, nous allons ajouter un bloc **Convective Heat Transfer** et un bloc **Temperature Source** pour modéliser la température de l'air qui reste constante (Figure 252).

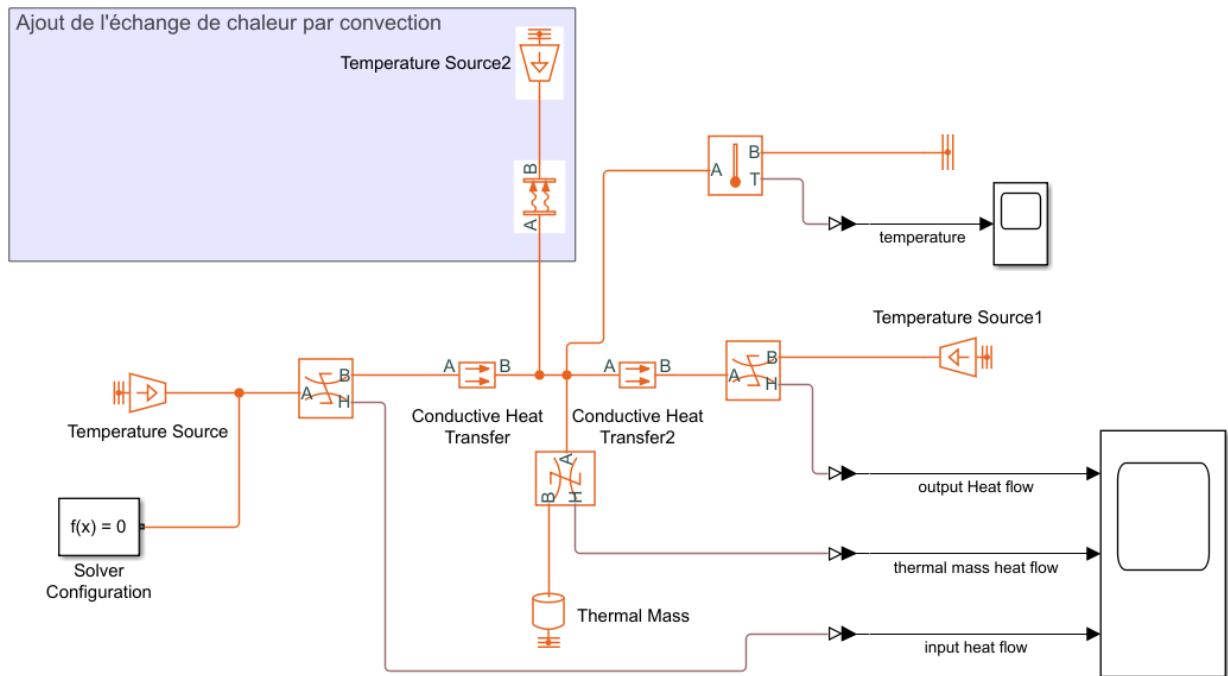
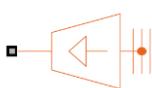


Figure 252 : modélisation de l'échange de chaleur par convection

Le fichier « **chauffage_barre_conv_1.slx** » contient le modèle de la Figure 252.

Paramétrage du bloc Ideal Temperature Source

Ideal Temperature
Source



Simscape/Foundation
Library/Thermal/Thermal Elements

Ce bloc permet de modéliser la température de l'air qui reste constante à 20°C.

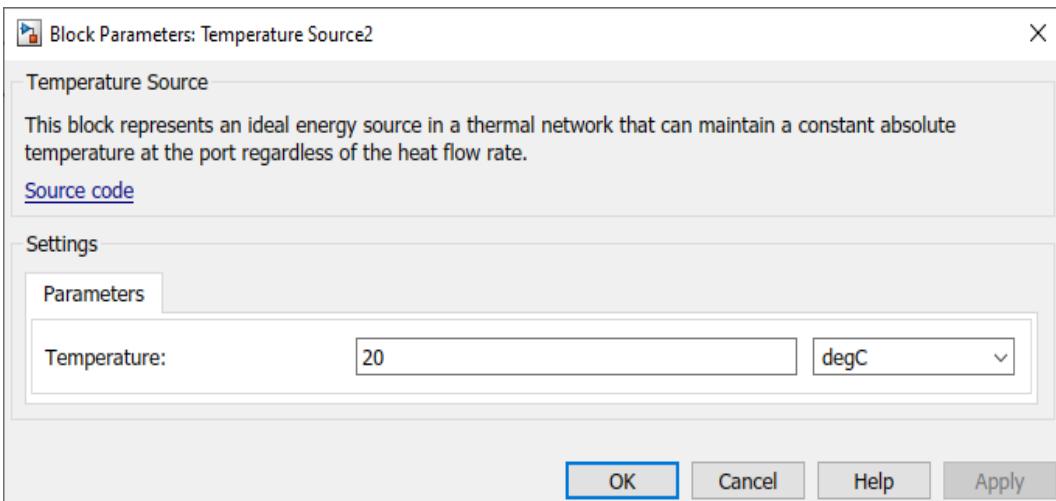
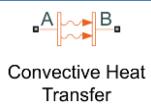


Figure 253 : paramétrage du bloc Temperature Source

Paramétrage du bloc Convective Heat Transfer

Convective Heat
Transfert



Simscape/Foundation
Library/Thermal/Thermal Elements

Ce bloc permet de définir les paramètres de l'échange de chaleur par convection. Les paramètres à renseigner sont :

Area : la surface correspondant à la surface d'échange thermique par convection. Dans notre exemple cela correspond à la surface extérieure du cylindre $S_{ext} = \pi DL = 201 \text{ cm}^2$.

Heat Transfer Coefficient : il faut renseigner ici le coefficient de transfert thermique par convection entre l'air et l'acier pour les condition de l'expérience, $k=32 \text{ W}/(\text{m}^2 \cdot \text{K})$.

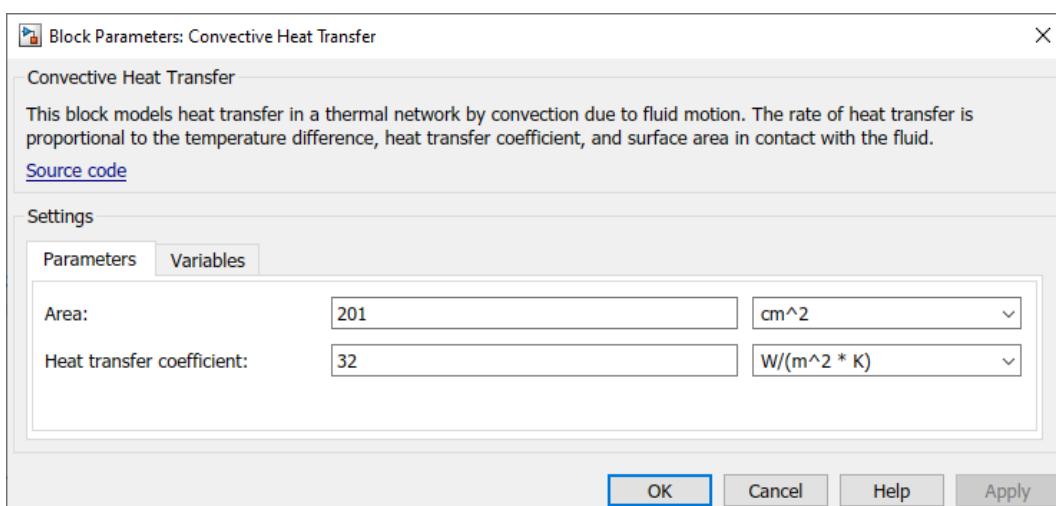


Figure 254 : paramétrage du bloc Convective Heat Transfer

Le temps de simulation étant réglé sur $3600*2$ secondes, lancer la simulation en appuyant sur dans la barre de commande Figure 245. Ouvrir les scopes donnant la température au milieu de la barre et les flux thermiques. Pour mettre en forme les courbes dans les scopes, vous pouvez vous rendre à l'annexe paramétrage des scopes en fin d'ouvrage.

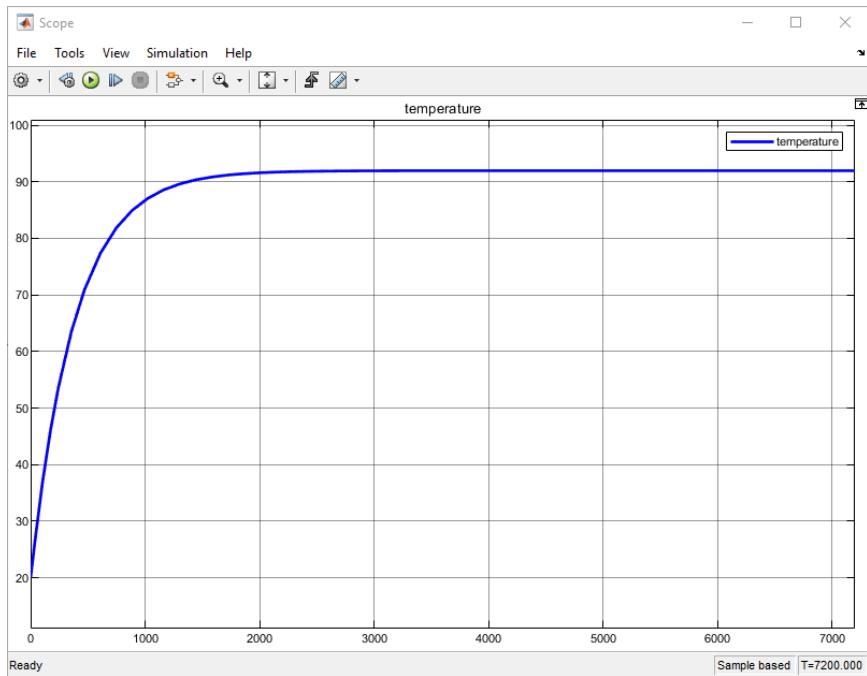


Figure 255 : température au milieu de la barre en fonction du temps avec pertes par convection

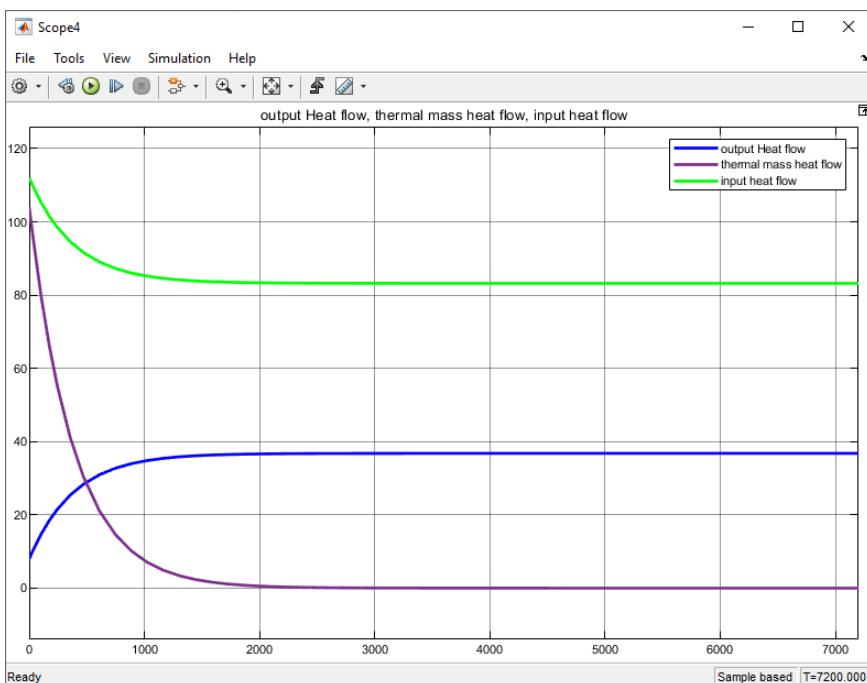


Figure 256 : flux thermiques mesurés sur le modèle avec pertes par convection

Interprétation des résultats de la simulation

La Figure 255 nous permet de voir que le régime permanent s'établit au bout de $t_{5\%}=1053 \text{ s}$ avec une température au centre de la barre $T_A=92^\circ\text{C}$. Nous constatons que la température de la barre en régime permanent est bien inférieure à celle obtenue lorsque la barre était isolée.

Nous pouvons remarquer que le flux entrant dans la barre au niveau de la source de température de 300°C (input heat flow) n'est plus égal au flux qui arrive au niveau de la source de température à 0°C (output heat flow). En effet le flux thermique de convection qui correspond aux pertes est maintenant pris en compte dans le modèle.

Pour évaluer ce flux thermique, il est possible d'ajouter un capteur de flux thermique dans notre modèle (Figure 257).

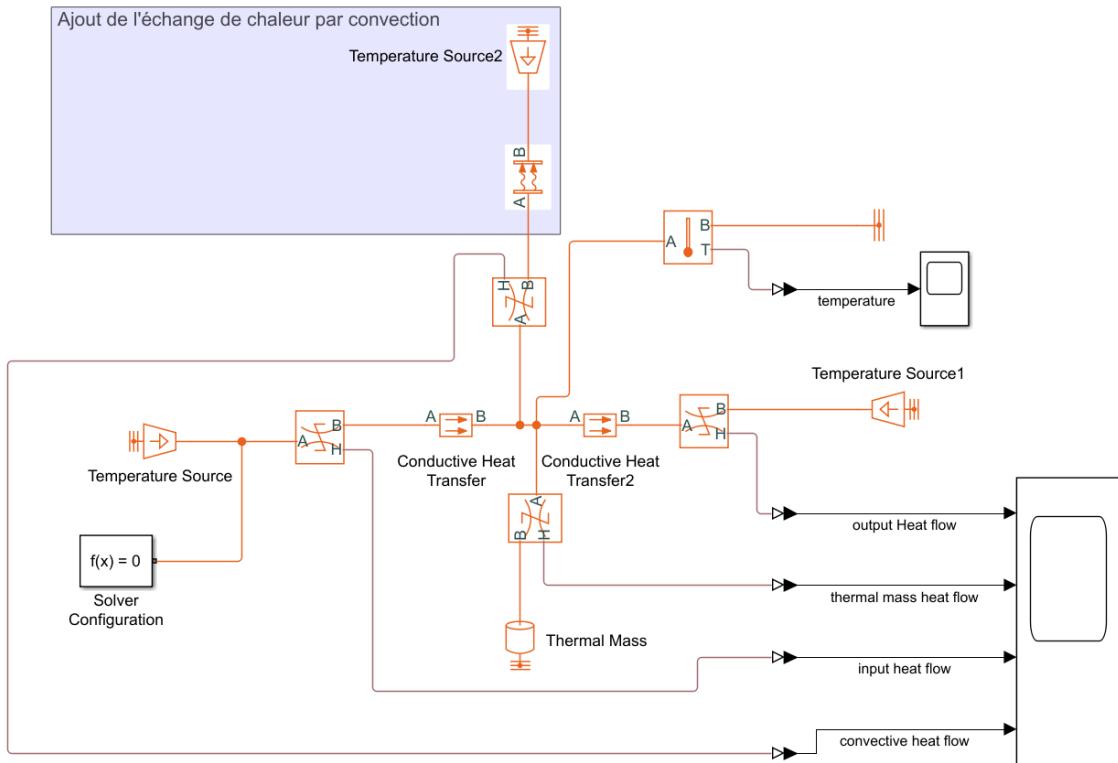


Figure 257 : ajout d'un capteur pour mesure le flux thermique de convection

Le fichier « **chauffage_barre_conv_2.slx** » contient le modèle de la Figure 257.

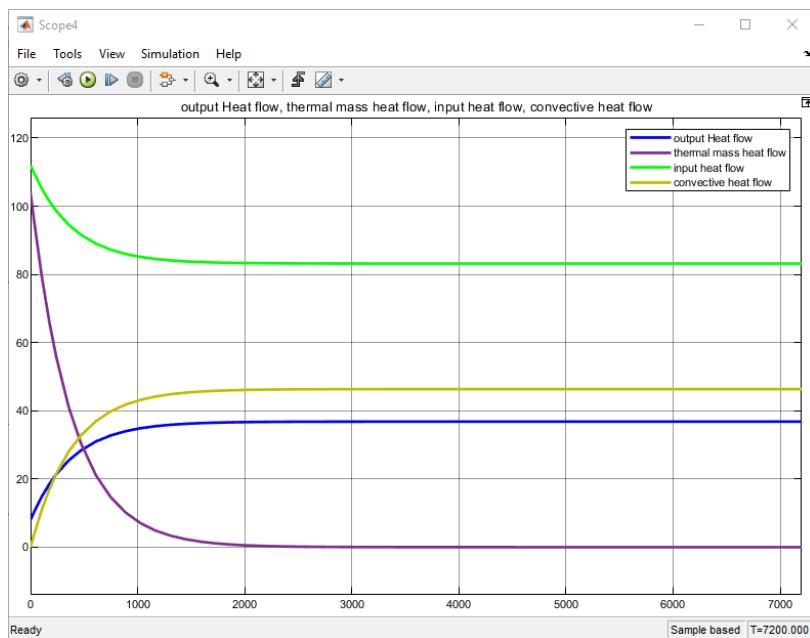


Figure 258 : visualisation de la conservation du flux thermique

Nous pouvons vérifier aisément sur le Figure 258 la conservation du flux thermique. Le flux thermique à l'entrée est bien égal à la somme de tous les autres flux.

17. Rappels théoriques sur les échanges thermiques à travers une paroi d'un bâtiment

Les échanges thermiques à travers une paroi d'un bâtiment sont de différentes natures. Des échanges par convection ont lieu sur la surface extérieure et sur la surface intérieure de la paroi. Des échanges thermiques par conduction ont lieu au travers des différentes couches de la paroi.

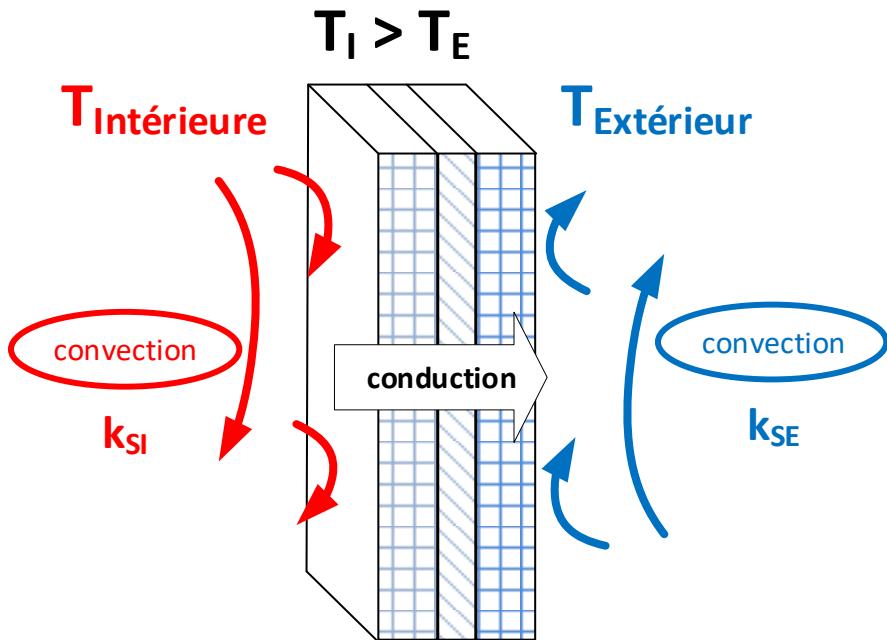


Figure 259 : les échanges thermiques à travers une paroi

Modélisation des échanges par convection sur la surface de la paroi

Il faudra utiliser le bloc **Convective Heat Transfer** en utilisant les valeurs du coefficient d'échange thermique par convection k regroupées dans les tableaux de la Figure 260 et de la Figure 261.

Paroi opaque Coefficient de transfert thermique par convection			
	k_{SI} en $\text{W}/(\text{m}^2.\text{K})$	k_{SE} en $\text{W}/(\text{m}^2.\text{K})$	
Paroi verticale	7.7	25	
Paroi horizontale (flux ascendant)	10	25	
Paroi horizontale (flux descendant)	5.9	25	

Figure 260 : coefficient de transfert thermique par convection

Paroi vitrée		
Coefficient de transfert thermique par convection		
	k_{SI} en W/(m ² .K)	k_{SE} en W/(m ² .K)
Paroi verticale (flux horizontal)	7.7	25
Paroi horizontale (flux ascendant)	10	25

Figure 261 : coefficient de transfert thermique par convection

Modélisation des échanges par conduction au travers des différentes couches de la paroi

Il faudra utiliser le bloc **Conductive Heat Transfer** en utilisant les valeurs de la conductivité thermique des matériaux usuels regroupées dans le tableau de la Figure 262.

Conductivité thermique			
Matériau	λ en W/(m.K)	Matériau	λ en W/(m.K)
Béton plein	1.75	Air	0.028
Brique pleine	0.74	Liège	0.05
Enduit ciment	1.15	Aluminium	230
Plâtre	0.5	Cuivre	380
Bois	0.23	Acier	50
Laine de verre	0.04	Eau	0.66

Figure 262 : conductivité thermique des matériaux usuels

18. Modélisation avec Simscape des échanges par conduction et convection dans une paroi monocouche

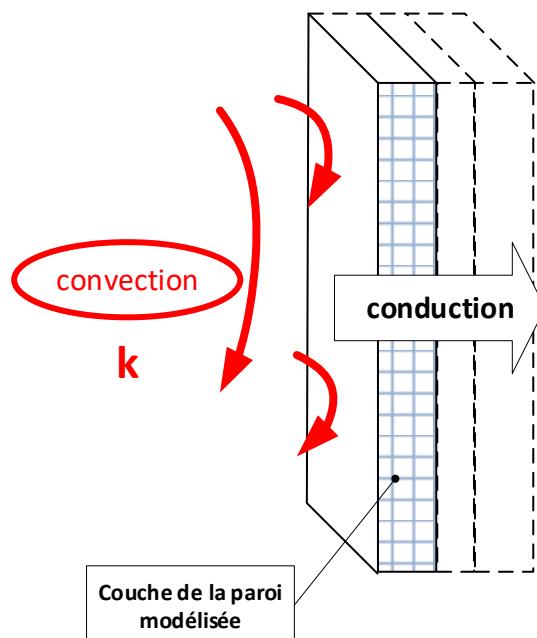


Figure 263 : modélisation d'une couche de la paroi

Il est possible de modéliser les échanges thermiques de deux manières différentes :

- En utilisant les blocs **Conductive Heat Transfer**, **Convective Heat Transfer**, **Radiative Heat Transfer**
- En utilisant le bloc **Thermal Resistance** qui introduit le concept de résistance thermique équivalente

Utilisation des blocs Conductive Heat Transfer, Convective Heat Transfer

La Figure 264 montre la correspondance entre les éléments de modélisation de Simscape et les phénomènes physiques intervenants dans les transferts thermiques au travers d'une couche de la paroi.

Caractéristiques de la couche de la paroi:

- Surface A
- Épaisseur e
- Conductivité thermique λ
- Chaleur spécifique C
- Masse volumique ρ

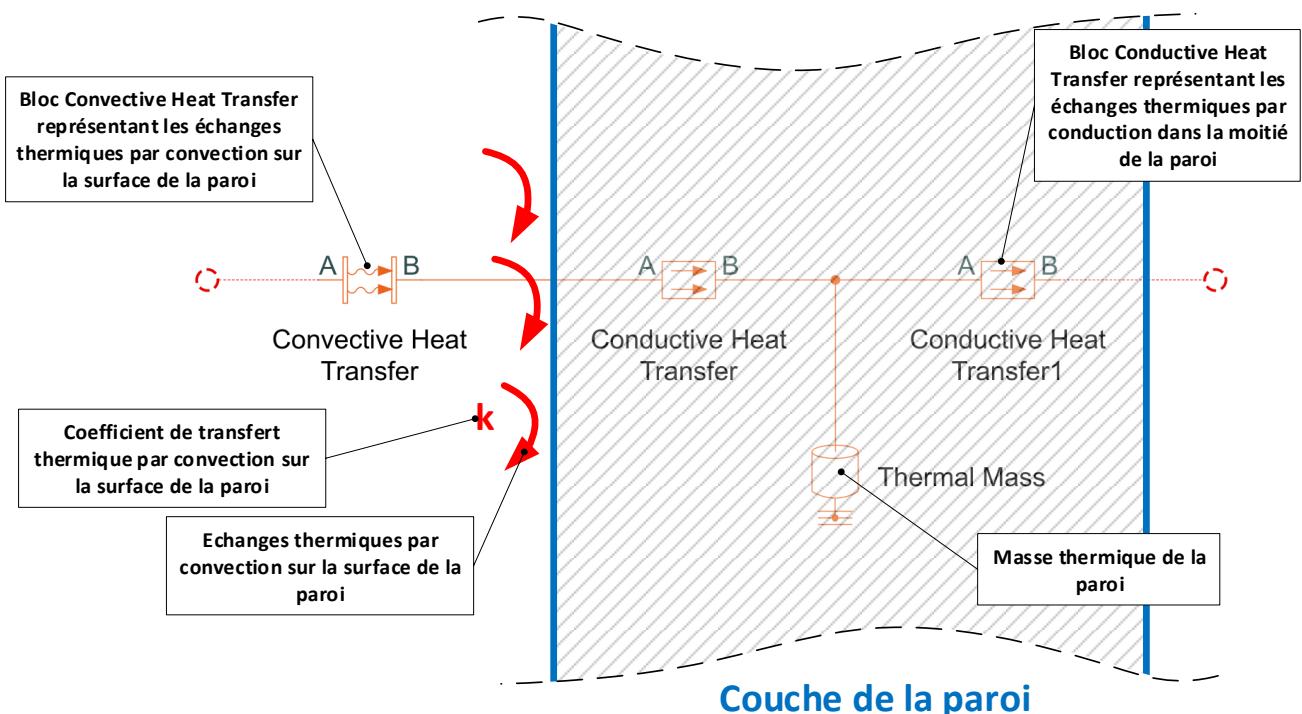


Figure 264 : modélisation Simscape d'une couche de la paroi

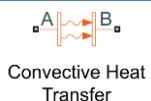
Les échanges thermiques par convection sur la surface de la paroi sont modélisés à l'aide du bloc **Convective Heat Transfer**.

Les échanges thermiques par conduction sont représentés par deux blocs **Conductive Heat Transfer** paramétrés de manière identique :

La masse thermique de la paroi est modélisée à l'aide du bloc **Thermal Mass**.

Paramétrage du bloc Convective Heat Transfert

Convective Heat
Transfert



Convective Heat
Transfer

Simscape/Foundation

Library/Thermal/Thermal Elements

Ce bloc permet de définir les paramètres de l'échange de chaleur par convection entre l'air et la surface de la paroi.

Area : Surface d'échange thermique par convection. Pour l'exemple de la paroi cette surface correspond à la surface **S** de la paroi

Heat Transfer Coefficient : le coefficient de transfert thermique par convection entre l'air et la paroi est **k**.

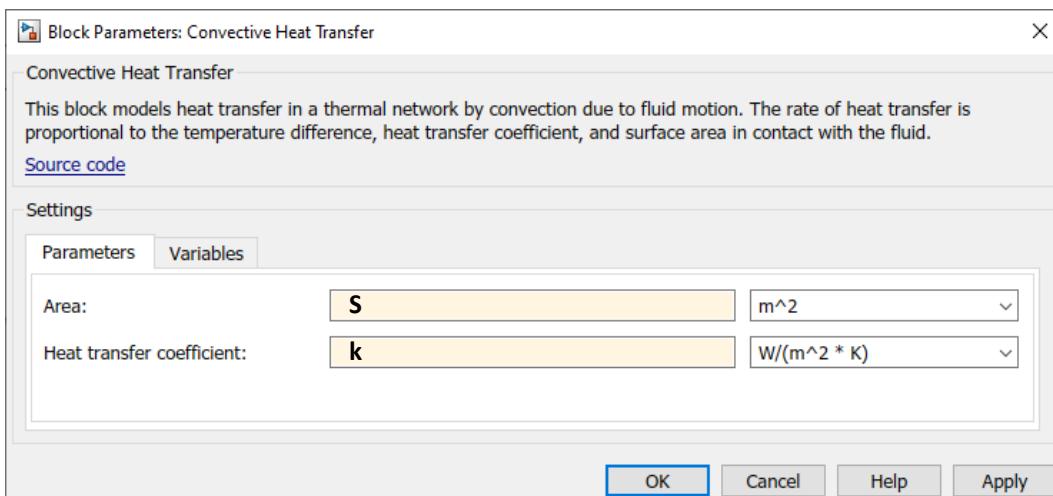


Figure 265 : paramétrage du bloc Convective Heat Transfer

Paramétrage du bloc Conductive Heat Transfert

Conductive Heat
Transfert



Conductive Heat
Transfer

Simscape/Foundation

Library/Thermal/Thermal Elements

Ce bloc permet de définir les paramètres de l'échange de chaleur par conduction au travers d'une moitié de la couche de la paroi.

Area : Surface correspondant à l'échange thermique. Dans cet exemple il s'agit de la surface de la paroi **S**.

Thickness : la longueur de l'éléments traversé par le flux thermique. Dans cet exemple, il s'agit de l'épaisseur de la paroi **e/2**.

Thermal Conductivity : il faut renseigner ici la conductivité thermique du matériau constitutif de la paroi **λ** .

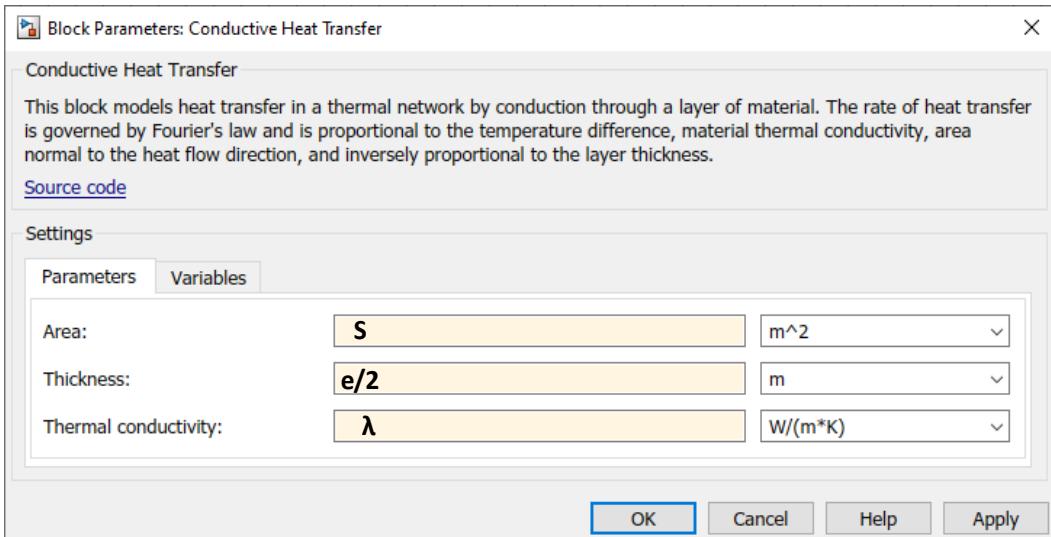
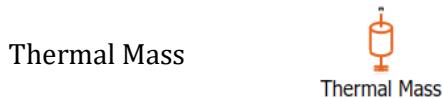


Figure 266 : paramétrage du bloc Conductive Heat Transfer

Paramétrage du bloc Thermal Mass



Simscape/Foundation
Library/Thermal/Thermal Elements

Ce bloc permet de modéliser la masse thermique de la couche de la paroi.

Mass : la masse de l'élément considéré. Dans la masse correspond à la masse de la couche de la paroi : **S.e.p**

Specific Heat : il faut renseigner ici la capacité thermique massique du matériau considéré **C**.

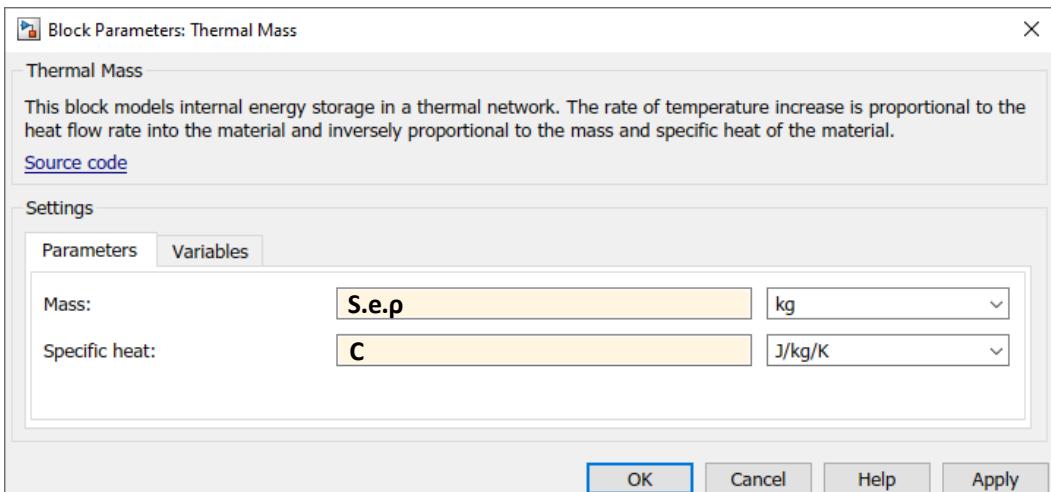


Figure 267 : paramétrage du bloc Thermal Mass

Il est ainsi possible de modéliser les différentes couches de la paroi. Il conviendra de repérer le type d'échange thermique qui intervient au niveau des surfaces (convection ou conduction).

19. Utilisation du bloc Thermal Résistance

La Figure 268 montre la correspondance entre les éléments de modélisation de Simscape et les phénomènes physiques intervenants dans les transferts thermiques au travers d'une couche de la paroi.

Caractéristiques de la couche de la paroi:

- Surface A
- Épaisseur e
- Conductivité thermique λ
- Chaleur spécifique C
- Masse volumique ρ

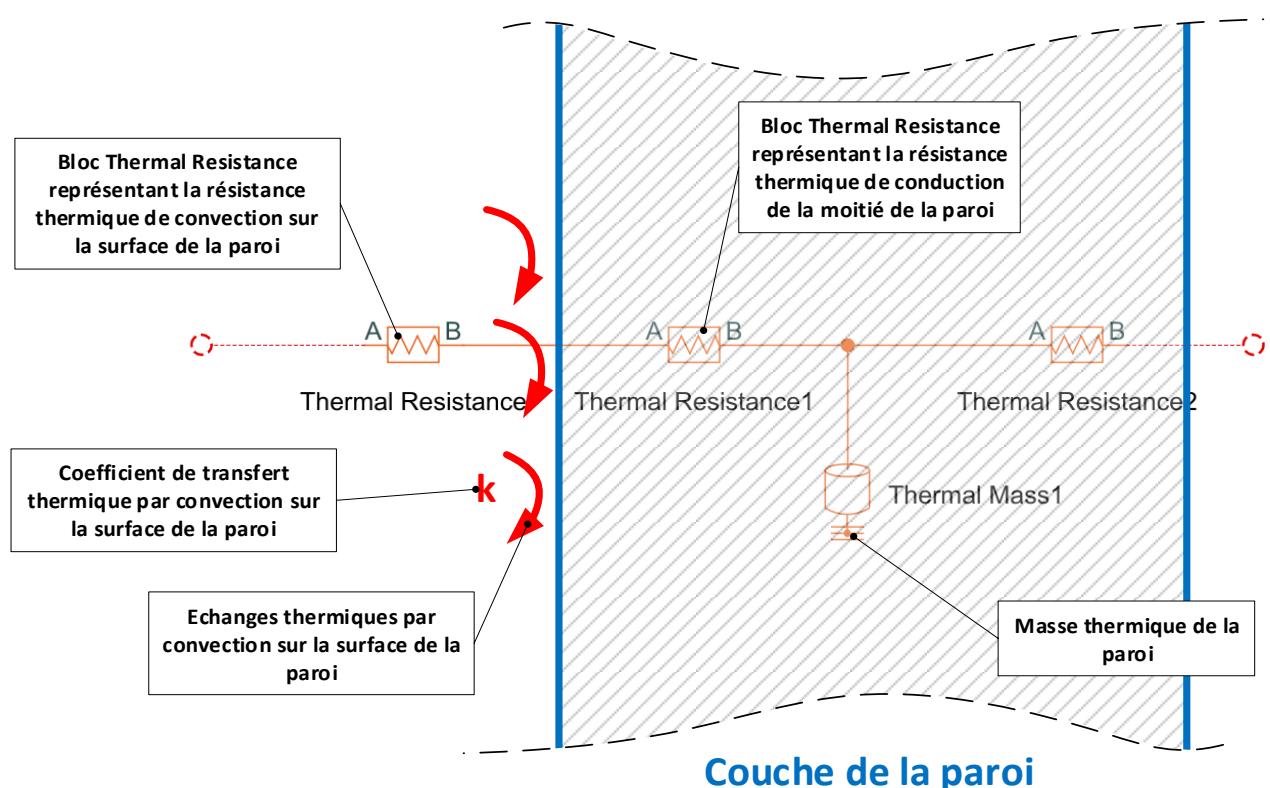


Figure 268 : modélisation Simscape d'une couche de la paroi

Le calcul des résistances thermiques de conduction et de convection équivalentes est détaillé en page 196.

Paramétrage du bloc Thermal Resistance représentant la résistance thermique de convection de la surface de la paroi

Paramétrage du bloc Thermal Resistance pour la convection

Thermal Resistance  Simscape/Foundation
Thermal Resistance Library/Thermal/Thermal Elements

La résistance thermique de convection est donnée par $R_{conv} = \frac{1}{k.S}$ où S représente la surface d'échange

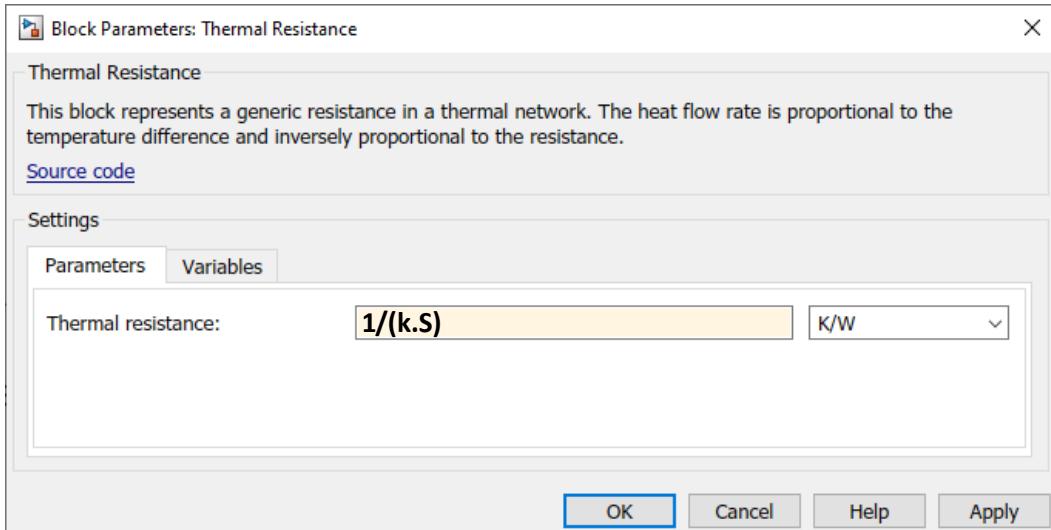


Figure 269 : Paramétrage du bloc Thermal Resistance pour la convection

Paramétrage du bloc Thermal Resistance représentant la résistance thermique de conduction d'une moitié de la couche de la paroi.

Paramétrage du bloc Thermal Resistance pour la conduction

Thermal Resistance  Simscape/Foundation
Thermal Resistance Library/Thermal/Thermal Elements

La résistance thermique de conduction de la moitié d'une couche de la paroi est donnée par
 $R_{conv} = \frac{(e/2)}{\lambda.S}$

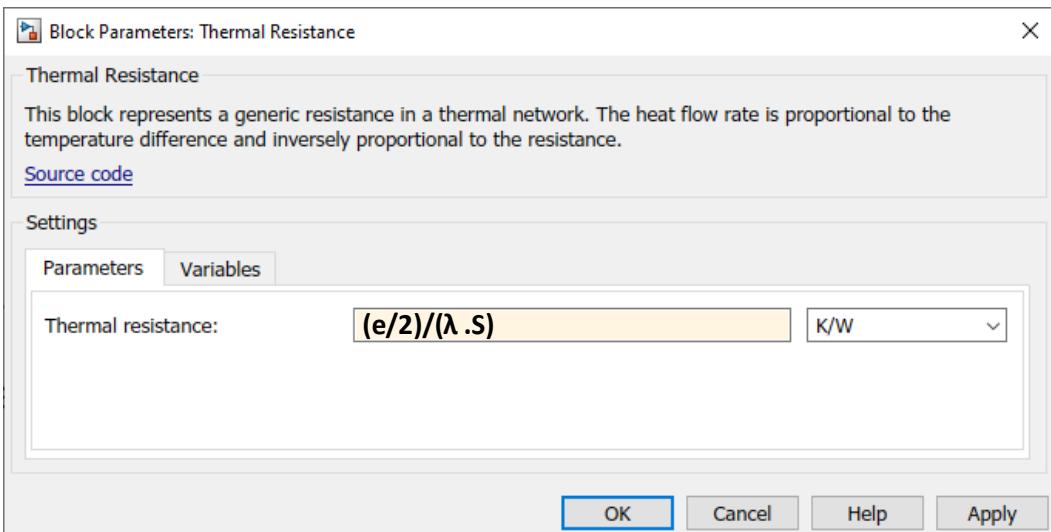


Figure 270 : Paramétrage du bloc Thermal Resistance pour la conduction

Ouvrir le fichier « **couche_0.slx** ». Ce fichier contient la modélisation d'une couche d'une paroi. Le modèle propose les deux types de modélisation, en utilisant les blocs de transfert thermique par conduction et convection et la modélisation équivalente en utilisant le concept de résistance thermique (Figure 271).

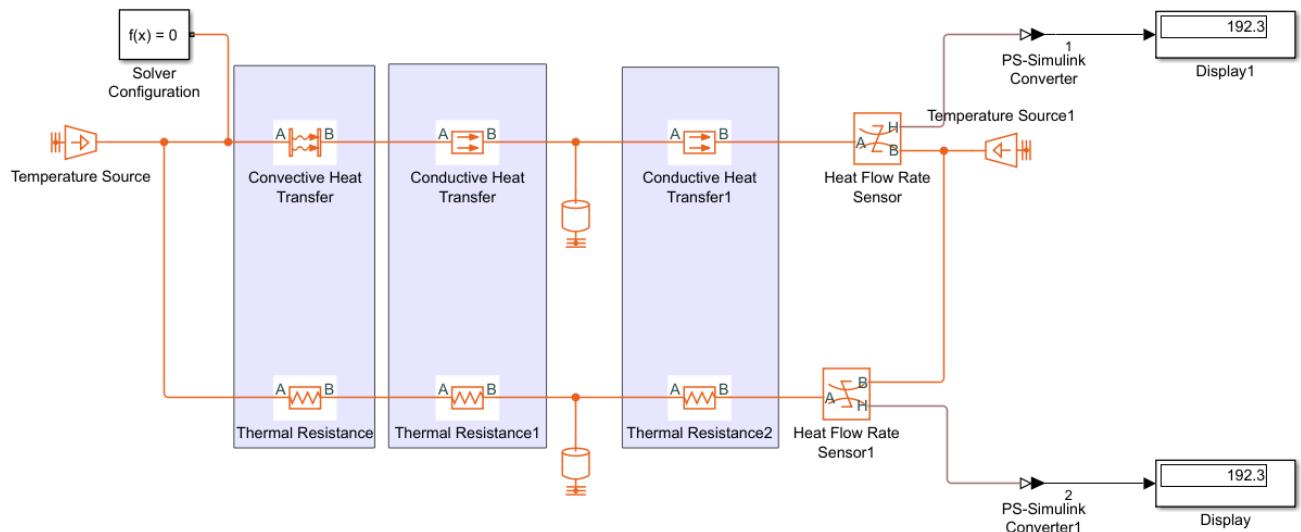


Figure 271 : modélisation d'une couche de paroi en utilisant

Ouvrir les différents blocs et observer le paramétrage effectué

Lancer la simulation et vérifier l'équivalence des deux modélisations en constatant l'égalité des flux thermiques qui traversent les parois.

20. Modélisation avec Simscape d'une paroi multicouche d'un bâtiment

La paroi que nous allons modéliser est constituée de trois couches (Figure 272). Les caractéristiques des différentes couches sont données Figure 273. La paroi est de hauteur $h = 10 \text{ m}$ et de largeur $L = 8 \text{ m}$, sa surface est $S = 80 \text{ m}^2$

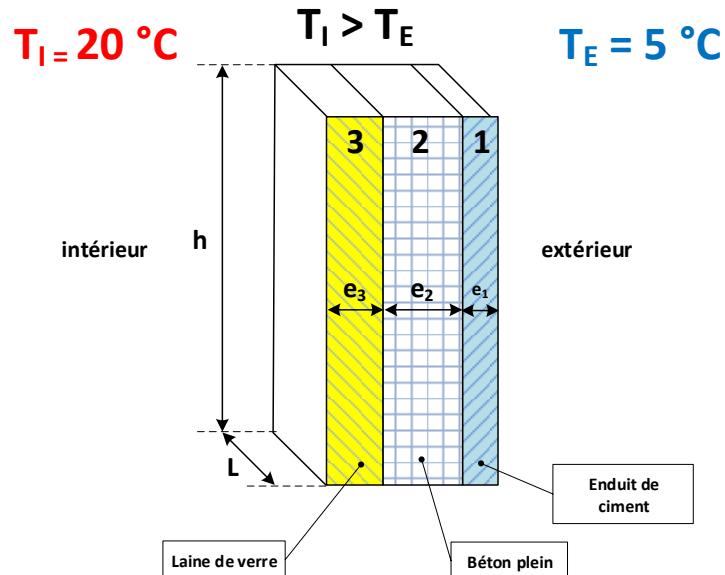


Figure 272 : modélisation d'une paroi d'un bâtiment

	Matériau	Epaisseur en cm	Masse volumique en kg/m ³	Conductivité thermique en W/(K.m)	Chaleur spécifique en J.kg ⁻¹ .K ⁻¹
Couche 1	Enduit ciment	$e_1 = 1$	$\rho_1 = 1800$	$\lambda_1=1.15$	$c_1=864$
Couche 2	Béton	$e_2 = 30$	$\rho_2 = 2500$	$\lambda_2=1.75$	$c_2=880$
Couche 3	Laine de verre	$e_3 = 10$	$\rho_3 = 20$	$\lambda_3=0.032$	$c_3=1030$

Figure 273 : caractéristiques physiques des différentes couches de matériau

Les coefficients de transfert thermique par convection sont donnés par le tableau de la Figure 274.

Paroi opaque Coefficient de transfert thermique par convection		
	k_{SI} en W/(m ² .K)	k_{SE} en W/(m ² .K)
Intérieur / Paroi		
Paroi verticale	7.7	25

Figure 274 : coefficients de transfert thermique par convection

L'objectif de la modélisation est de connaître le flux thermique qui s'établit en régime permanent de l'intérieur vers la paroi. Ce flux thermique correspond à ce qui devra être compensé par le chauffage de l'intérieur du bâtiment.

Compte tenu du grand nombre de paramètres qui interviennent dans cette modélisation, nous allons regrouper l'ensemble des valeurs numériques dans un script qui sera associé au modèle. Le script sera

exécuté pour créer les variables dans le Workspace. Ces variables pourront alors être utilisées par le modèle Simscape.

Ouvrir le script « **Parametres_multi_couches.m** » et l'exécuter afin de créer les variables correspondant aux paramètres de modélisation. Vérifiez la création des variables dans le Workspace de MATLAB.

```
% Paramètres dimensionnels
h = 6; %hauteur del a paroi en m
L = 10; %largeur de la paroi en m
S = h*L; %surface de la paroi en m2

% Epaisseur des différentes couches de la paroi en cm
e1 = 1; e2 = 30; e3 = 10;

% Caractéristiques des matériaux des différentes couches
% Masse volumique en kg/m3
p1 = 1800; p2 = 2500; p3 = 20;

% Masse des couches de la paroi en kg
m1 = p1*e1*1e-2*S; m2 = p2*e2*1e-2*S; m3 = p3*e3*1e-2*S;

% Conductivité thermique en W/(K.m)
lam_1 = 1.15; lam_2 = 1.75; lam_3 = 0.032;

% Chaleur spécifique en J/kg/K
c1 = 864; c2 = 880; c3 = 1030;

% Coefficients de transfert thermique par convection en W/(m2.K)
ksi = 7.7; kse = 25;

% Température en degrés Celsius
T_int = 20; % température intérieure
T_ext = 5; % Température extérieure
```

Figure 275 : paramètres nécessaires à la construction du modèle de la paroi multicouche

Ouvrir le fichier « **modelisation_paroi_multicouche_0.slx** » (Figure 276).

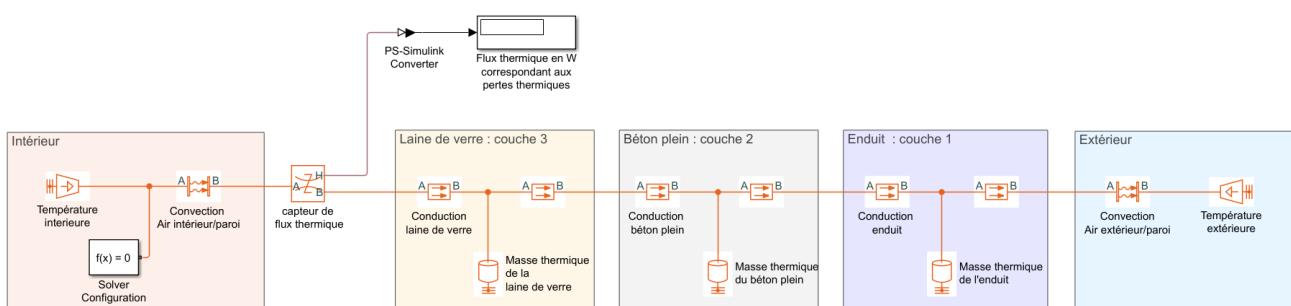


Figure 276 : modélisation d'une paroi multicouche, démarrage de la modélisation

Ce fichier représente la modélisation d'une paroi multicouche. Chaque couche a été modélisée conformément aux explications données au paragraphe 18.

Afin de pouvoir lancer la simulation il faut préalablement renseigner tous les blocs avec les paramètres de modélisation définis dans le script de la Figure 275.

Paramétrage du bloc : Température intérieure

Ce bloc représente la température de l'intérieur de la pièce.

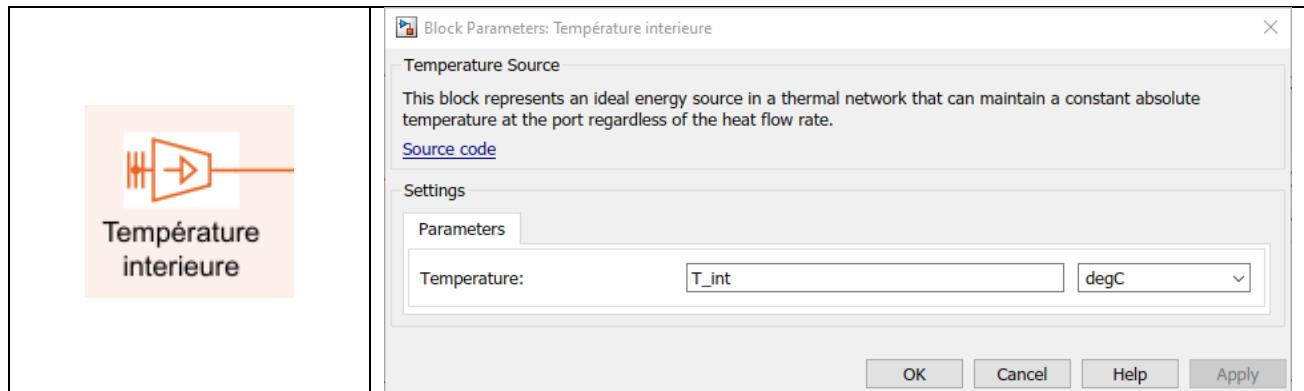


Figure 277 : paramétrage du bloc Température intérieure

Paramétrage du bloc : Convection Air intérieur/paroi

Ce bloc modélise les échanges thermiques par convection entre la surface de la paroi et l'intérieur de la pièce.

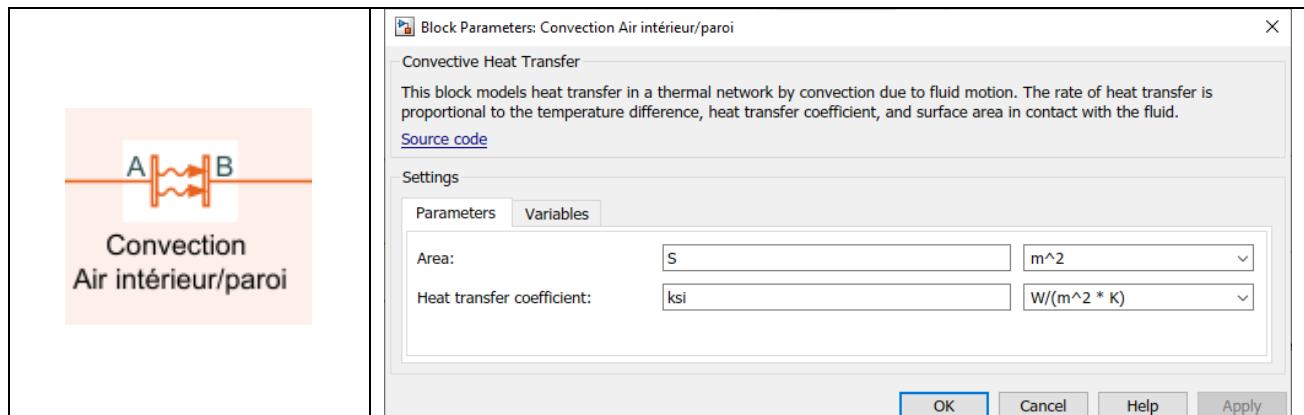


Figure 278 : paramétrage du bloc Convection Air intérieur/paroi

Paramétrage du bloc : capteur de flux thermique

Ce bloc ne nécessite pas de paramétrage. Ce capteur permet de mesurer le flux thermique qui s'établit entre l'intérieur du bâtiment et la paroi.

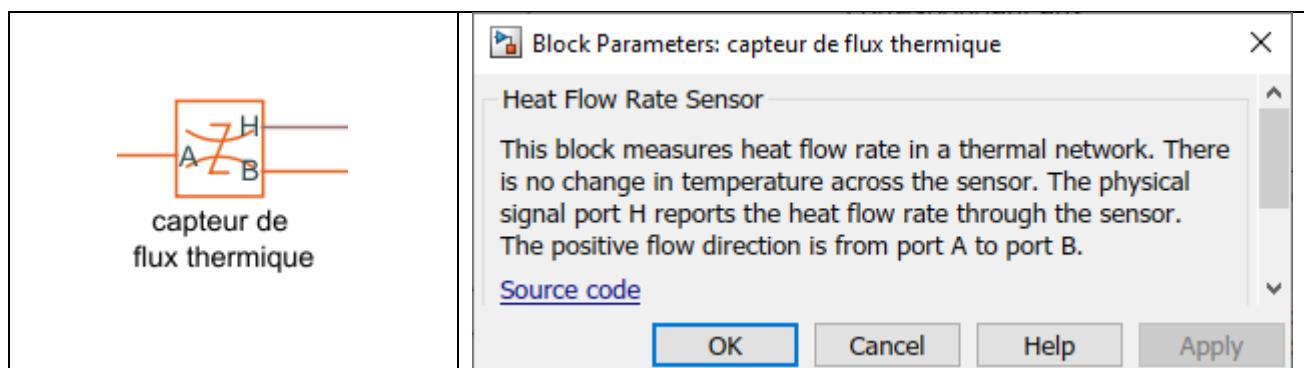


Figure 279 : paramétrage du bloc Capteur de flux thermique

Paramétrage du bloc : PS-Simulink Converter

Il faut préciser dans ce bloc l'unité d'affichage du flux thermique (Watt).

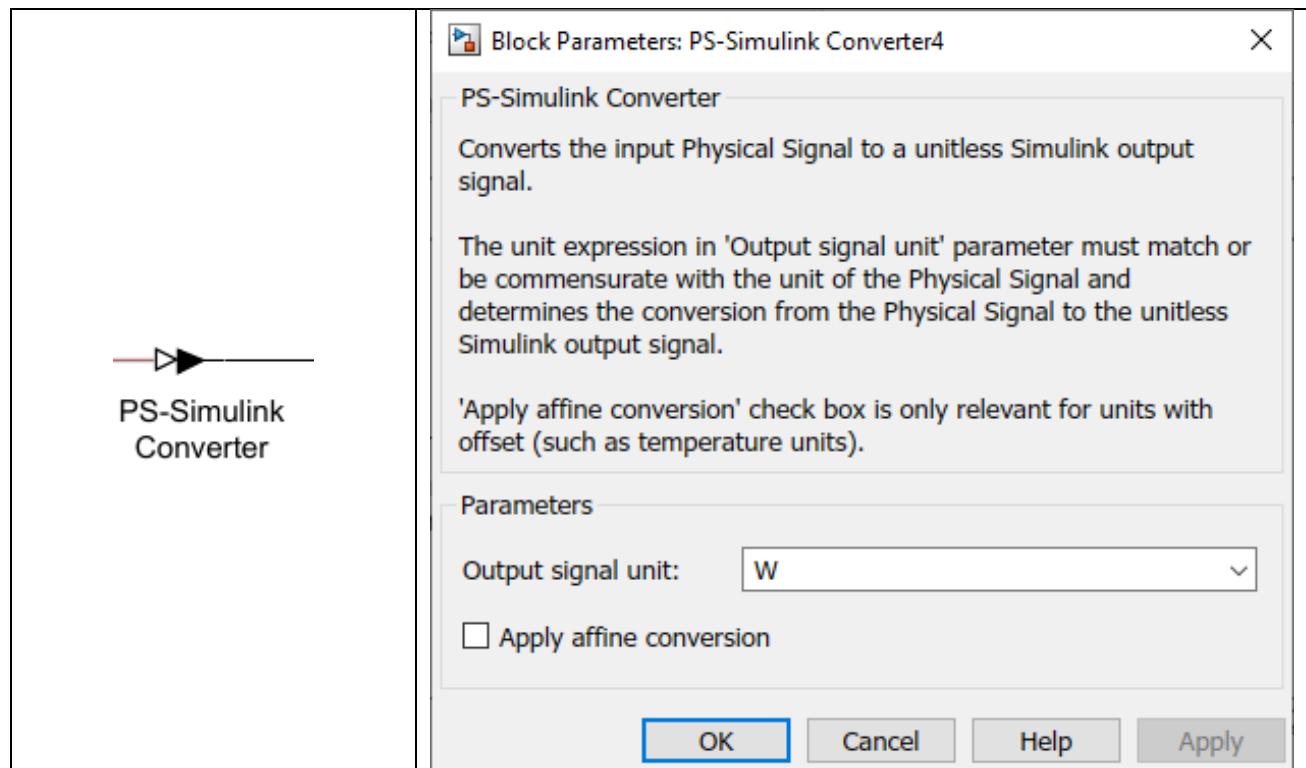


Figure 280 : paramétrage du bloc PS – Simulink Converter

Paramétrage du bloc : Conduction laine de verre

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur de laine de verre.

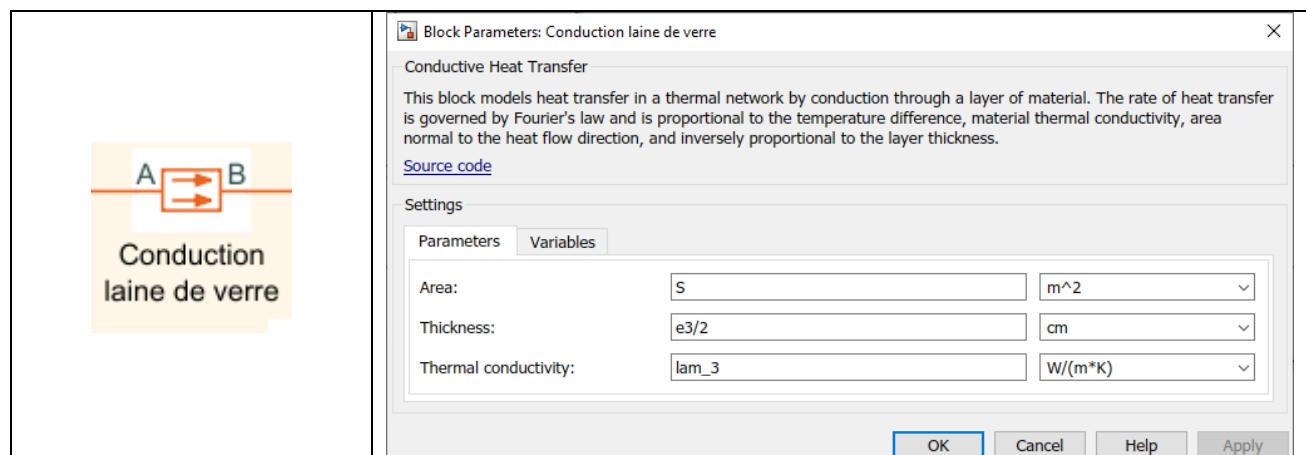


Figure 281 : paramétrage du bloc Conduction laine de verre

Paramétrage du bloc : *Masse thermique de la laine de verre*

Ce bloc modélise la masse thermique de la laine de verre. Il y a deux onglets à paramétriser. L'onglet **Parameters** qui permet de renseigner la masse de la laine de verre et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que la laine de verre est à la même température que l'intérieur de la pièce en début de simulation.

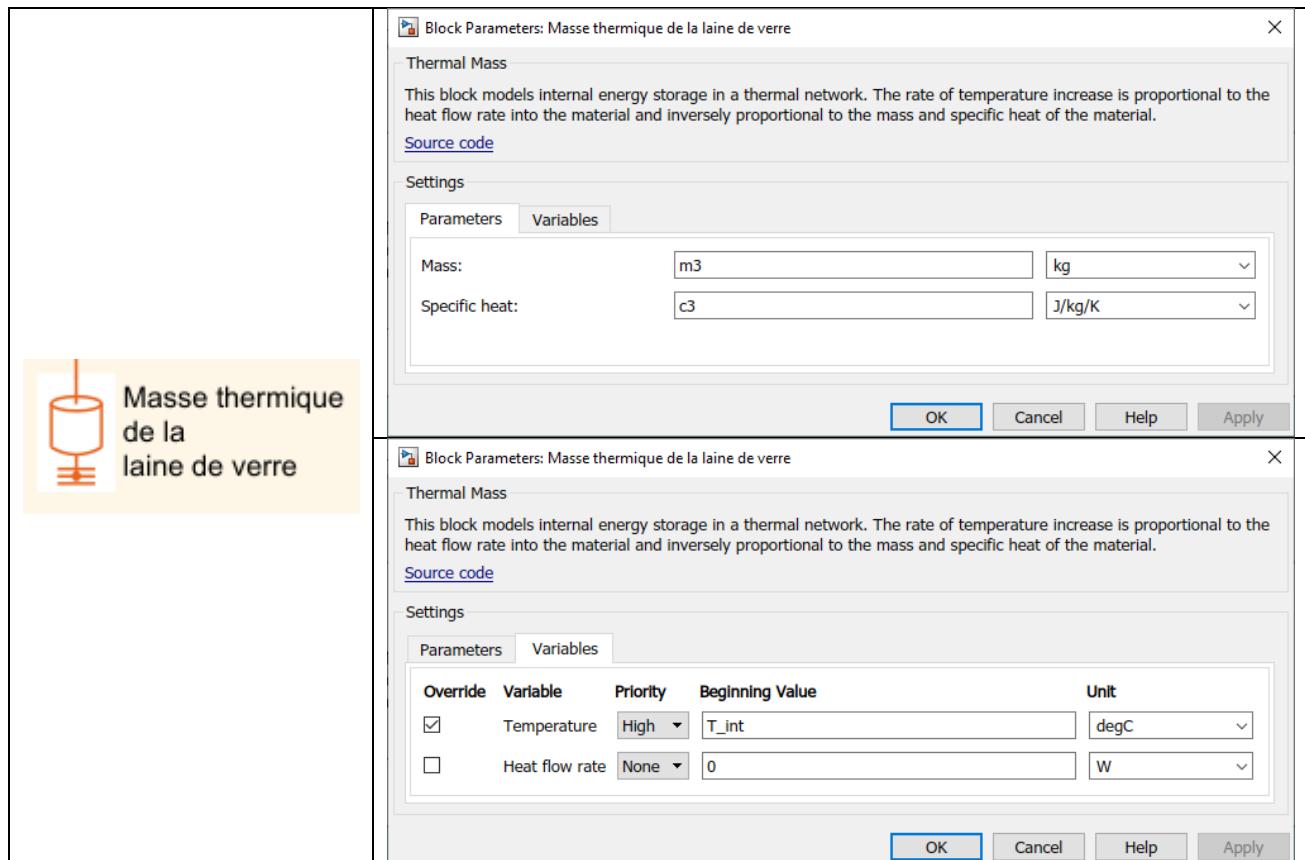


Figure 282 : paramétrage du bloc **Masse thermique de la laine de verre**

Paramétrage du bloc : *Conduction Béton plein*

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur de béton plein.

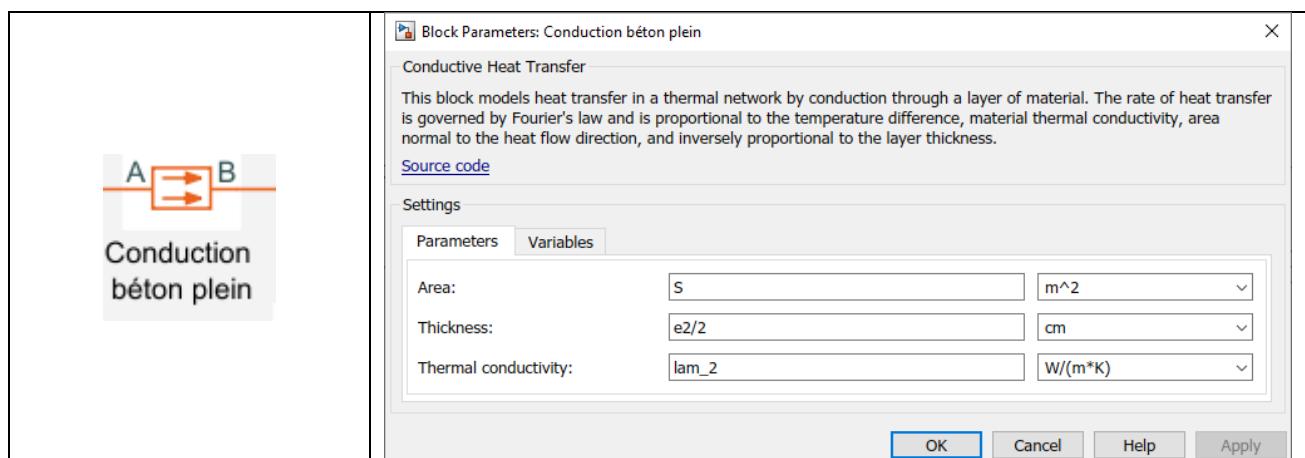


Figure 283 : paramétrage du bloc **Conduction béton plein**

Paramétrage du bloc : *Masse thermique de béton plein*

Ce bloc modélise la masse thermique de la couche de béton plein. Il y a deux onglets à paramétrier. L'onglet **Parameters** qui permet de renseigner la masse du béton plein et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que le béton plein est à une température $(T_{int}+T_{ext})/2$ en début de simulation.

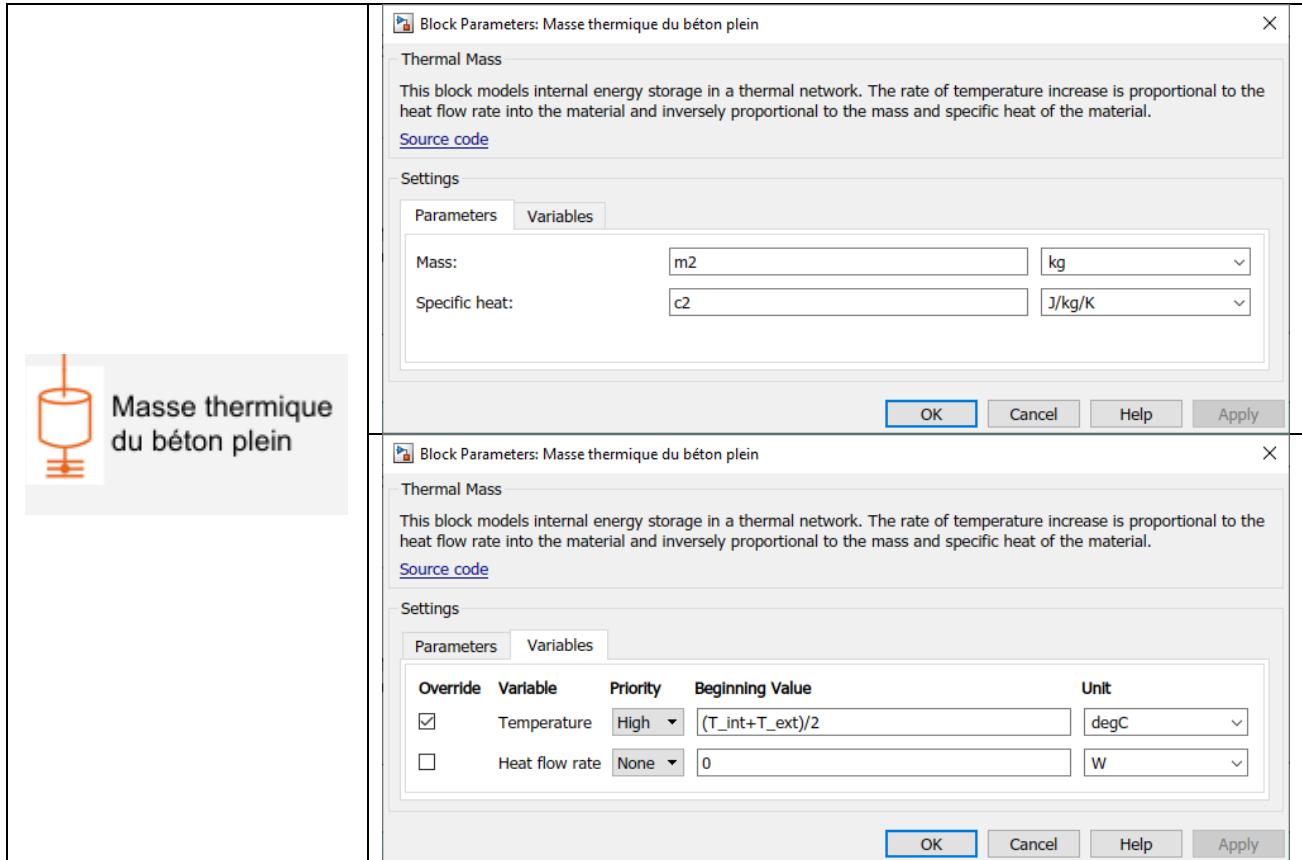


Figure 284 : paramétrage du bloc **Masse thermique du béton plein**

Paramétrage du bloc : Conduction enduit

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur d'enduit.

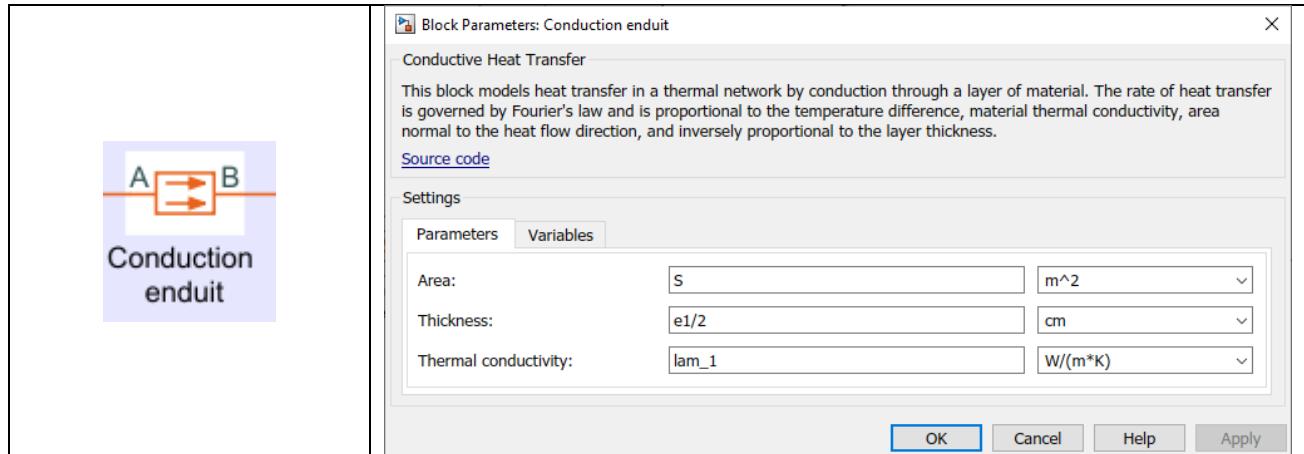


Figure 285 : paramétrage du bloc Conduction enduit

Paramétrage du bloc : Masse thermique de l'enduit

Ce bloc modélise la masse thermique de la couche d'enduit. Il y a deux onglets à paramétrer. L'onglet **Parameters** qui permet de renseigner la masse de l'enduit et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que l'enduit est à la température extérieure en début de simulation.

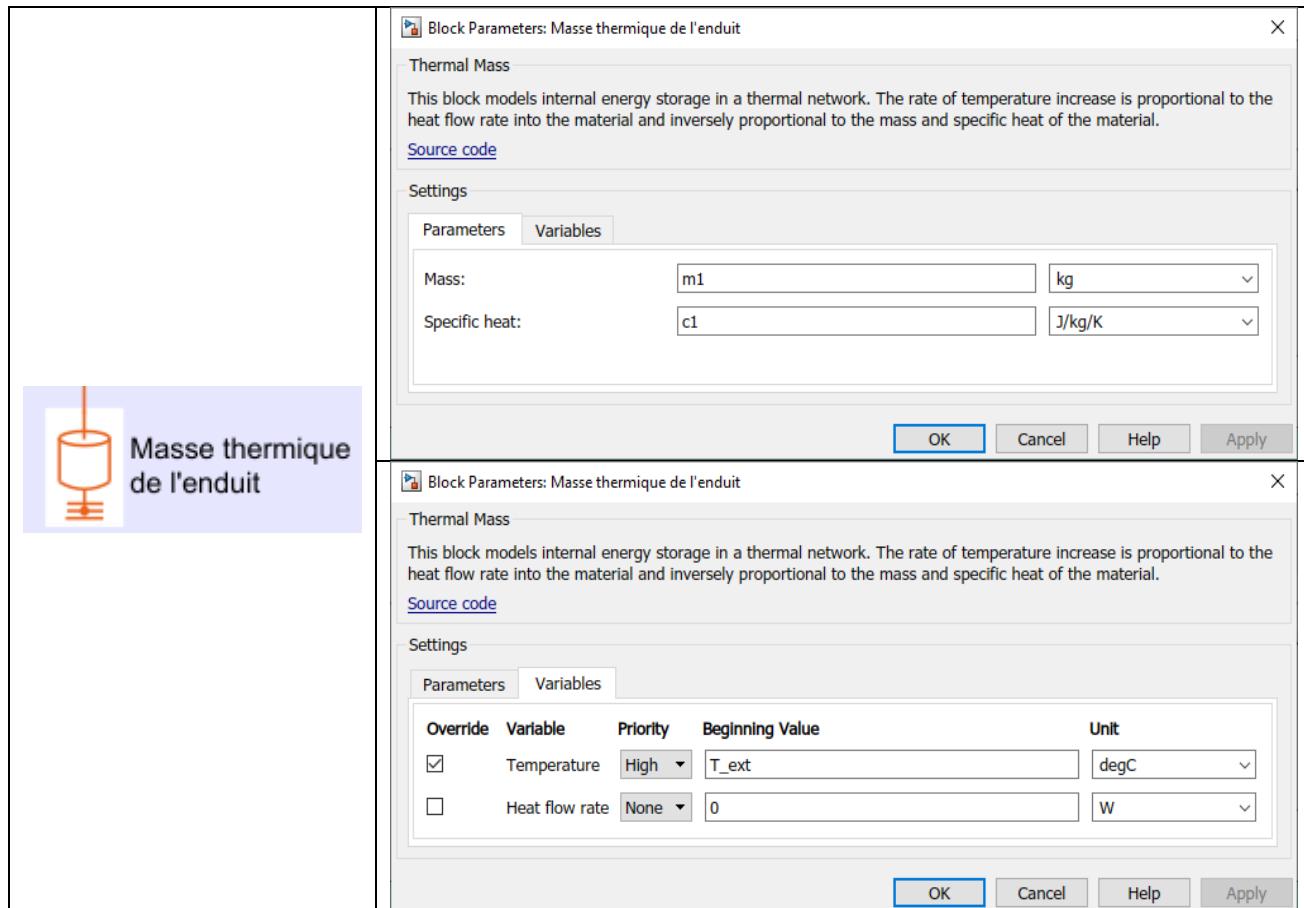


Figure 286 : paramétrage du bloc Masse thermique de l'enduit

Paramétrage du bloc : Convection Air extérieur/paroi

Ce bloc modélise les échanges thermiques par convection entre la surface de la paroi et l'extérieur.

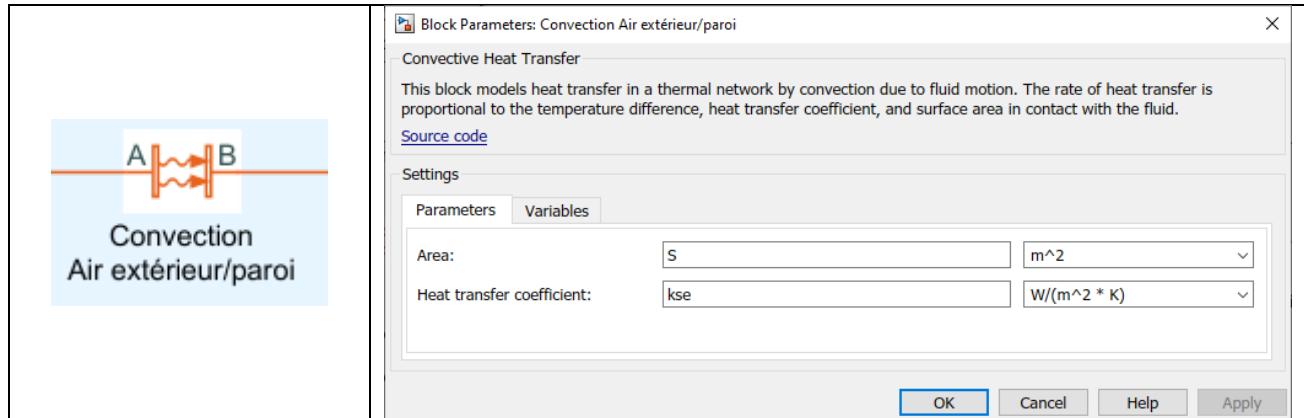


Figure 287 : paramétrage du bloc Convection Air extérieur/paroi

Paramétrage du bloc : Température extérieure

Ce bloc représente la température de l'extérieur.

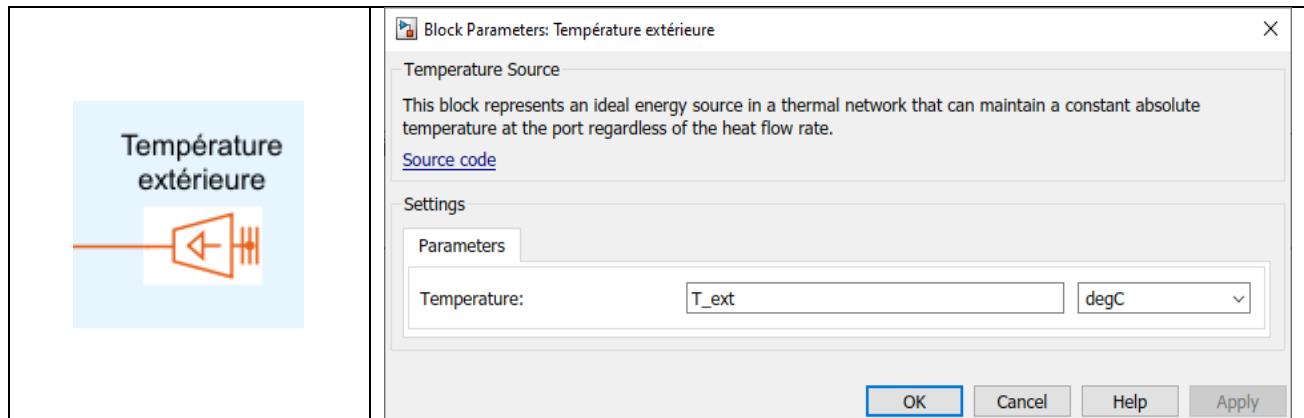


Figure 288 : paramétrage du bloc Température extérieure

Le fichier « **modelisation_paroi_multi_couche_1.slx** » contient le modèle entièrement paramétré.

Pour être certain d'obtenir la valeur en régime permanent du flux thermique, il faut choisir un temps de simulation assez long. Nous allons donc régler le temps de simulation sur 2 jours soit $48*3600$ s.



Figure 289 : réglage du temps de simulation

Lancer la simulation en cliquant sur .

Il est possible de visualiser le flux thermique en régime permanent, c'est-à-dire la puissance de chauffage qu'il faudra fournir pour maintenir la température de la pièce intérieure.

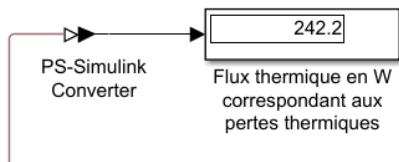


Figure 290 : visualisation du flux thermique en intérieur de paroi en régime permanent

Nous pouvons constater que le flux thermique s'établit à 242.2 W.

Afin d'évaluer l'influence des différentes couches, il est maintenant possible de modifier les paramètres du modèle regroupés dans le script.

Si nous voulons par exemple évaluer l'influence de l'épaisseur de la laine de verre (paramètre e3=10 cm), nous pouvons modifier le script en imposant e3 = 20 cm ce qui revient à doubler l'épaisseur de la laine de verre.

Modifier la valeur de e3 dans le script (Figure 291).

```
% Epaisseur des différentes couches de la paroi en cm
e1 = 1; e2 = 30; e3 = 20;
```

Figure 291 : modification d'un paramètre dans le script

Exécuter à nouveau le script pour générer la nouvelle variable e3 dans le Workspace. **Relancer** la simulation du modèle Simscape et relever la nouvelle valeur du flux thermique.

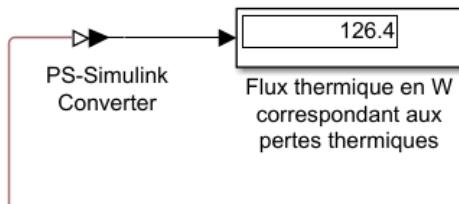


Figure 292 : visualisation de la nouvelle valeur du flux thermique

Nous pouvons constater la diminution du flux thermique dû à l'augmentation de l'épaisseur du matériau isolant. Il est possible en utilisant cette méthode d'évaluer l'influence de tous les paramètres du modèle.

Il est également pu choisir de modéliser tous les échanges thermiques en utilisant le concept de résistance thermique en modifiant le modèle.

Ouvrir le modèle « **modelisation_paroi_multicouche_2.slx** » (Figure 293).

Ce fichier contient la modélisation de la paroi multicouche que nous venons de construire et son modèle équivalent qui utilise le concept de résistance thermique.



Lancer la simulation en cliquant sur .

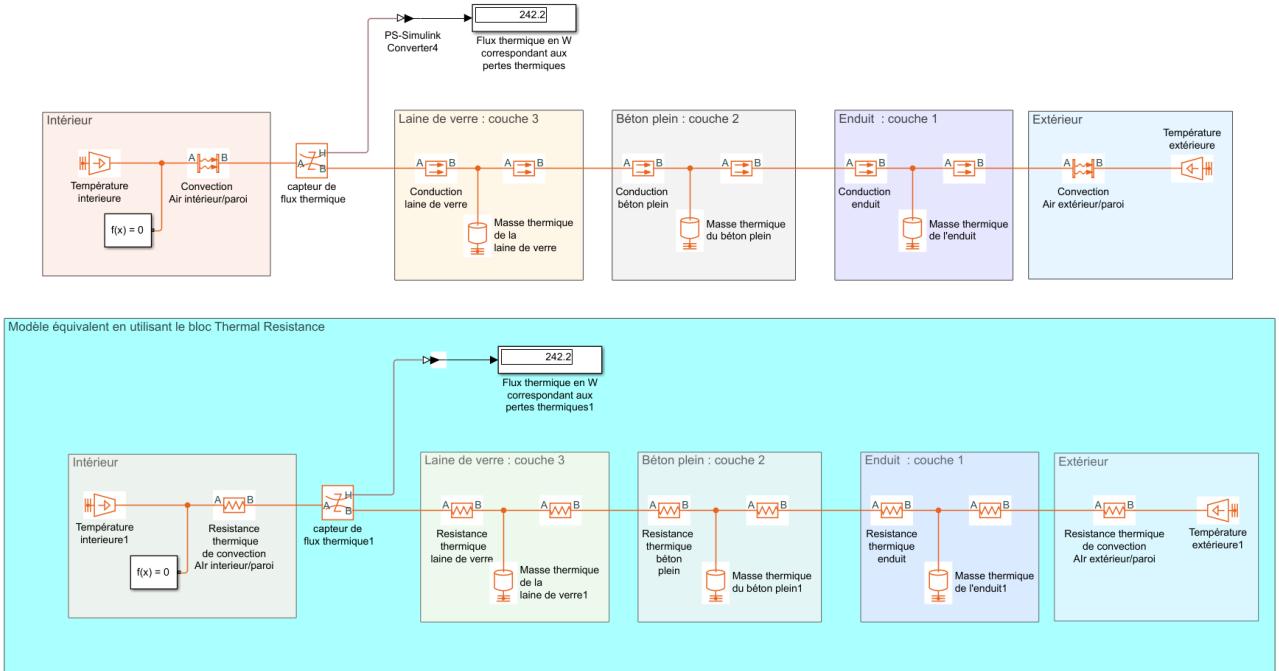


Figure 293 : modélisation équivalente avec des résistances thermiques

Le paramétrage des blocs « résistances thermiques » a été effectué conformément aux explications données paragraphe 19.

Paramétrage du bloc : Résistance thermique de convection Air intérieur/paroi

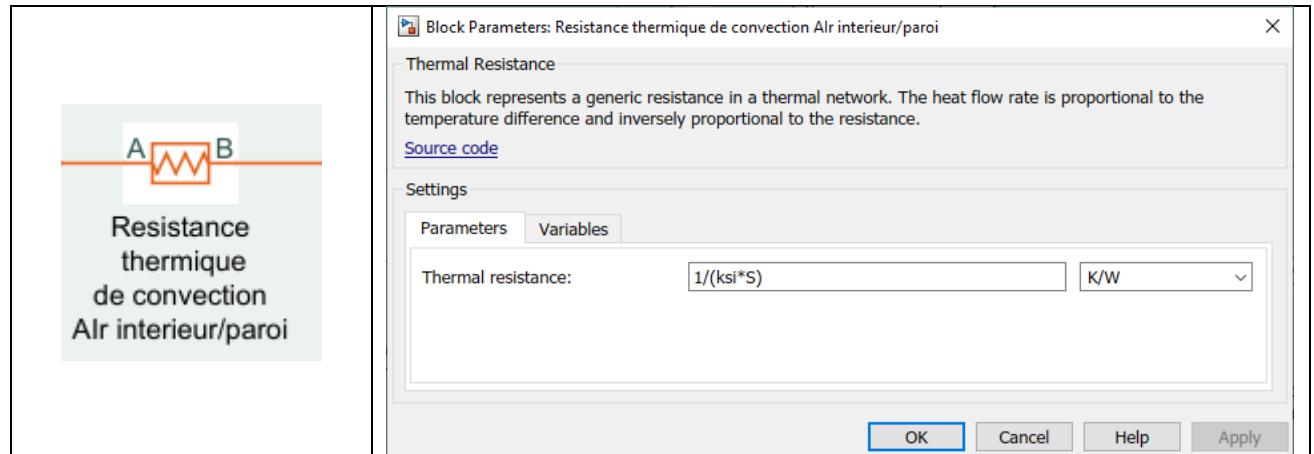


Figure 294 : paramétrage du bloc Résistance thermique de convection Air intérieur/paroi

Paramétrage du bloc : Résistance thermique de conduction de la laine de verre

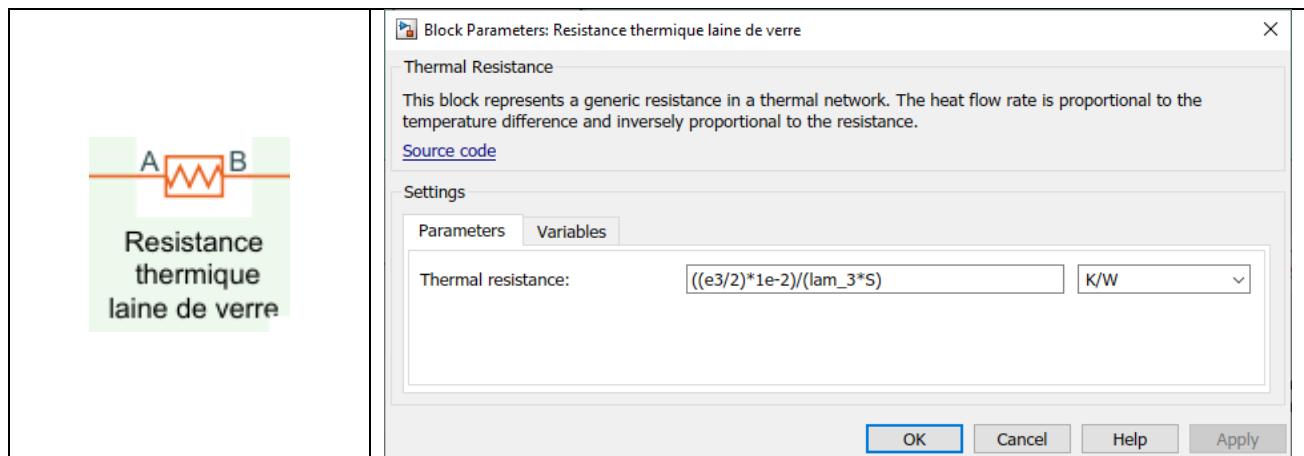


Figure 295 : paramétrage du bloc Résistance thermique de conduction de la laine de verre

Paramétrage du bloc : Résistance thermique de conduction du béton plein

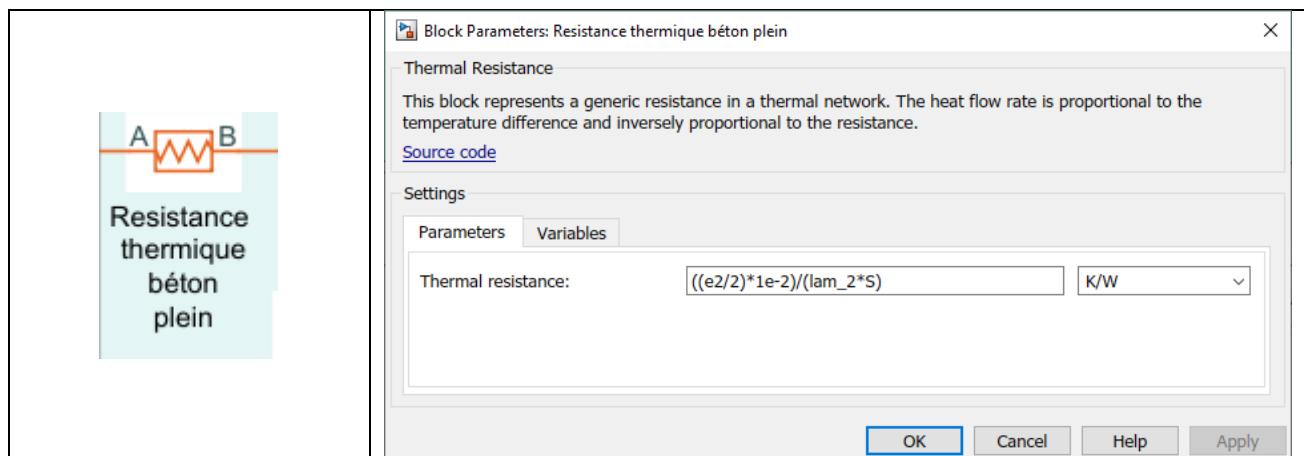


Figure 296 : paramétrage du bloc Résistance thermique de conduction du béton plein

Paramétrage du bloc : Résistance thermique de conduction de l'enduit

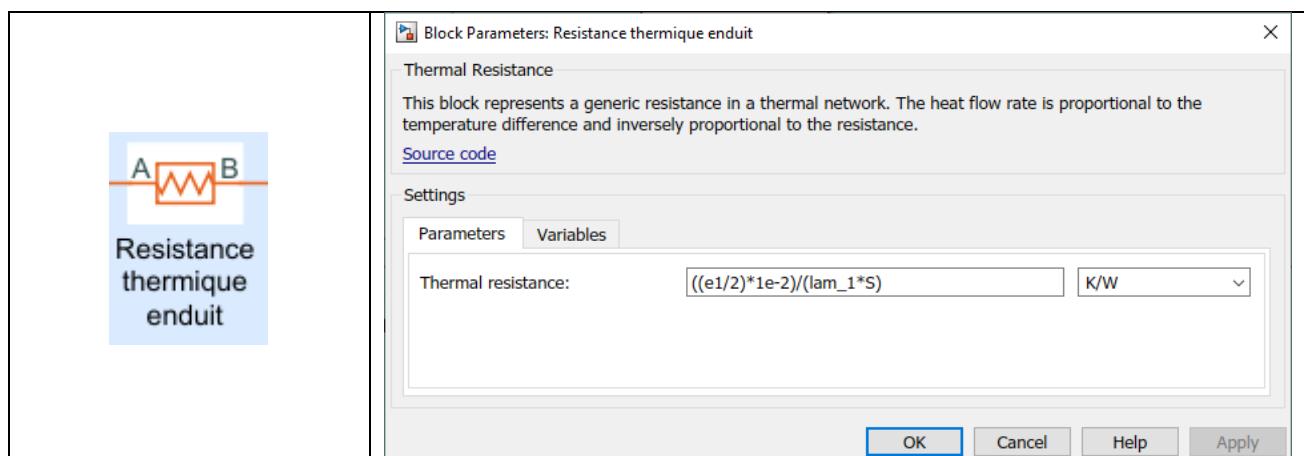


Figure 297 : paramétrage du bloc Résistance thermique de conduction de l'enduit

Paramétrage du bloc : Résistance thermique de convection Air extérieure/paroi

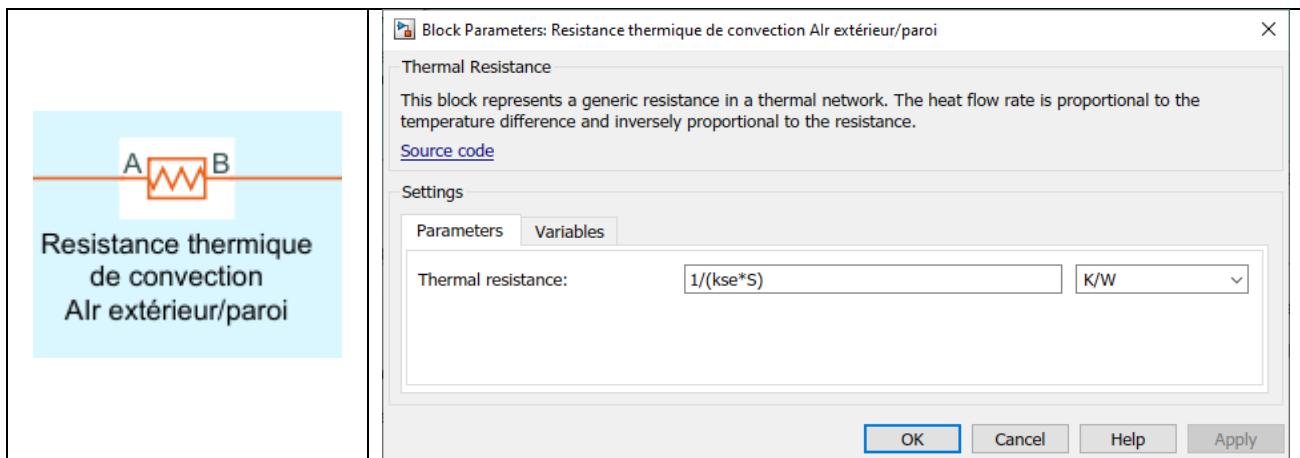


Figure 298 : paramétrage du bloc Résistance thermique de convection Air extérieur/paroi

21. Modélisation de l'efficacité d'un double vitrage avec couche d'argon

Nous souhaitons comparer l'efficacité thermique d'une paroi en double vitrage simple (Figure 299) et celle d'une paroi en double vitrage avec une couche d'argon entre les deux vitres (Figure 300).

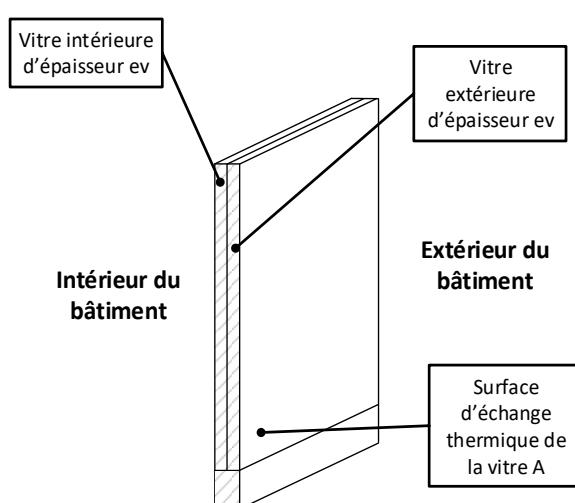


Figure 299 : double vitrage simple

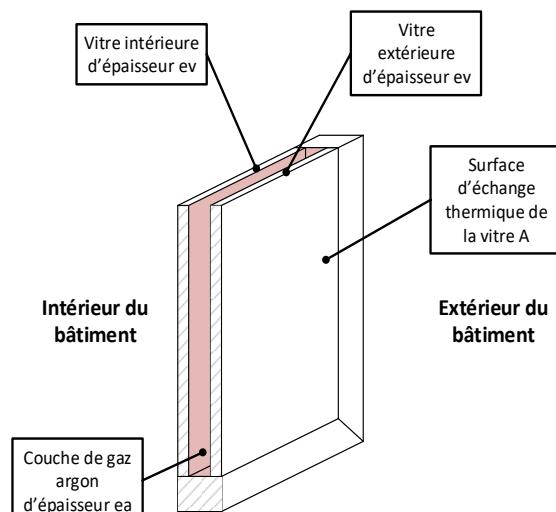


Figure 300 : double vitrage avec couche d'argon

La pièce intérieure à une surface $S=60\text{m}^2$ et une hauteur de plafond $h=2.5 \text{ m}$ (Figure 301). Elle sera modélisée par une masse thermique constitué de l'air contenu dans la pièce. La vitre aura une surface $A=2 \text{ m}^2$. Les seuls échanges thermiques pris en compte dans le modèle seront ceux qui auront lieu entre la paroi vitrée et l'intérieur de la pièce. La température de l'intérieur de la pièce est $T_{int}=20^\circ\text{C}$ et la température extérieure est $T_{ext}=10^\circ\text{C}$.

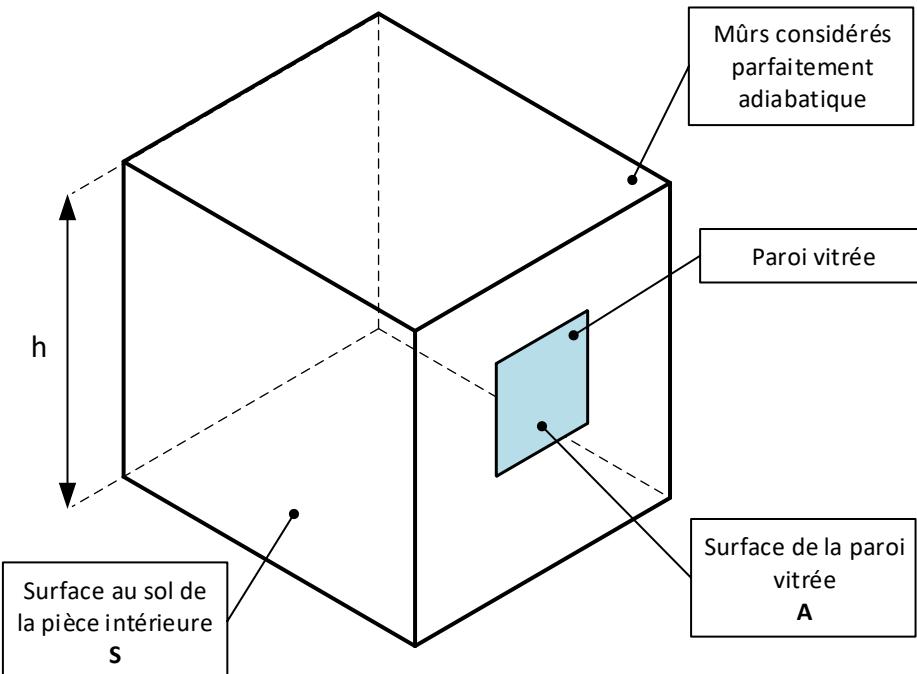


Figure 301 : modèle retenu pour la pièce intérieure et pour la paroi vitrée

Les caractéristiques des matériaux qui interviennent dans la modélisation sont données Figure 302

Matériau	Epaisseur en mm	Masse volumique en kg/m ³	Conductivité thermique en W/(K.m)	Chaleur spécifique en J.kg ⁻¹ .K ⁻¹
Verre	e _v = 8	ρ _v = 2530	λ _v =1	c _v =720
Argon	e _a = 10	ρ _a = 1.784	λ _a =0.018	c _a =320
Air		ρ _{air} = 1.255	λ _{air} =0.025	c _{air} =1030

Figure 302 : caractéristiques des matériaux

L'objectif de la modélisation est de connaître l'évolution de la température à l'intérieur de la pièce pendant 24 heures.

Compte tenu du grand nombre de paramètres qui interviennent dans cette modélisation, nous allons regrouper l'ensemble des valeurs numériques dans un script qui sera associé au modèle. Le script sera exécuté pour créer les variables dans le Workspace. Ces variables pourront alors être utilisées par le modèle Simscape.

Ouvrir le script « **Parametres_double_vitrage.m** » et l'exécuter afin de créer les variables correspondant aux paramètres de modélisation. Vérifiez la création des variables dans le Workspace de MATLAB.

```

% Paramètres dimensionnels
A = 2 ; %surface de la vitre en m2

% Dimensions de la pièce intérieure en m
S = 60; h = 2.5;

% Volume d'air contenu dans la pièce
Vair = S * h;

% Epaisseur des différentes couches
ev = 8; % épaisseur de la couche de verre en mm
ea = 10; % épaisseur de la couche d'argon en mm

% Caractéristiques des matériaux des différentes couches
% Masse volumique en kg/m3
pv = 2530; % verre
pa = 1.784; % argon
pair = 1.225; % air

% Chaleur spécifique en J/kg/K
cv = 720; ca = 320; cair = 1030;

% Masse des couches de la paroi en kg
Mv = pv * A * ev * 1e-3;
Ma = pa * A * ea * 1e-3;

% Masse de l'air contenu dans la pièce
Mair = pair * Vair;

% Conductivité thermique en W/(K.m)
lam_v = 1; lam_a = 0.018; lam_air = 0.025;

% Coefficients de transfert thermique par convection en W/(m2.K)
ksi = 7.7; % entre intérieur et paroi vitrée
kse = 25; % entre extérieur et paroi vitrée

% Température en degrés Celsius
T_int = 20; % température intérieure
T_ext = 10; % Température extérieure

```

Figure 303 : paramètres nécessaires à la construction du modèle d'un double vitrage

Ouvrir le fichier « **double_vitrage_0.slx** » (Figure 304).

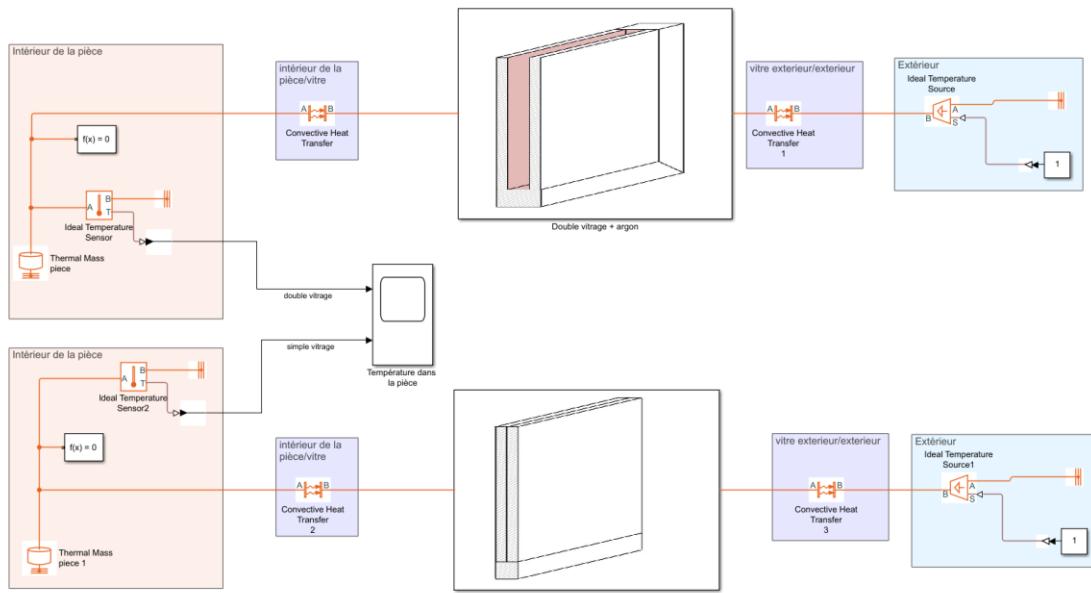


Figure 304 : modélisation d'une paroi multicouche, démarrage de la modélisation

Ce fichier contient la structure de la modélisation de l'efficacité thermique d'un double vitrage avec couche d'argon intermédiaire et d'un double vitrage simple. Le modèle contient 2 sous-systèmes « Double vitrage + argon » (Figure 305) et « Double vitrage » (Figure 306) que l'on peut développer en double cliquant sur chaque bloc.

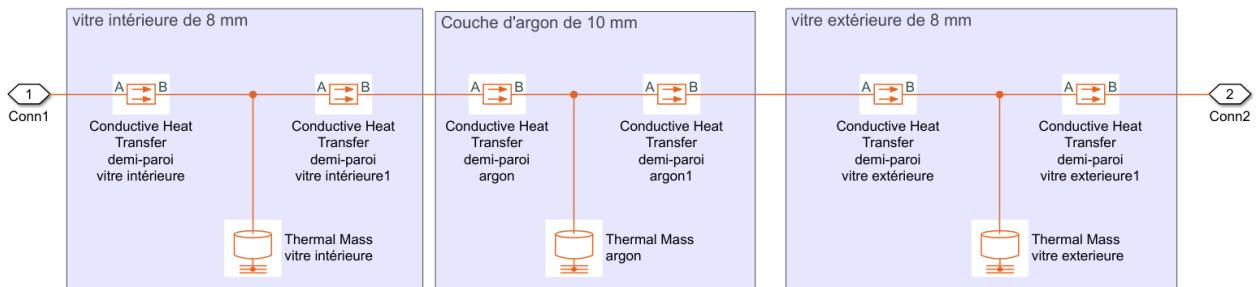


Figure 305 : sous-système « Double vitrage + argon »

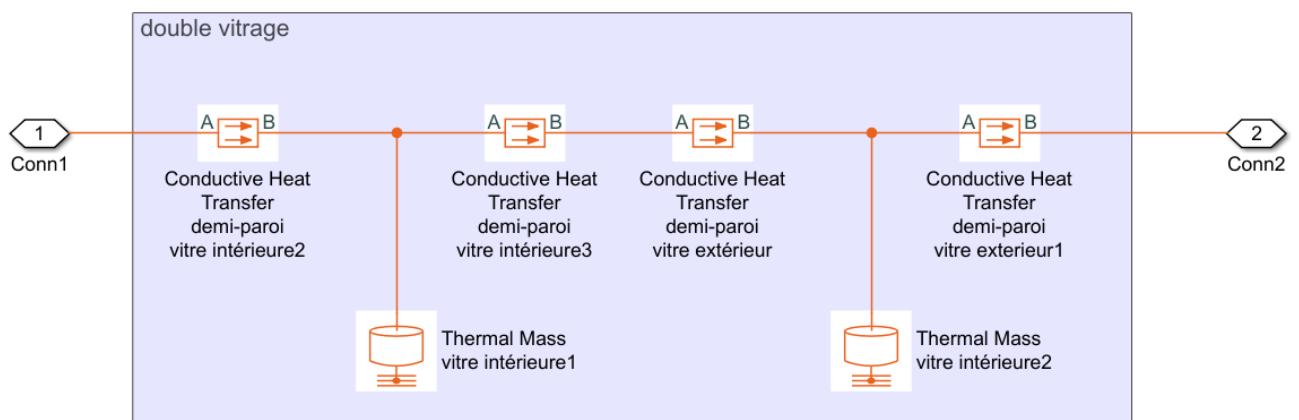
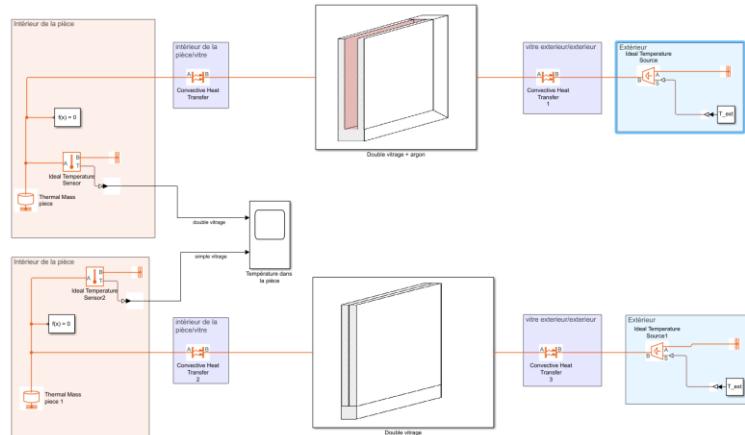


Figure 306 : sous-système « Double vitrage »

Afin de pouvoir lancer la simulation, il faut préalablement renseigner tous les blocs avec les paramètres de modélisation définis dans le script de la Figure 303.

Paramétrage du modèle global



Paramétrage du bloc : Masse thermique de la pièce intérieure

La pièce est modélisée par la masse thermique de l'air qu'elle contient. Les deux blocs « Thermal Mass pièce » et « Thermal Mass pièce 1 » sont identiques et recevront le même paramétrage.

L'onglet **Parameters** qui permet de renseigner la masse de l'air et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que l'intérieur est à la température T_{int} en début de simulation.

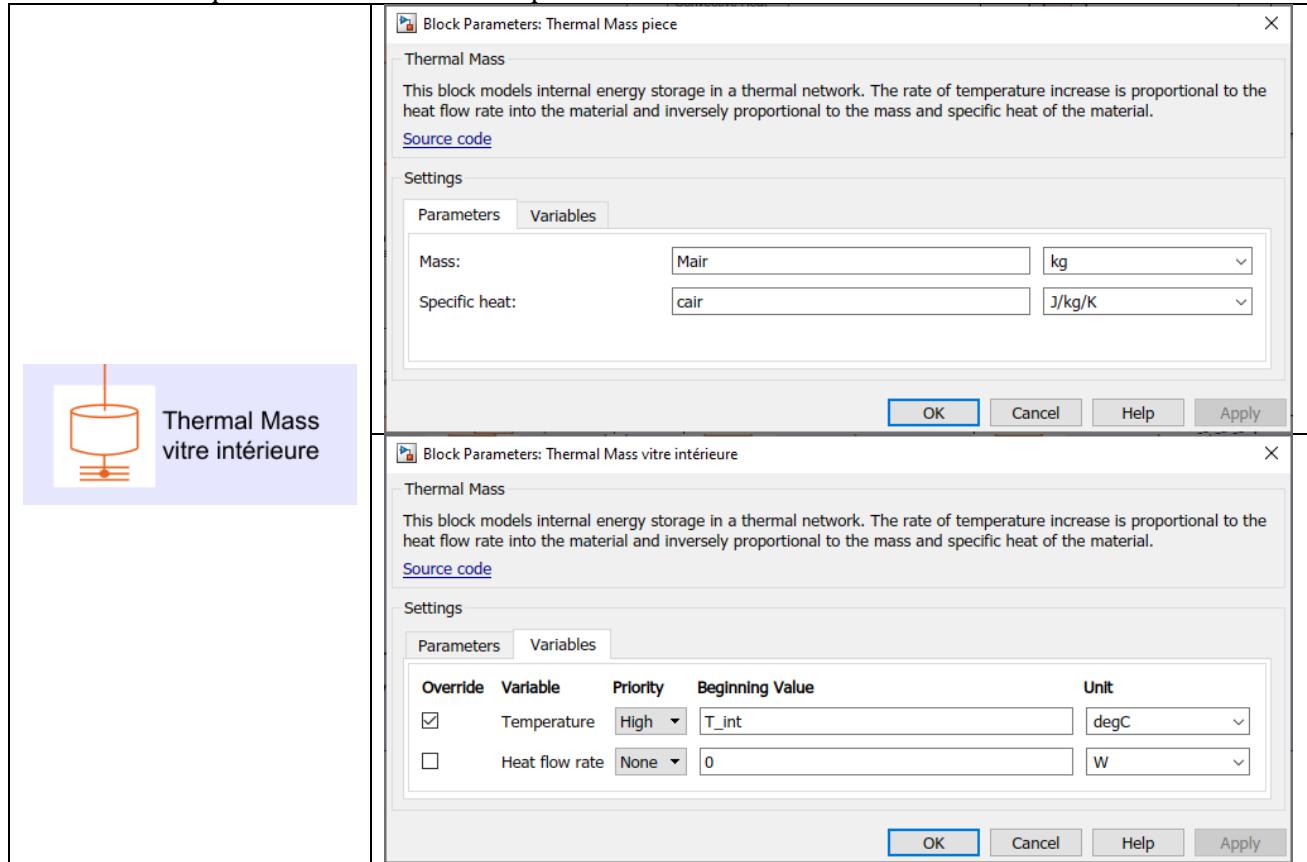


Figure 307 : paramétrage du bloc Masse thermique de l'air de la pièce intérieure

Paramétrage du bloc : Convection Air intérieur/paroi vitrée

Ce bloc modélise les échanges thermiques par convection entre la surface de la paroi vitrée et l'intérieur de la pièce. Les deux blocs « Convective Heat Transfer » et « Convective Heat Transfer2 » sont identiques et recevront le même paramétrage.

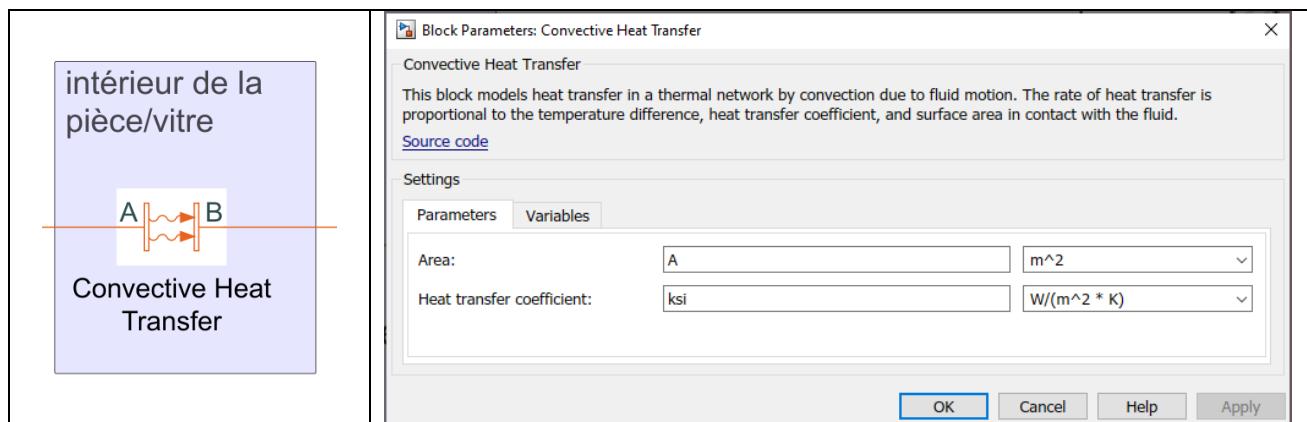


Figure 308 : paramétrage du bloc Convection Air intérieur/paroi

Paramétrage du bloc : Convection paroi vitrée/air extérieur

Ce bloc modélise les échanges thermiques par convection entre la surface de la paroi vitrée et l'extérieur. Les deux blocs « Convective Heat Transfer1 » et « Convective Heat Transfer3 » sont identiques et recevront le même paramétrage.

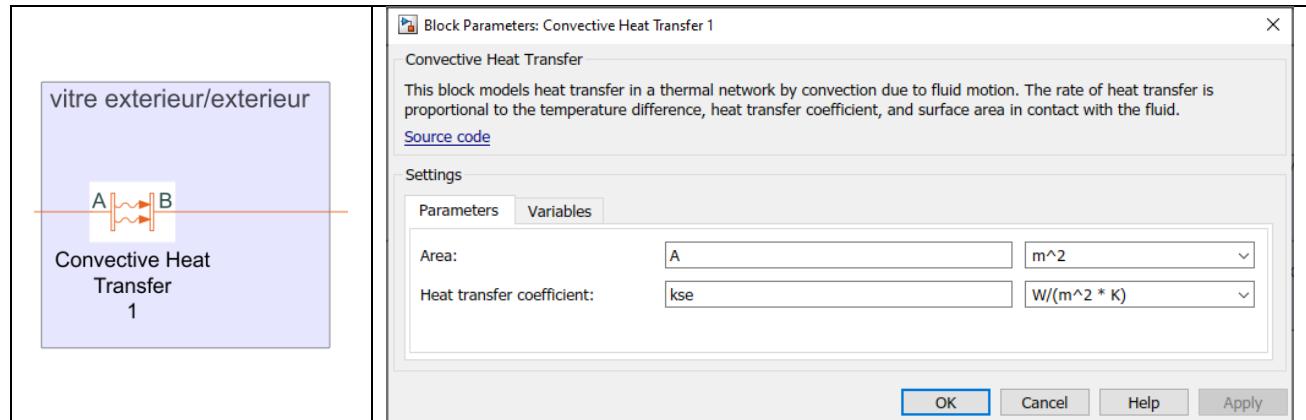


Figure 309 : paramétrage du bloc Convection paroi vitrée/air extérieur

Paramétrage du bloc : Température extérieure

Ce bloc représente la température de l'extérieur. Il est modélisé par un bloc Constant.

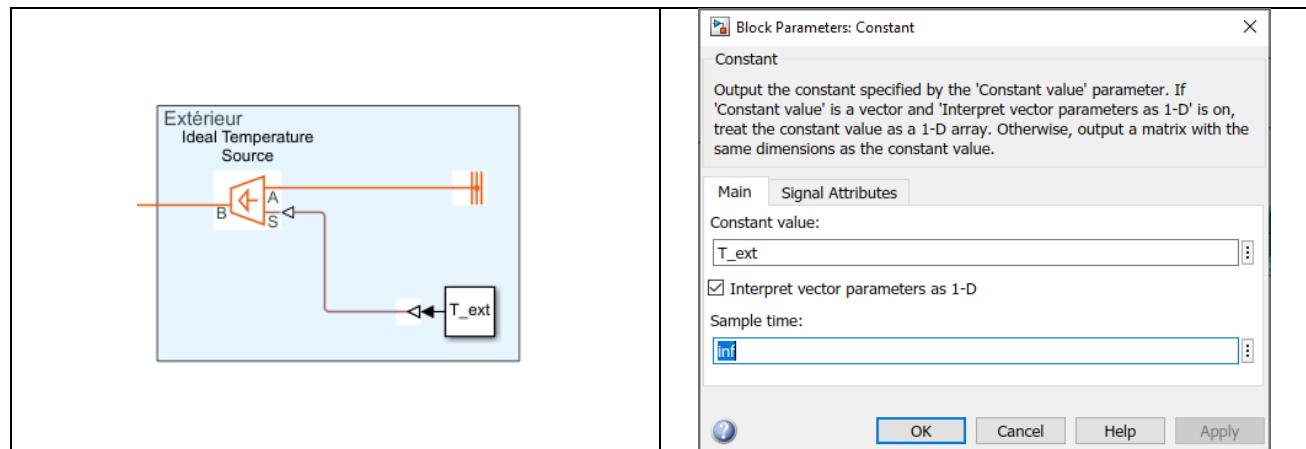


Figure 310 : paramétrage du bloc Température extérieure

Paramétrage du bloc : PS-Simulink Converter

Il faut préciser dans ce bloc l'unité d'affichage de la température est le degré Celsius. Choisir l'unité **degC** et cocher **Apply affine conversion**.

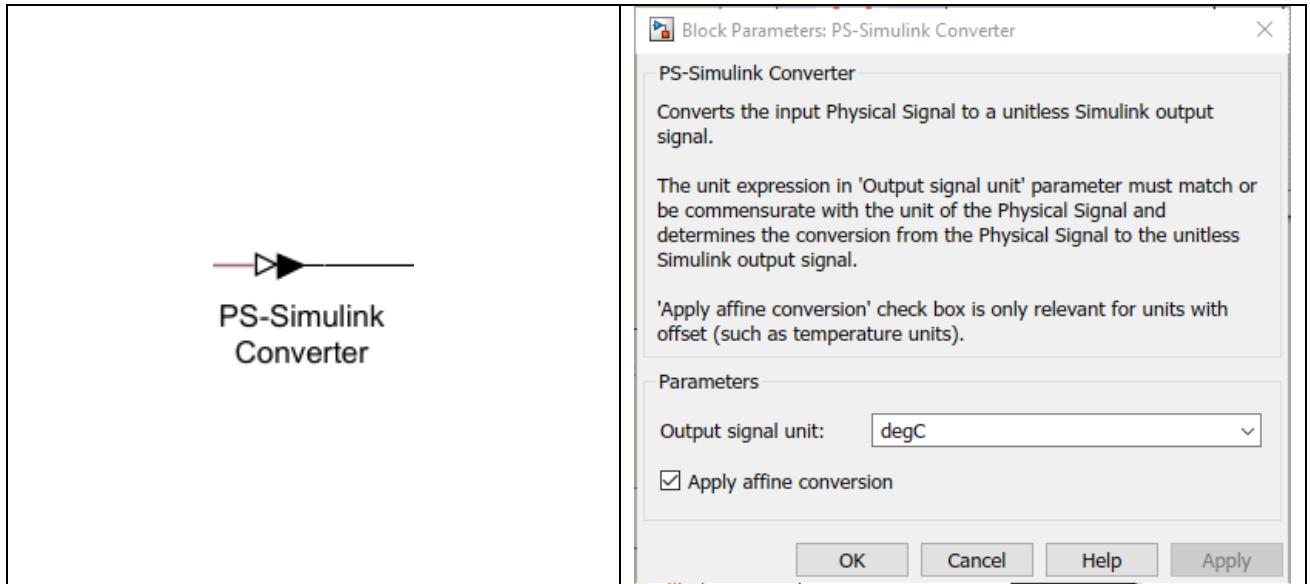
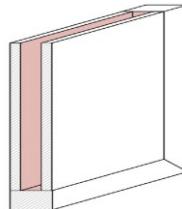


Figure 311 : paramétrage du bloc PS – Simulink Converter

Paramétrage du sous-système « Double vitrage + argon »



Paramétrage du bloc : Conduction du verre de la vitre intérieure

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur de verre

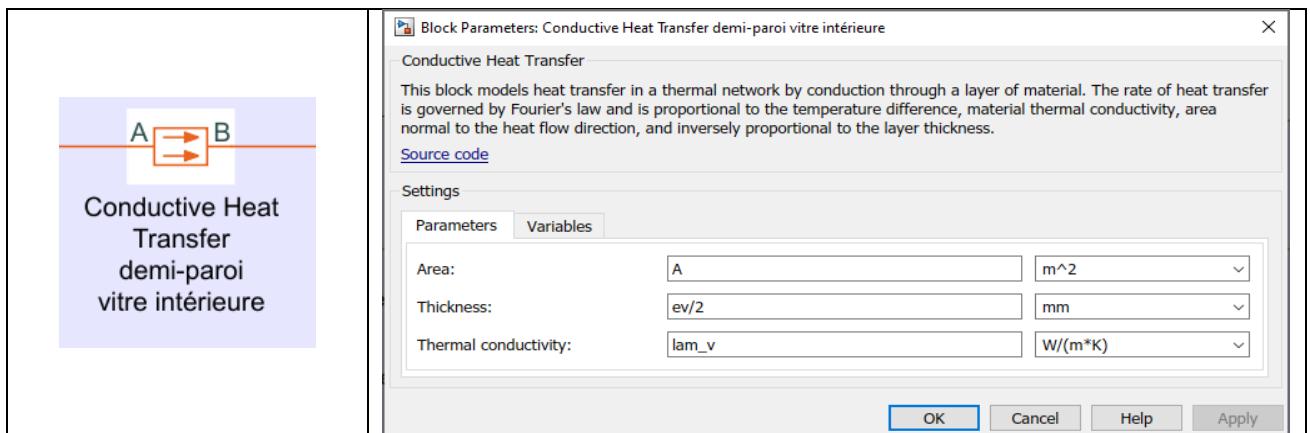


Figure 312 : paramétrage du bloc Conduction de la vitre intérieure

Paramétrage du bloc : *Masse thermique de la vitre intérieure*

Ce bloc modélise la masse thermique du verre de la vitre intérieure. Il y a deux onglets à paramétriser. L'onglet **Parameters** qui permet de renseigner la masse du verre et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que le verre de la vitre intérieure est à la même température que l'intérieur de la pièce en début de simulation.

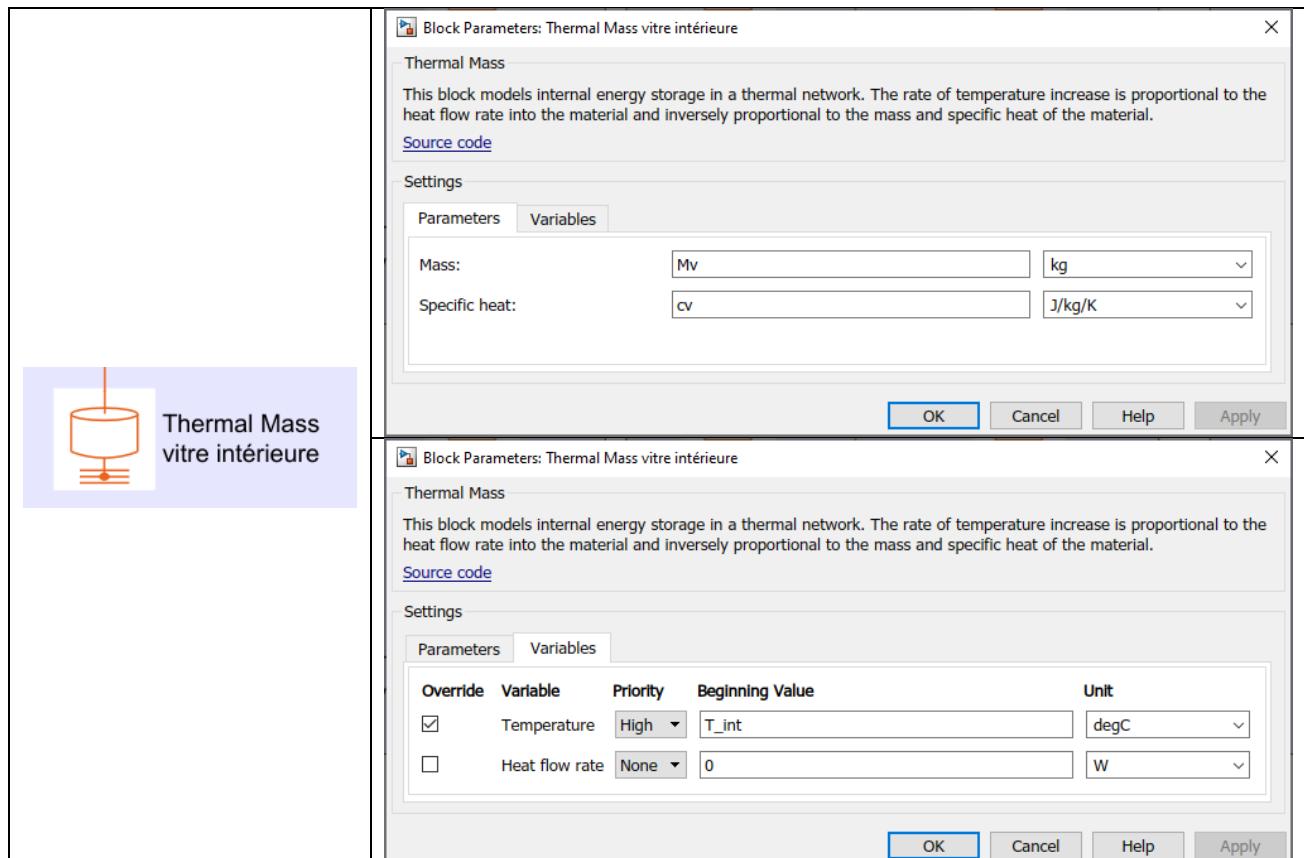


Figure 313 : paramétrage du bloc **Masse thermique de la vitre intérieure**

Paramétrage du bloc : *Conduction de la couche d'argon*

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur d'argon

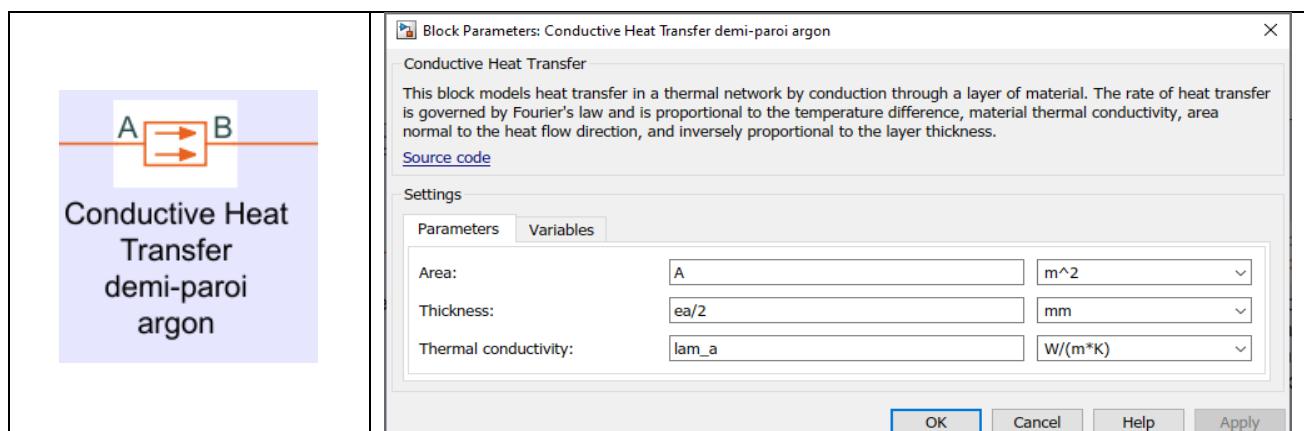


Figure 314 : paramétrage du bloc **Conduction de la couche d'argon**

Paramétrage du bloc : Masse thermique de l'argon

Ce bloc modélise la masse thermique de l'argon. Il y a deux onglets à paramétrier. L'onglet **Parameters** qui permet de renseigner la masse de l'argon et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que l'argon est à la température $(T_{int}+T_{ext})/2$ en début de simulation.

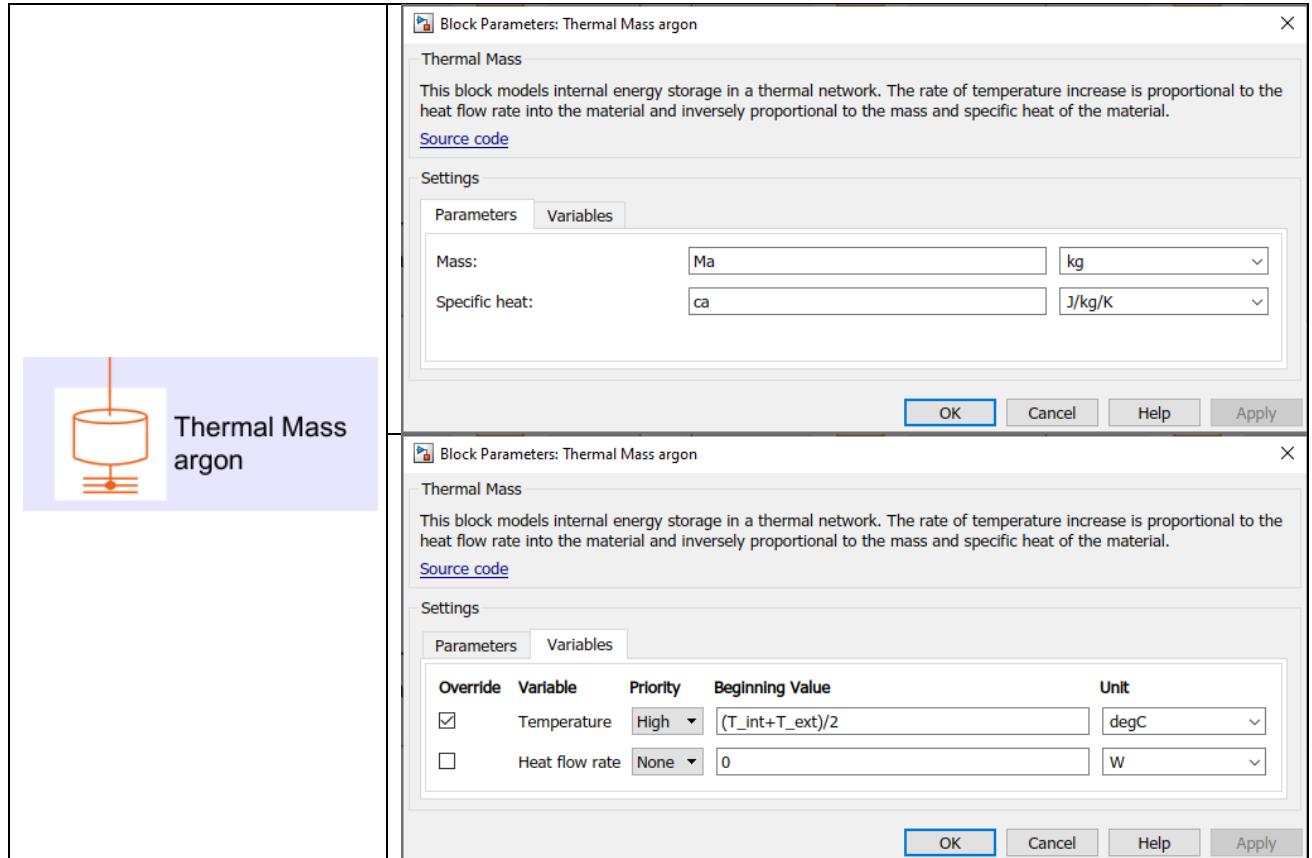


Figure 315 : paramétrage du bloc Masse thermique de l'argon

Paramétrage du bloc : Conduction du verre de la vitre extérieure

Ces deux blocs modélisent le transfert thermique par conduction dans une demi épaisseur de verre de la vitre extérieure.

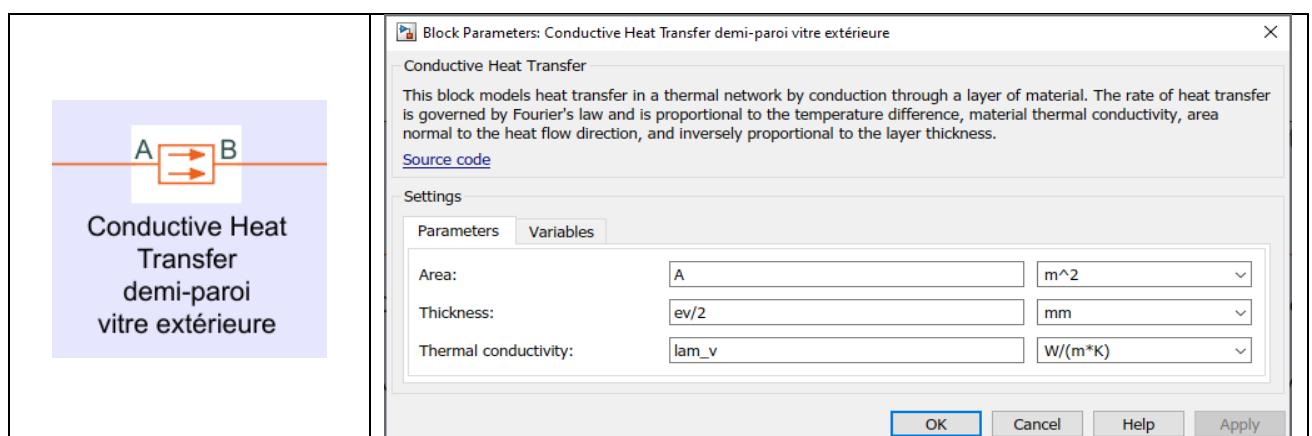


Figure 316 : paramétrage du bloc Conduction de la vitre extérieure

Paramétrage du bloc : *Masse thermique de la vitre extérieure*

Ce bloc modélise la masse thermique du verre de la vitre extérieure. Il y a deux onglets à paramétriser. L'onglet **Parameters** qui permet de renseigner la masse du verre et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que le verre de la vitre intérieure est à la même température que l'extérieur en début de simulation.

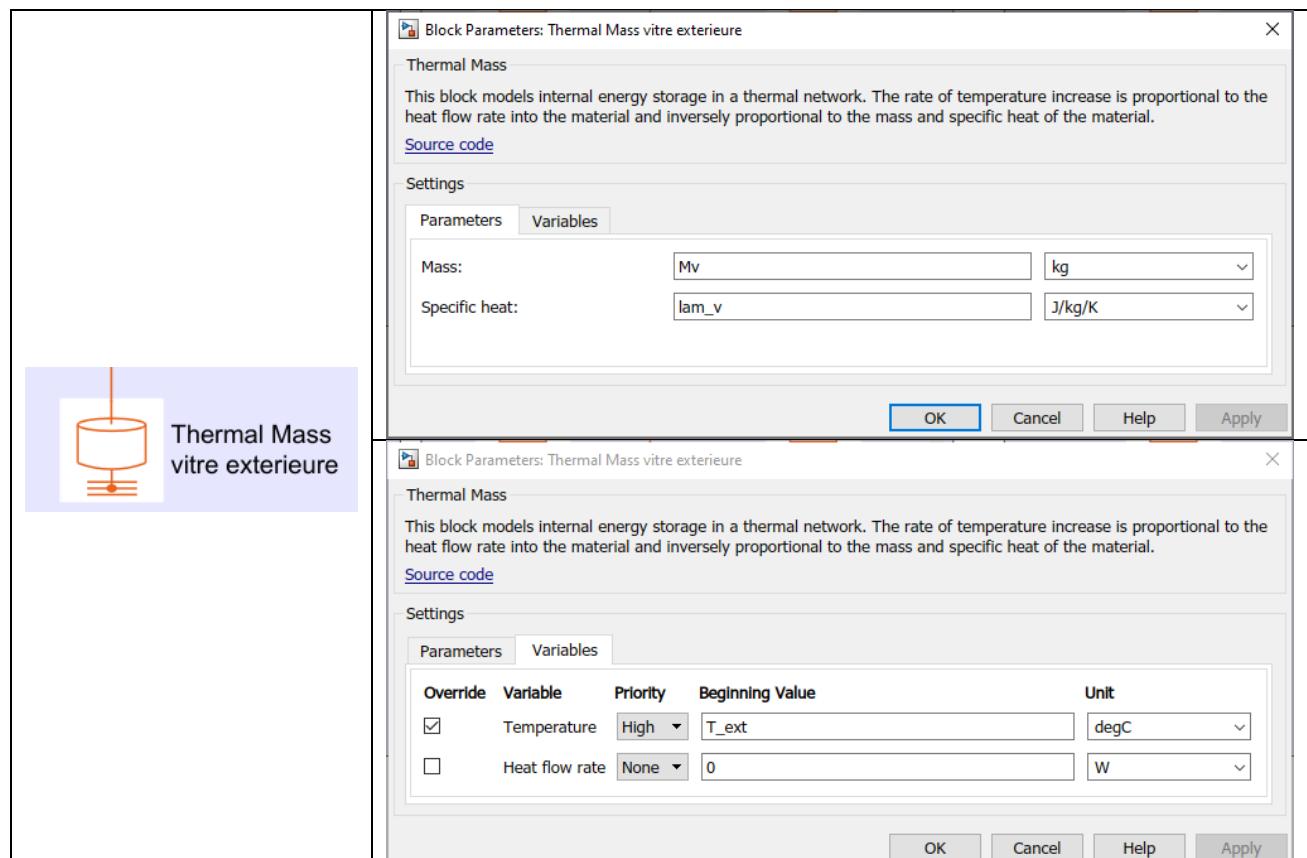
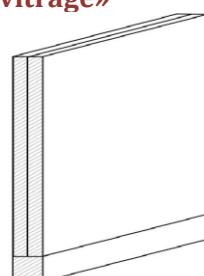


Figure 317 : paramétrage du bloc **Masse thermique de la vitre extérieure**

Paramétrage du sous-système « Double vitrage »



Paramétrage du bloc : Conduction du verre de la vitre intérieure/extérieur

Ces quatre blocs modélisent le transfert thermique par conduction dans une demi épaisseur de verre.

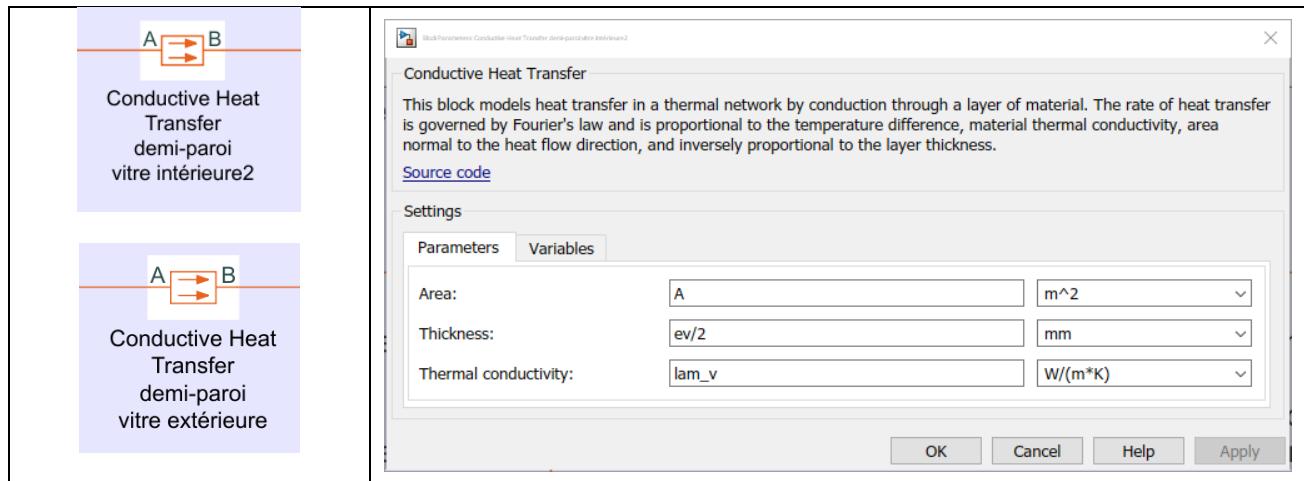


Figure 318 : paramétrage du bloc Conduction de la vitre intérieure

Paramétrage du bloc : Masse thermique de la vitre intérieure

Ce bloc modélise la masse thermique du verre de la vitre intérieure. Il y a deux onglets à paramétriser. L'onglet **Parameters** qui permet de renseigner la masse du verre et à sa chaleur spécifique. L'onglet **Variables** qui permet de fixer la température de la masse thermique en début de simulation. Nous considérons ici que le verre de la vitre intérieure est à la même température que l'intérieur de la pièce en début de simulation.

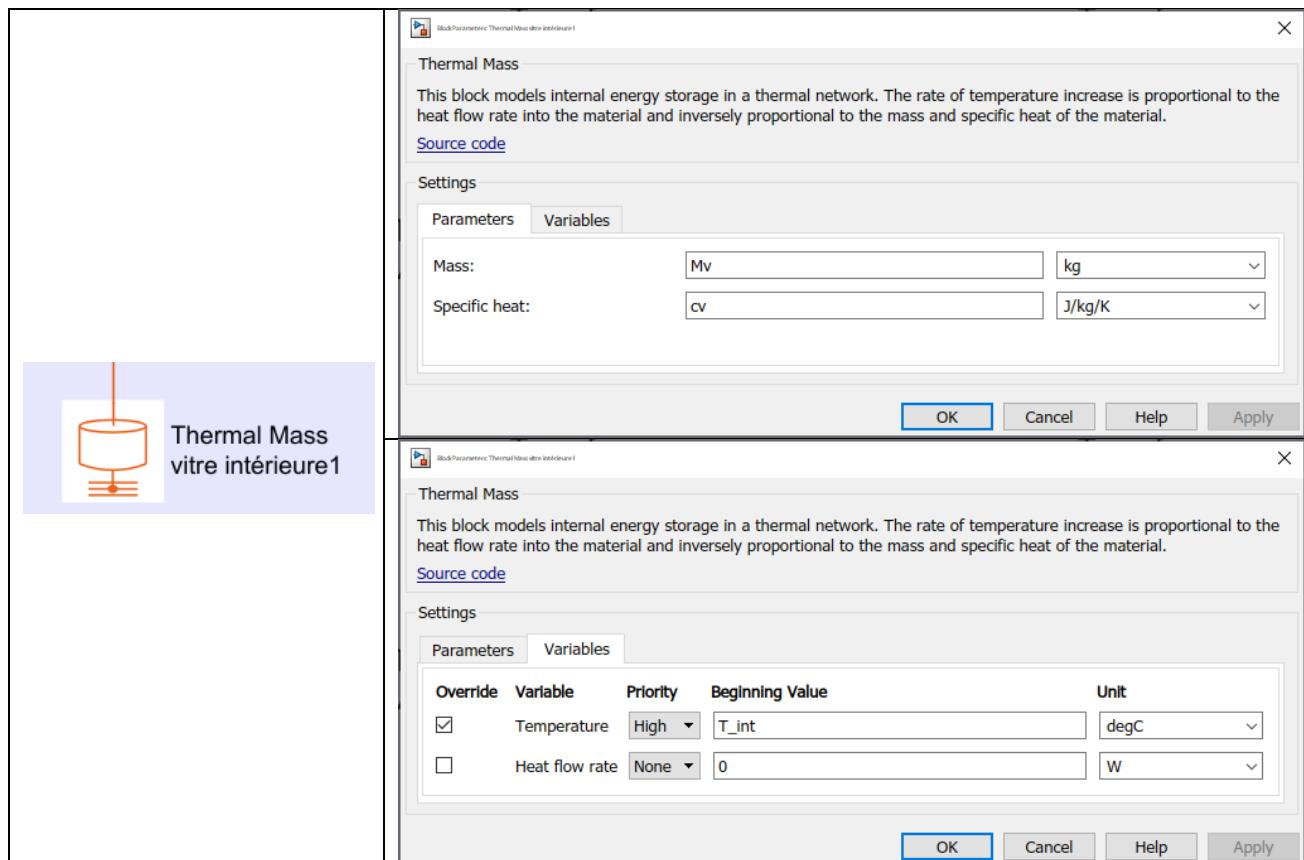


Figure 319 : paramétrage du bloc Masse thermique de la vitre intérieure

Le fichier « **double_vitrage_1.slx** » contient le modèle entièrement paramétré et prêt à être simulé.

Pour être certain d'obtenir la valeur en régime permanent du flux thermique, le temps de simulation est réglé à 24 heures.

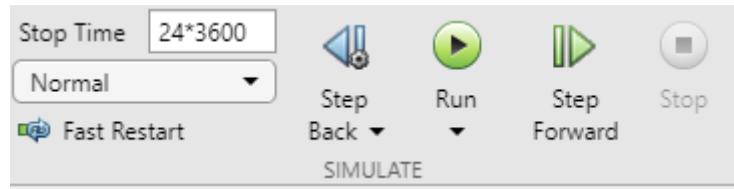


Figure 320 : réglage du temps de simulation

Lancer la simulation en cliquant sur .

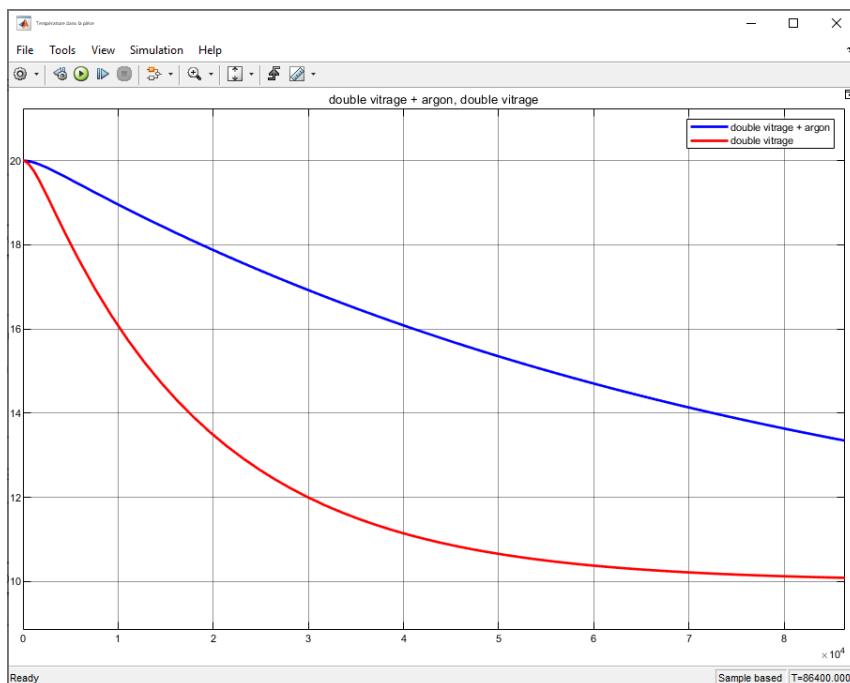


Figure 321 : évolution de la température dans la pièce en fonction du temps pour les deux types de vitrage

Nous pouvons constater l'efficacité thermique du double vitrage + argon qui permet à la température de la pièce de se maintenir plus durablement.

Il est très simple à ce stade de choisir les paramètres du modèle à faire varier en modifiant les valeurs des paramètres dans le script.

22. Modélisation thermique d'une maison

Les paragraphes précédents ont présenté la théorie et les concepts permettant de modéliser les différents types d'échanges thermiques que l'on peut rencontrer dans la modélisation thermique d'une maison. Il est maintenant possible de comprendre l'exemple de la documentation Mathworks® qui modélise le chauffage d'une maison en prenant en compte les échanges thermiques par conduction et convection au travers du toit, des murs et des fenêtres.

Afin d'accéder à cet exemple, il faut ouvrir la documentation de **MATLAB** à partir de la barre de commande principale du logiciel.

- Cliquer sur le point d'interrogation  dans la barre de commande de **MATLAB** afin d'ouvrir la **Documentation**.
- Choisir l'onglet **Examples** dans la partie supérieure de la fenêtre et faire défiler le menu **Category** dans la partie gauche de l'écran pour faire apparaître **Simscape**
- Sélectionner **Simscape** afin que les exemples Simscape soit consultables sur la page principale
- Faire défiler les exemples jusqu'à voir apparaître les modèles thermiques de Simscape (Figure 322).

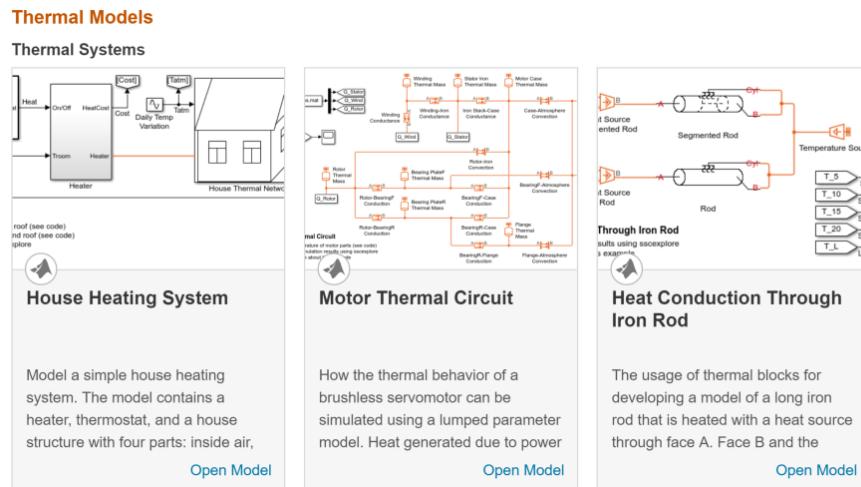


Figure 322 : modèles Simscape Thermal

Cliquer sur **Open Model** pour l'exemple **House Heating System**© afin de faire apparaître le modèle thermique d'une maison et la régulation de son système de chauffage.

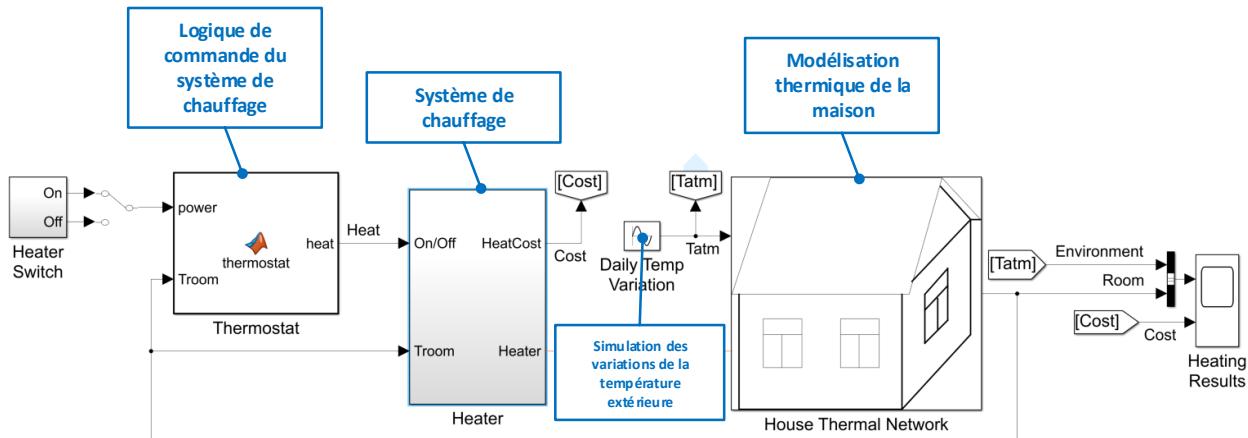


Figure 323 : modèle House Heating System , Mathworks ©

Ce modèle permet de simuler le fonctionnement du système de chauffage de la maison et de sa logique de commande. Le comportement thermique de la maison est entièrement modélisé à l'aide de Simscape. En cliquant sur le sous-système **House Thermal Networks**, il est possible de visualiser le modèle Simscape de la maison (Figure 324).

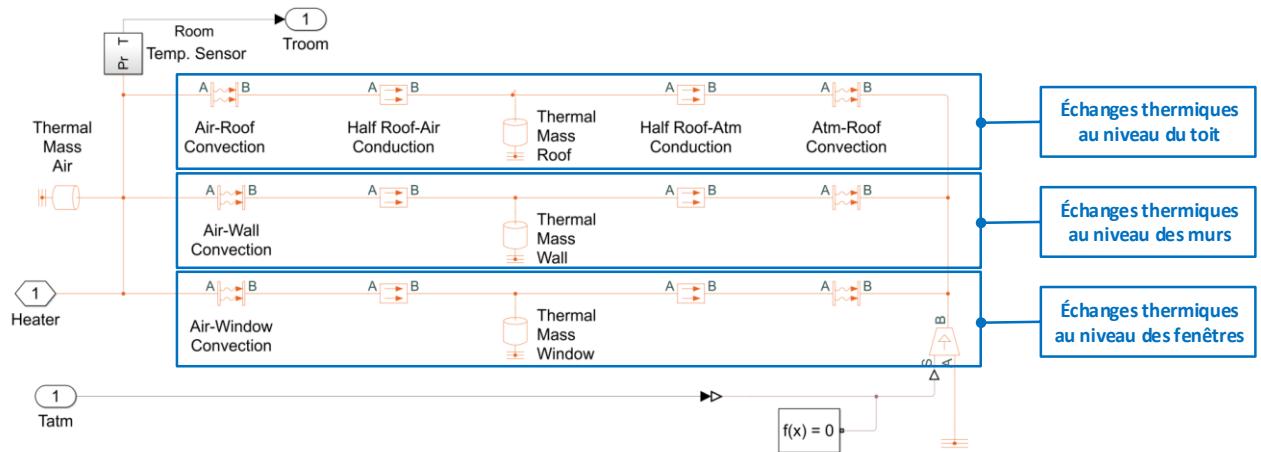


Figure 324 : modèle thermique de la maison réalisé avec Simscape

On retrouve sur la Figure 324 les structures des modèles présentées dans les paragraphes précédents et qui permettent de prendre en compte les échanges thermiques par conduction et convection au travers du toit, des murs et des fenêtres.

Le modèle permet de voir comment il est possible de stabiliser la température à l'intérieur de la maison en fonction des fluctuations de la température extérieure. Il sera alors possible de valider les performances du modèle :

- Fonctionnement de la logique de commande du système de chauffage
- Capacité du système de chauffage à fournir une puissance suffisante
- Fluctuation de la température à l'intérieur de la maison...

Il sera également possible d'avoir accès à de nombreuses informations techniques utiles lors du processus de conception :

- Température du mur, du toit des fenêtres au centre des parois
- Flux thermique au travers des différentes parois...

Lancer la simulation et ouvrir le scope (Figure 325).

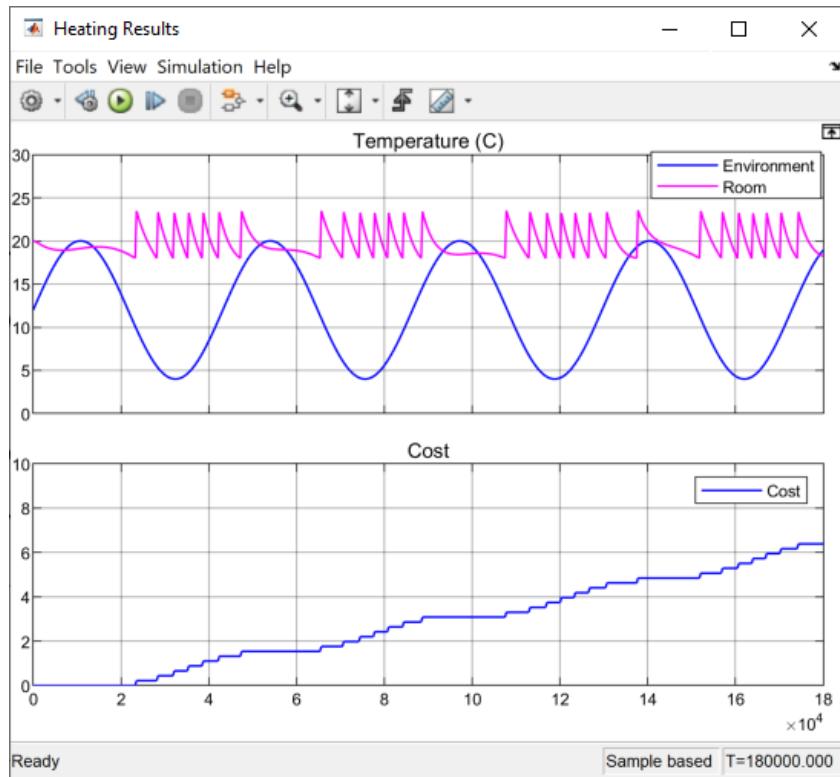


Figure 325 : évolution des températures intérieures et extérieures et du coût du chauffage

L'analyse de la Figure 325 nous permet de voir que la température à l'intérieur de la maison reste approximativement constante autour de 20°C avec une amplitude de variation de 5°C maximum alors que la température extérieure présente des variations de l'ordre de 15°C.

En cliquant sur **Plot Temperature** dans le modèle il est possible de visualiser les températures des différentes parois de la maison (Figure 326).

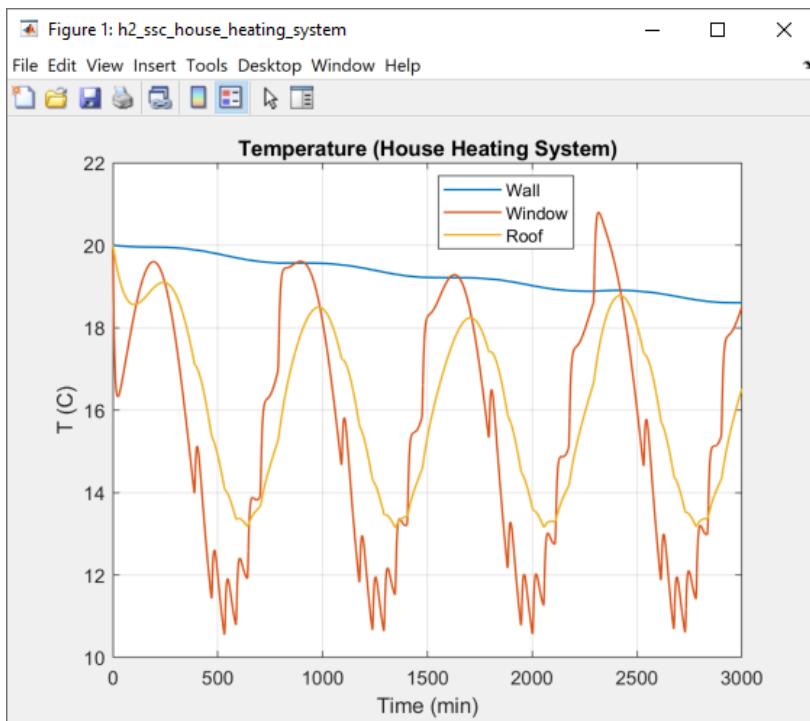


Figure 326 : température des différentes parois de la maison

En cliquant sur **Plot Heat Flow** dans le modèle il est possible de visualiser les flux de chaleur au travers des différentes parois de la maison (Figure 327).

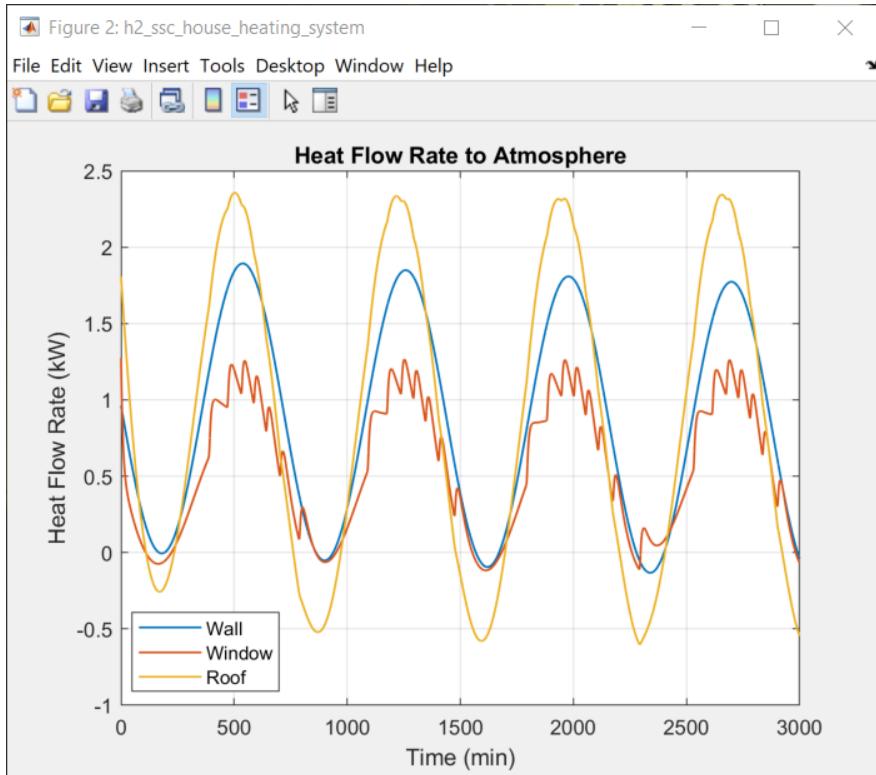


Figure 327 : flux de chaleur au travers des différentes parois de la maison

Il est également possible en double-cliquant sur l'interrupteur **Heater Switch** de couper le système de chauffage.

Placer l'interrupteur sur la position **Off** et relancer la simulation.
Ouvrir le scope et visualiser le résultat sur la Figure 328.

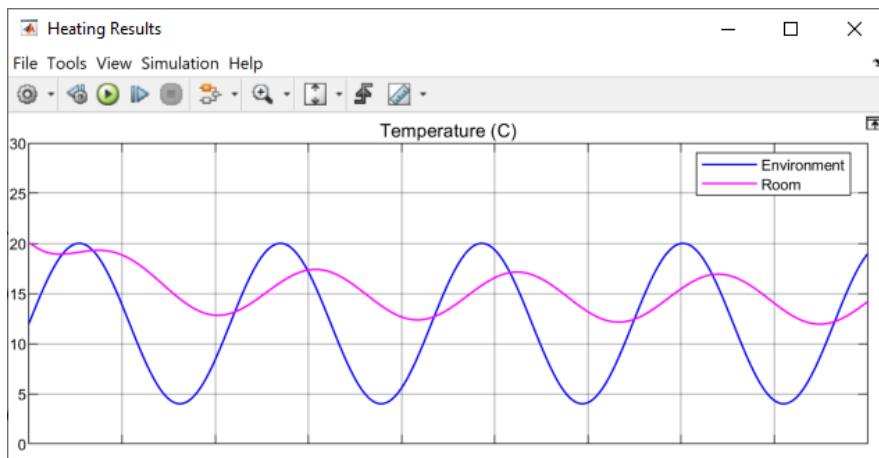


Figure 328 : évolution de la température sans système de chauffage

Nous constatons que sans le système de chauffage la fluctuation de la température intérieure augmente et que la température moyenne est plutôt de l'ordre de 15°C.

Il est ainsi possible de tester un grand nombre de scénarios afin d'optimiser les performances du modèle et le coût du système de chauffage.

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 formation.

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 329).

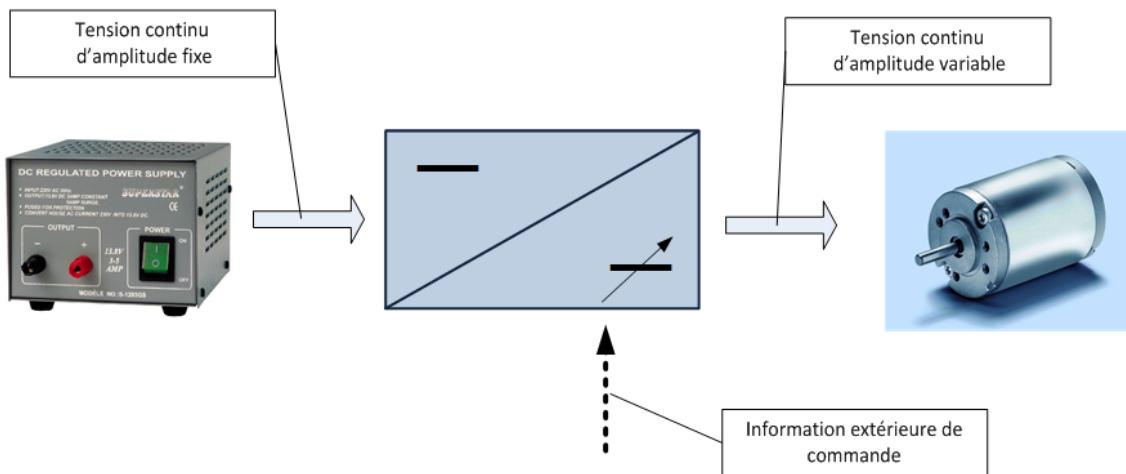


Figure 329 : présentation du hacheur série

Le hacheur va convertir une tension continue en une tension qui prend la forme d'un signal carré (Figure 330)

- T : période de hachage (ou de découpage)
- $\alpha = \frac{t_{on}}{T}$: rapport cyclique, si $\alpha = 1$, 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).

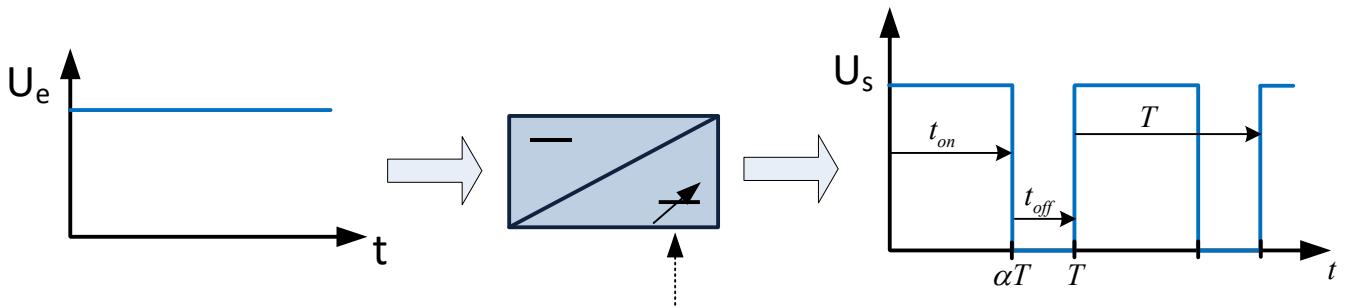


Figure 330 : principe de fonctionnement du hacheur série

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

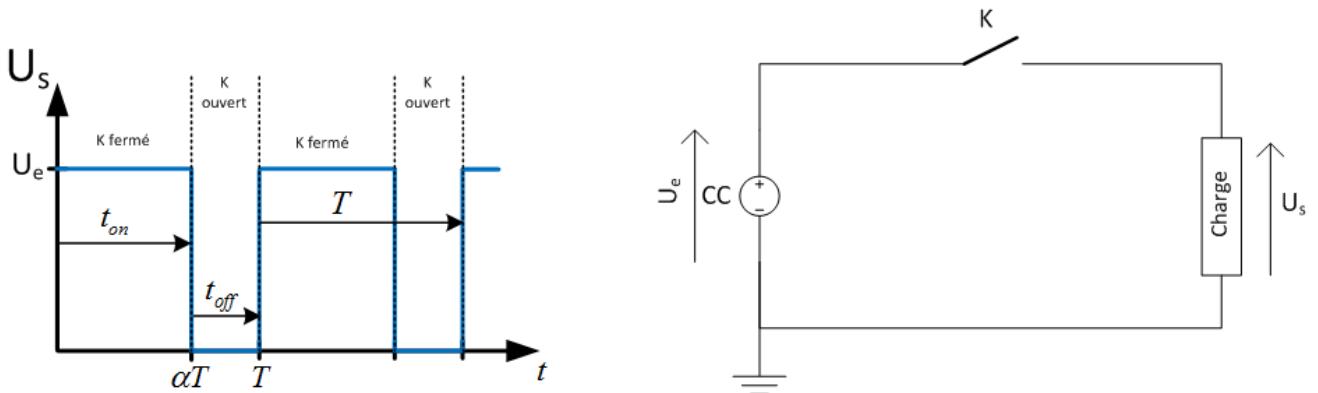


Figure 331 : schéma de principe du hacheur série

Durant une période T :

- la charge est alimentée durant la durée t_{on} , c'est la phase active
- l'alimentation est coupée durant la durée 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 $\langle U_c \rangle = \alpha U_e$.

Le hacheur série permettant de commander un moteur à courant continu doit comporter 2 interrupteurs (Figure 332) :

- 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 K₁ est équivalent à un interrupteur fermé. La diode K₂ est bloquée (VK>VA) et se comporte comme un interrupteur fermé.

En phase de roue libre, le transistor K₁ est équivalent à un interrupteur ouvert. La diode K₂ 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)

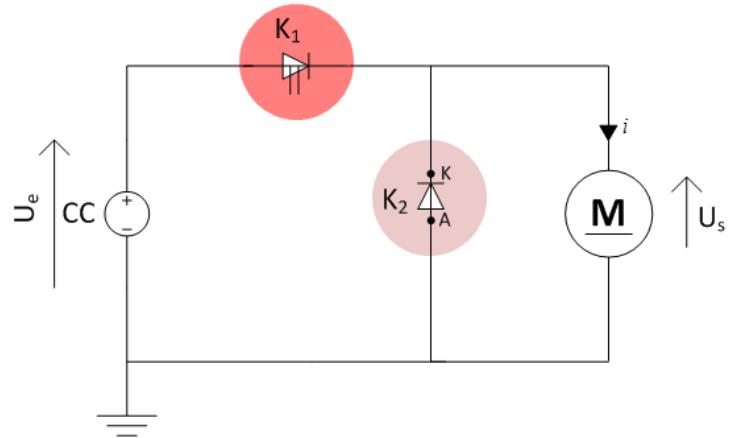
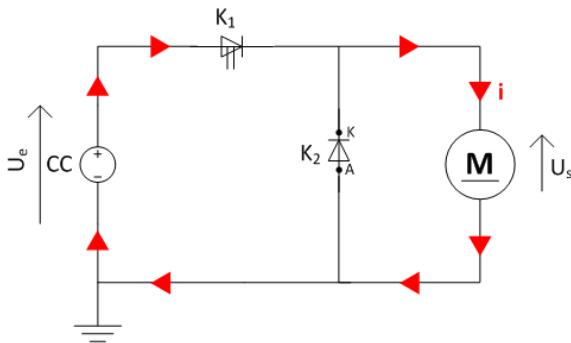


Figure 332 : schéma de commande d'un moteur à courant par un hacheur série

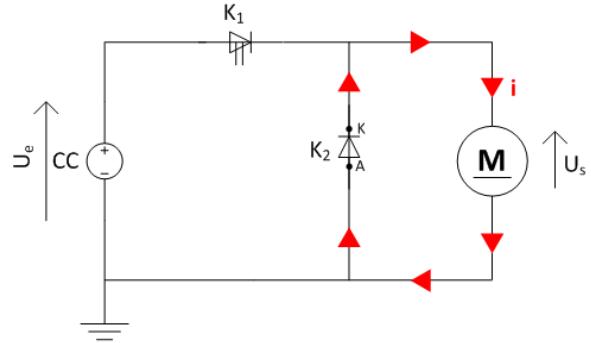
B. Objectifs pédagogiques

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 333).



Phase active



Phase de roue libre

Figure 333 : 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 334)

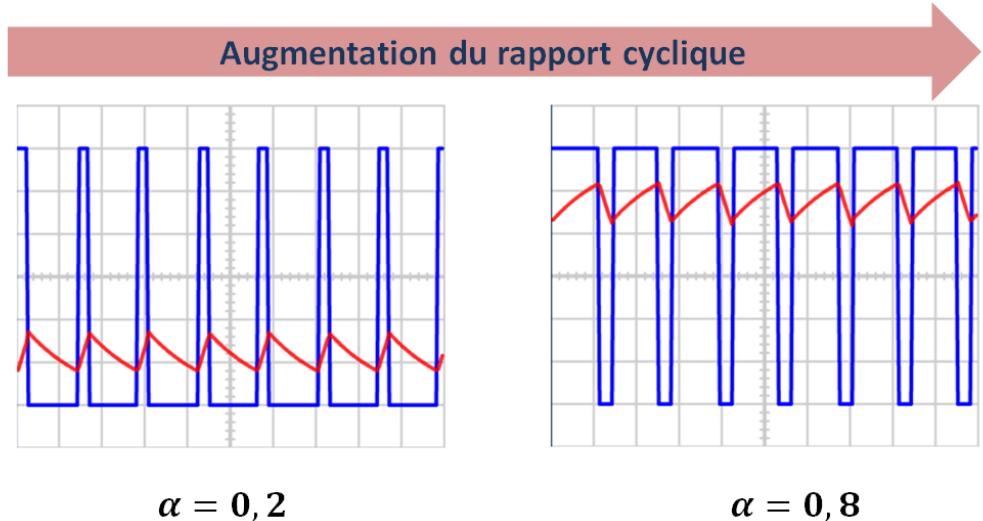


Figure 334 : visualisation de l'influence du rapport cyclique sur l'ondulation de courant

Objectif 3 : Visualiser et évaluer l'influence de la fréquence de hachage sur l'ondulation du courant (Figure 335)

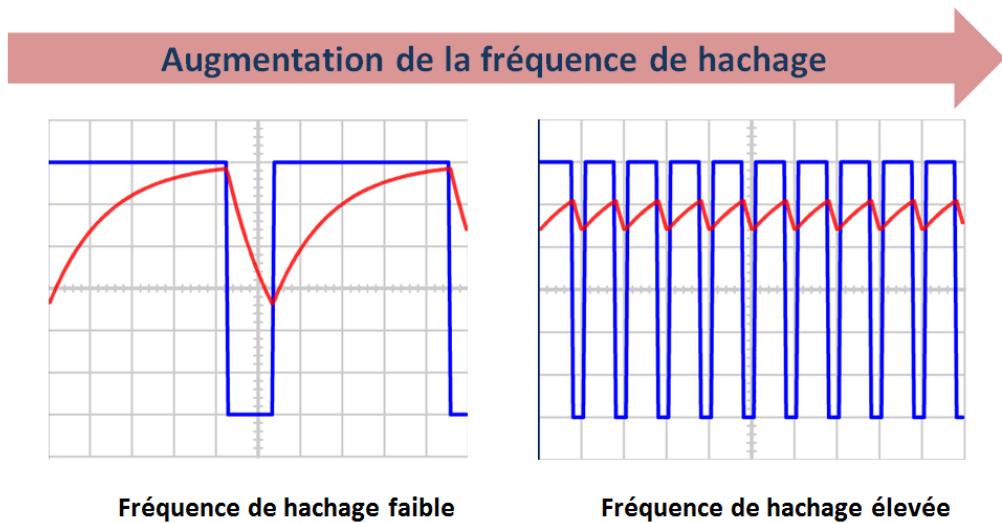


Figure 335 : visualisation de l'influence de la fréquence de hachage sur l'ondulation de courant

Objectif 4 : Visualiser et évaluer l'influence de la valeur de l'inductance sur l'ondulation du courant (Figure 336)

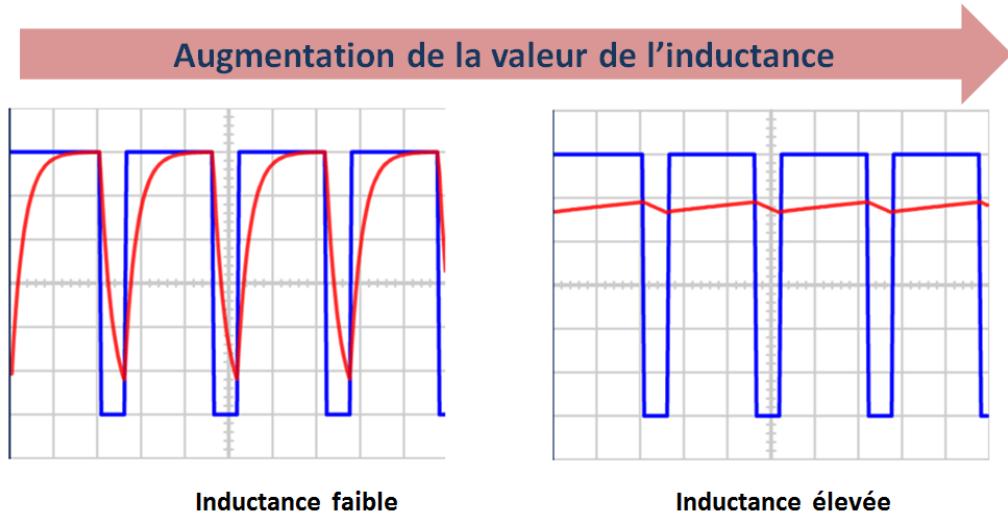


Figure 336 : visualisation de l'influence de la valeur de l'inductance sur l'ondulation de courant

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 337). L'étudiant identifie facilement tous les composants et peut faire le lien avec ses connaissances théoriques.

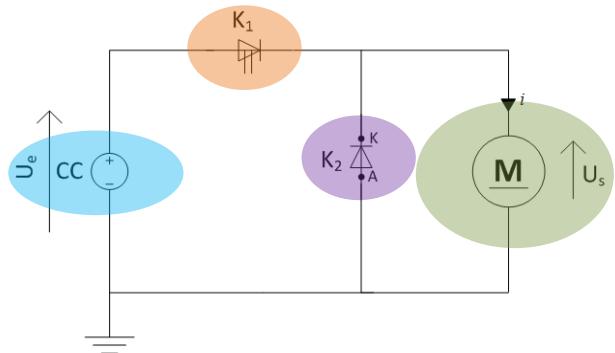
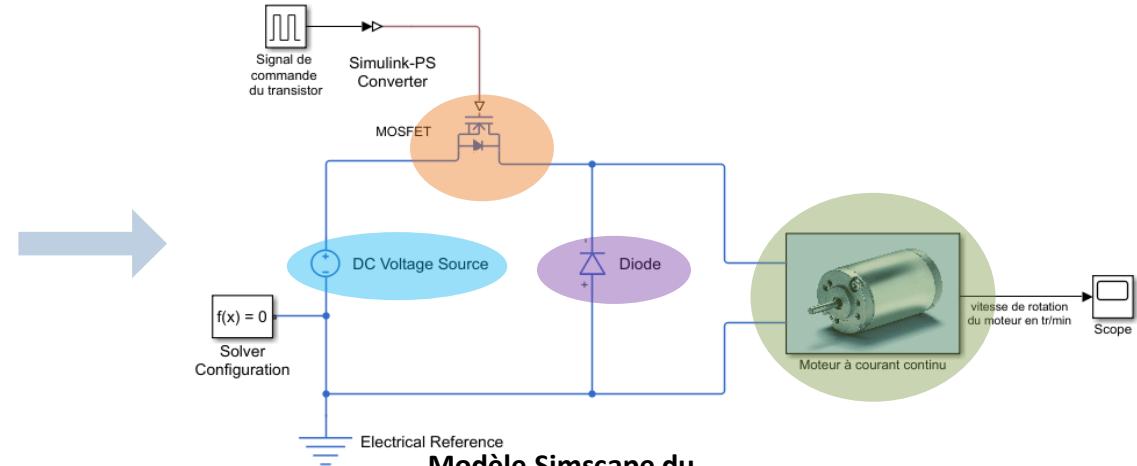


Schéma électrique de principe du hacheur série



Modèle Simscape du hacheur série

Figure 337 : analogie entre la forme du modèle Simscape et le schéma électrique de principe

Ouvrir le modèle « hacheur-série_0.slx ».

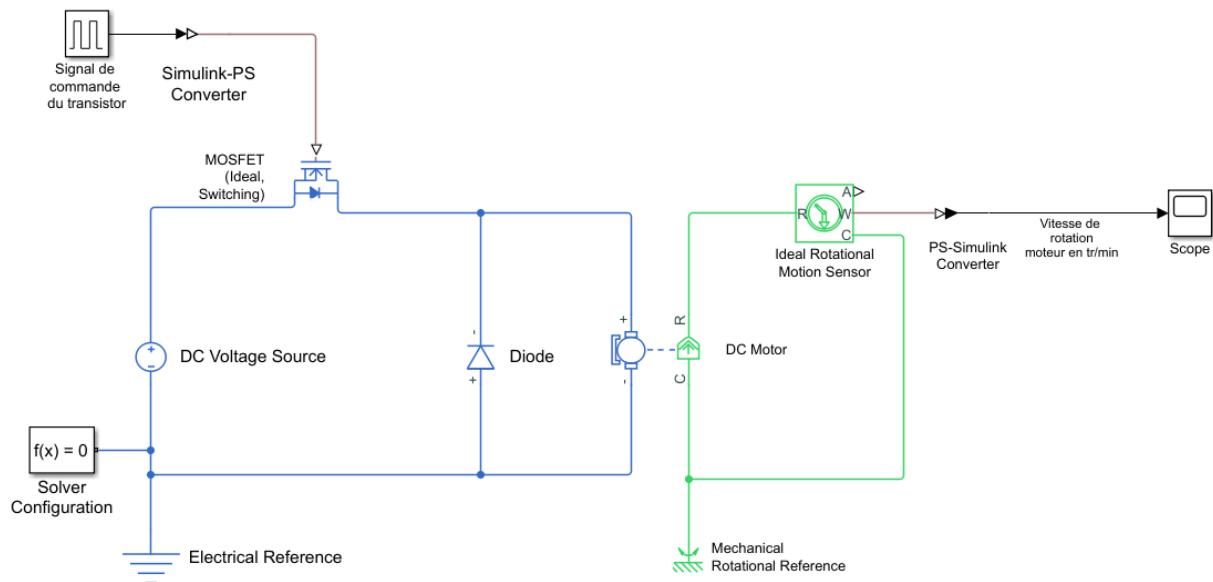


Figure 338 : modèle Simscape du hacheur série

Seuls sont référencés dans le tableau de la Figure 339 les composants qui n'ont pas encore été utilisés dans l'ouvrage.

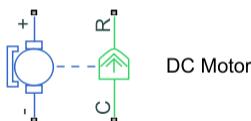
Fonction du composant	Représentation	Bibliothèque
Transistor MOSFET (Ideal Switching)		Simscape/Electrical/Components/SemiConductors & Converters
Diode		Simscape/Foundation Library/Electrical/Electrical Elements
Générateur de signaux carrés		Simulink/Sources

Figure 339 : les composants nécessaires à la modélisation du hacheur série

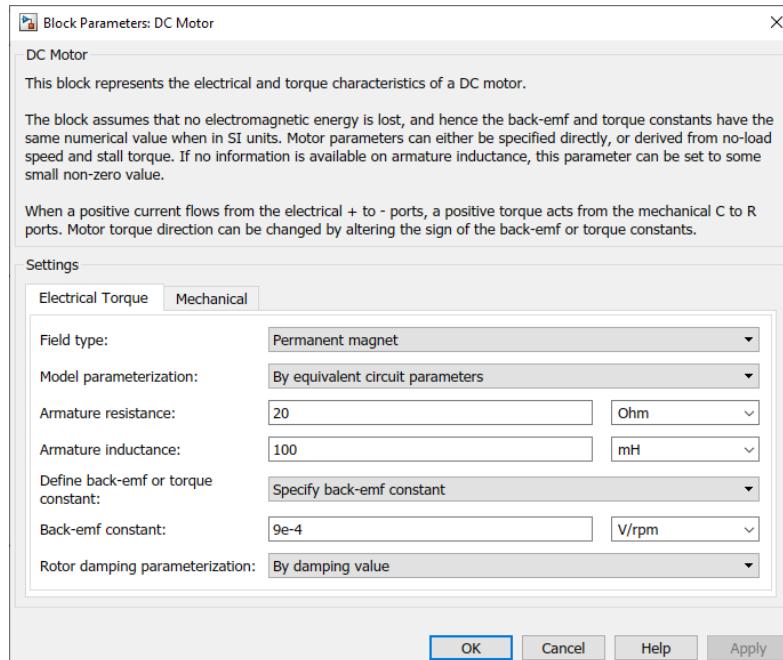
Ce modèle fait intervenir le domaine électrique et le domaine mécanique de rotation.

Paramétrage

DC MOTOR



Simscape/Electrical/Electromechanical/Brushed Motors



Paramétrage

DIODE



Simscape/Foundation Library/Electrical/Electrical Elements

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

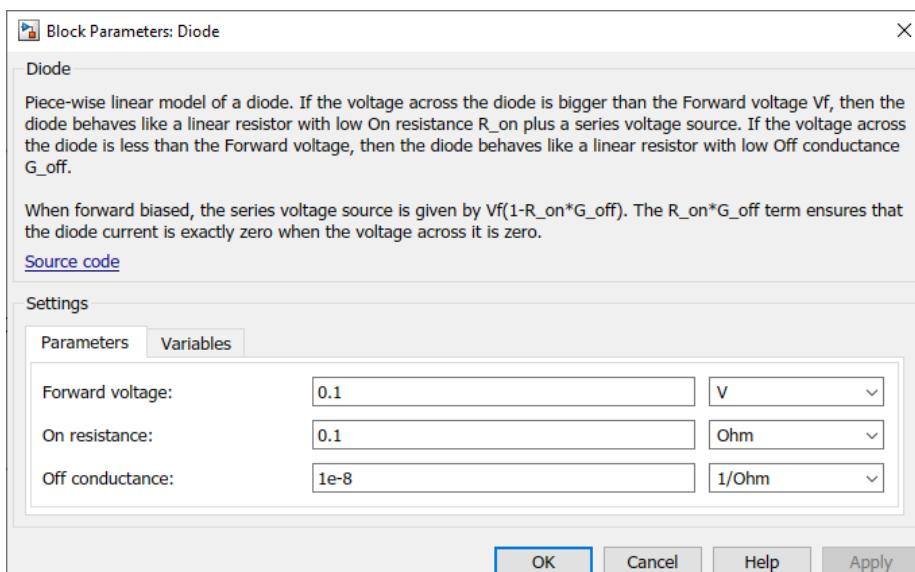


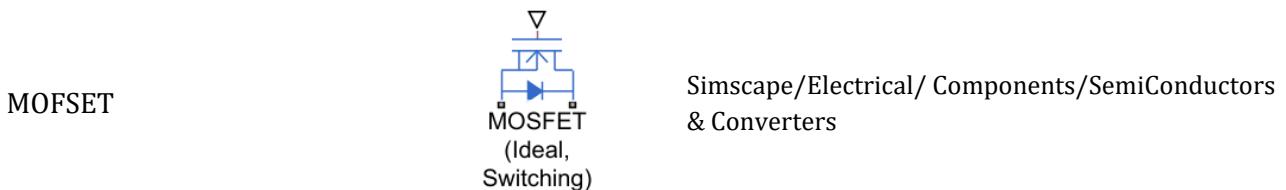
Figure 340 : paramétrage du bloc Diode

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

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

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

Paramétrage



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

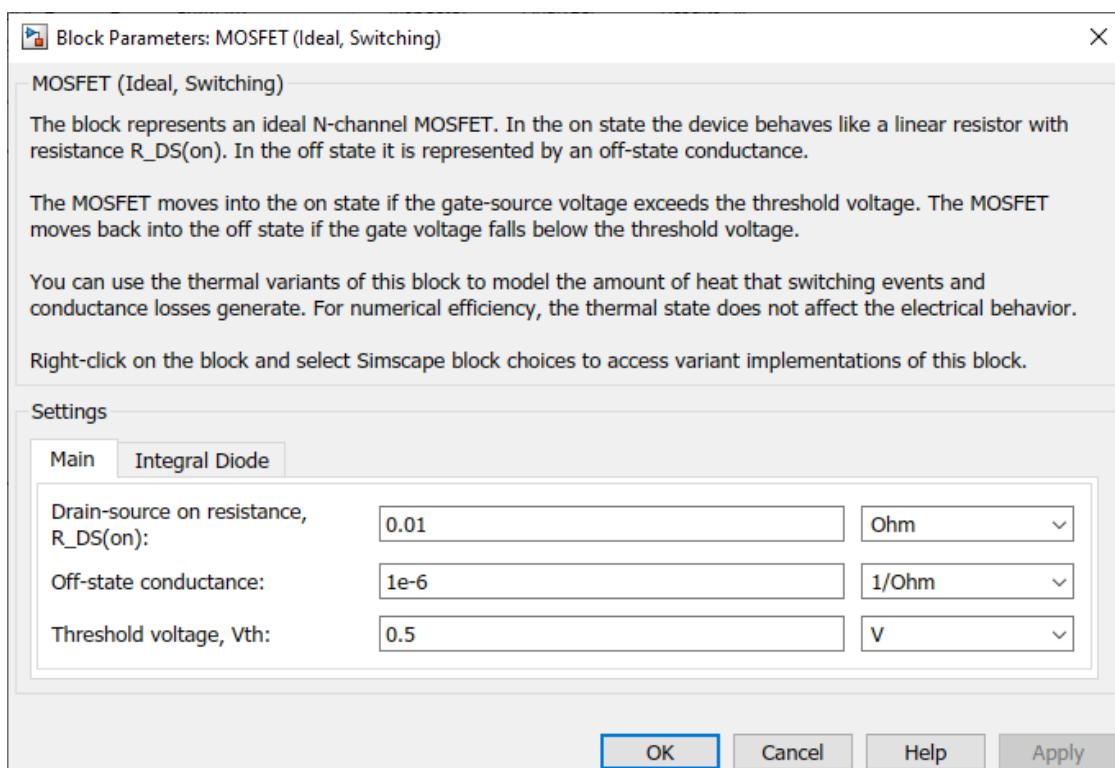


Figure 341 : paramétrage du bloc MOSFET

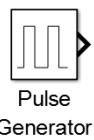
Drain-source on resistance : résistance à l'état passant

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.

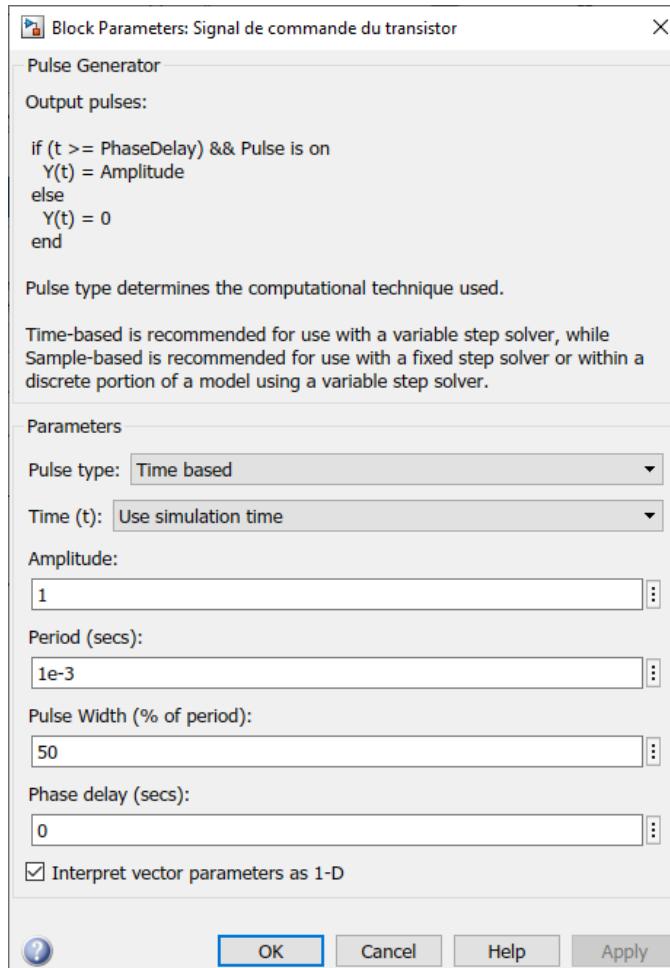


Figure 342 : paramétrage du bloc Pulse Generator

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.

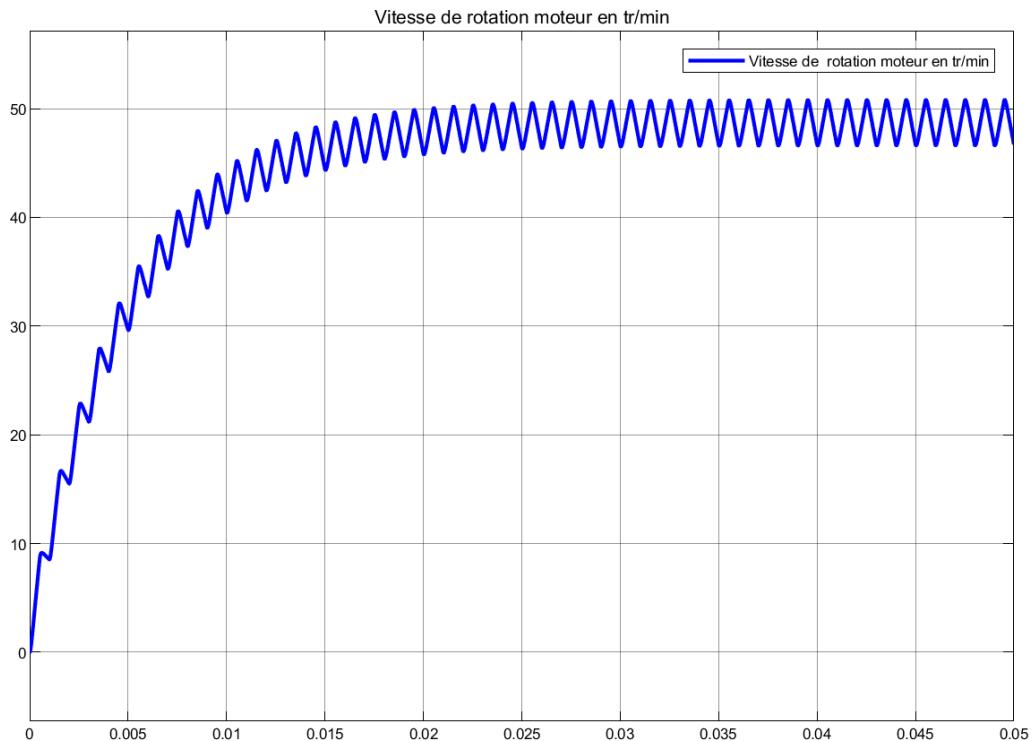


Figure 343 : évolution de la vitesse de rotation du moteur commandé par un hacheur série

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 étapes 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.

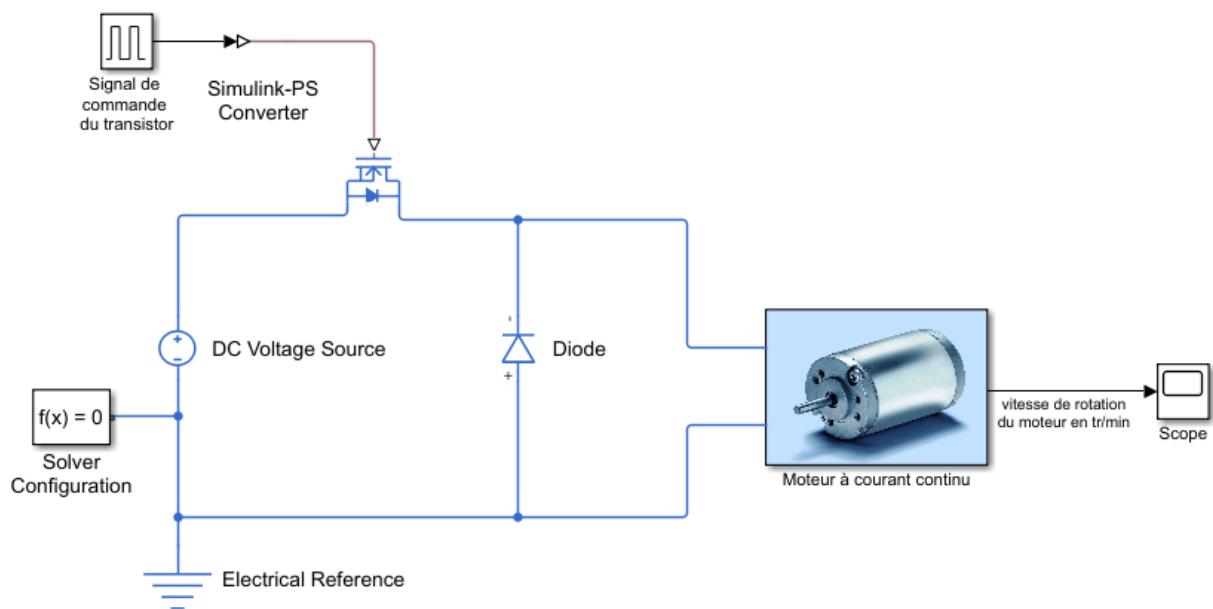


Figure 344 : création d'un sous-système pour didactiser le modèle

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 345).

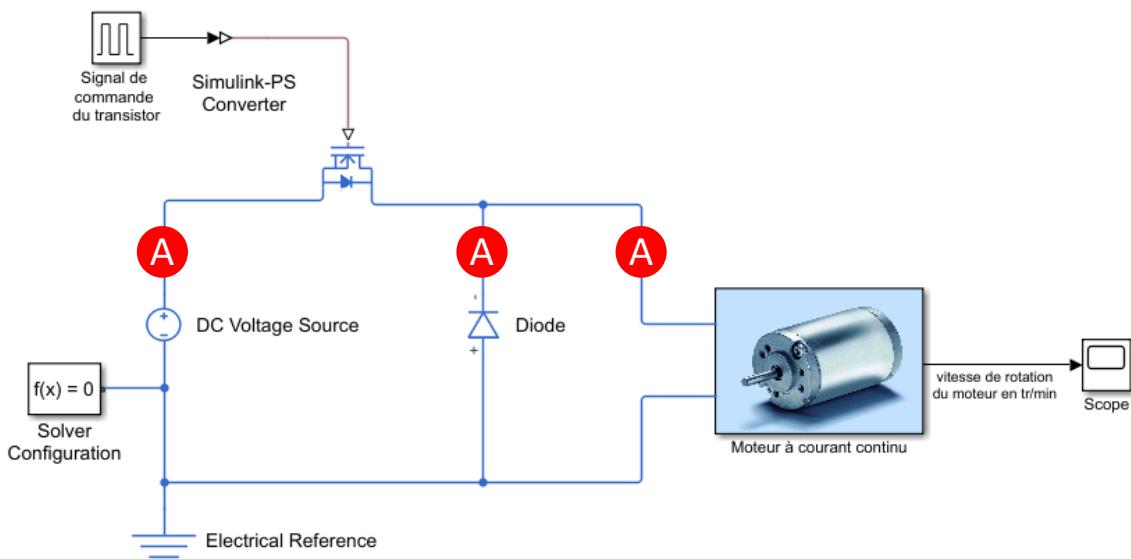


Figure 345 : mise en place des capteurs de courant dans le circuit

Nous pouvons mettre en place le premier capteur pour obtenir le modèle de la Figure 346.

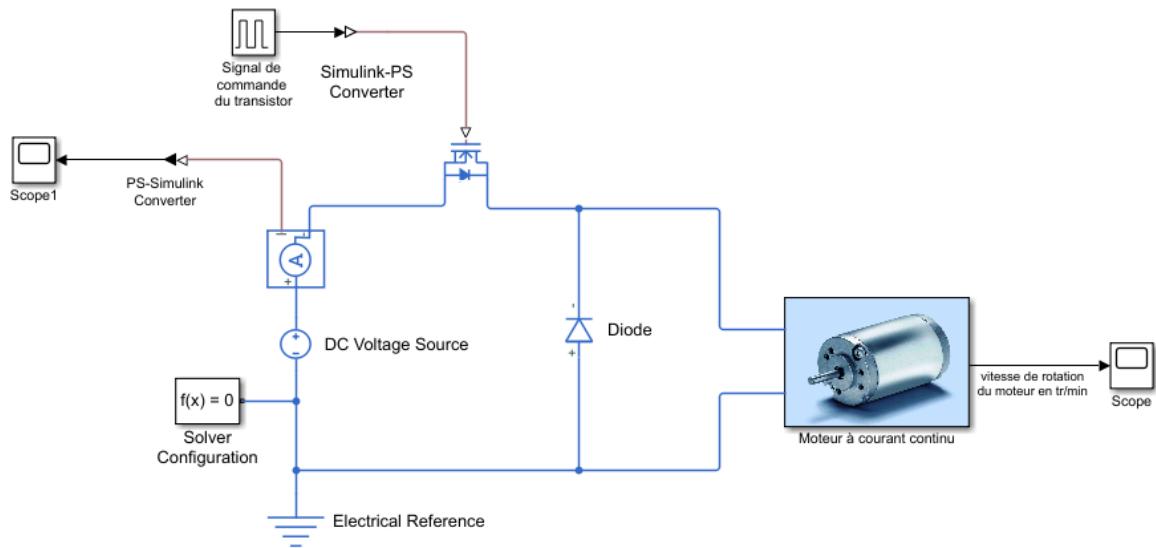


Figure 346 : ajout d'un capteur sur le modèle

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 présentées page 161).

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

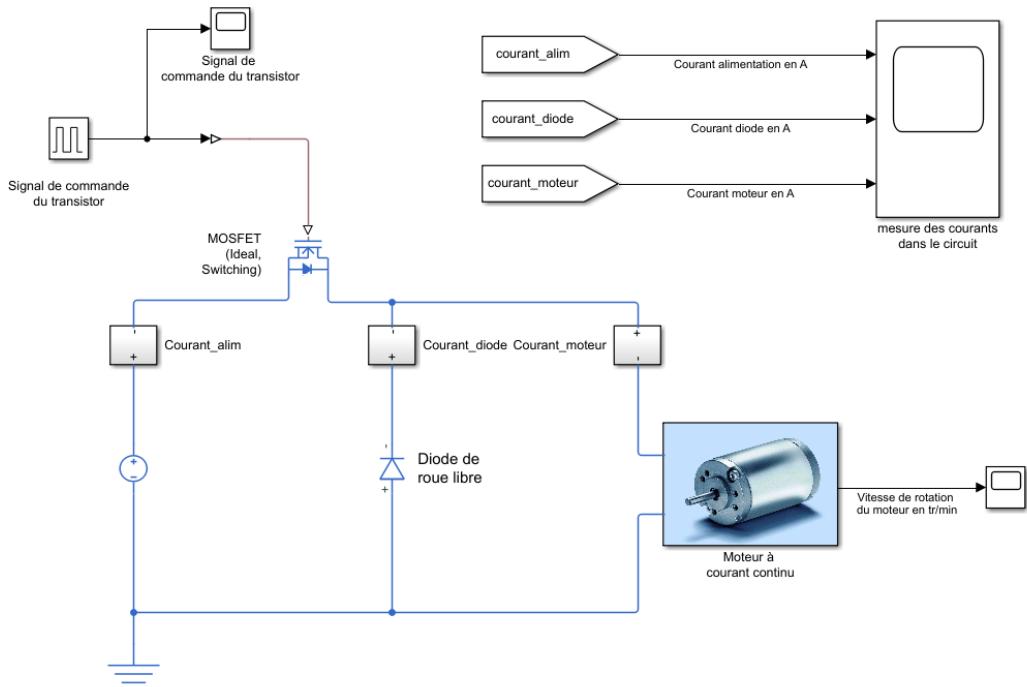


Figure 347 : modèle instrumenté et didactisé

Double-cliquer sur le capteur de courant « **Courant_alim** » et observer le contenu du sous-système (Figure 348).

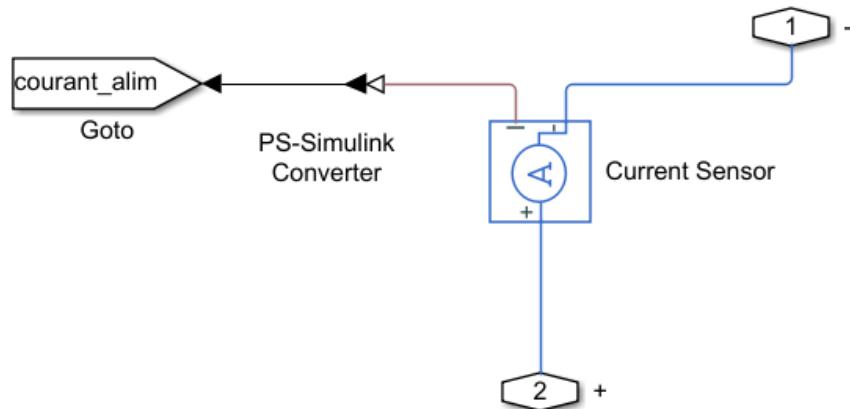


Figure 348 : sous-système « capteur de courant »

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.

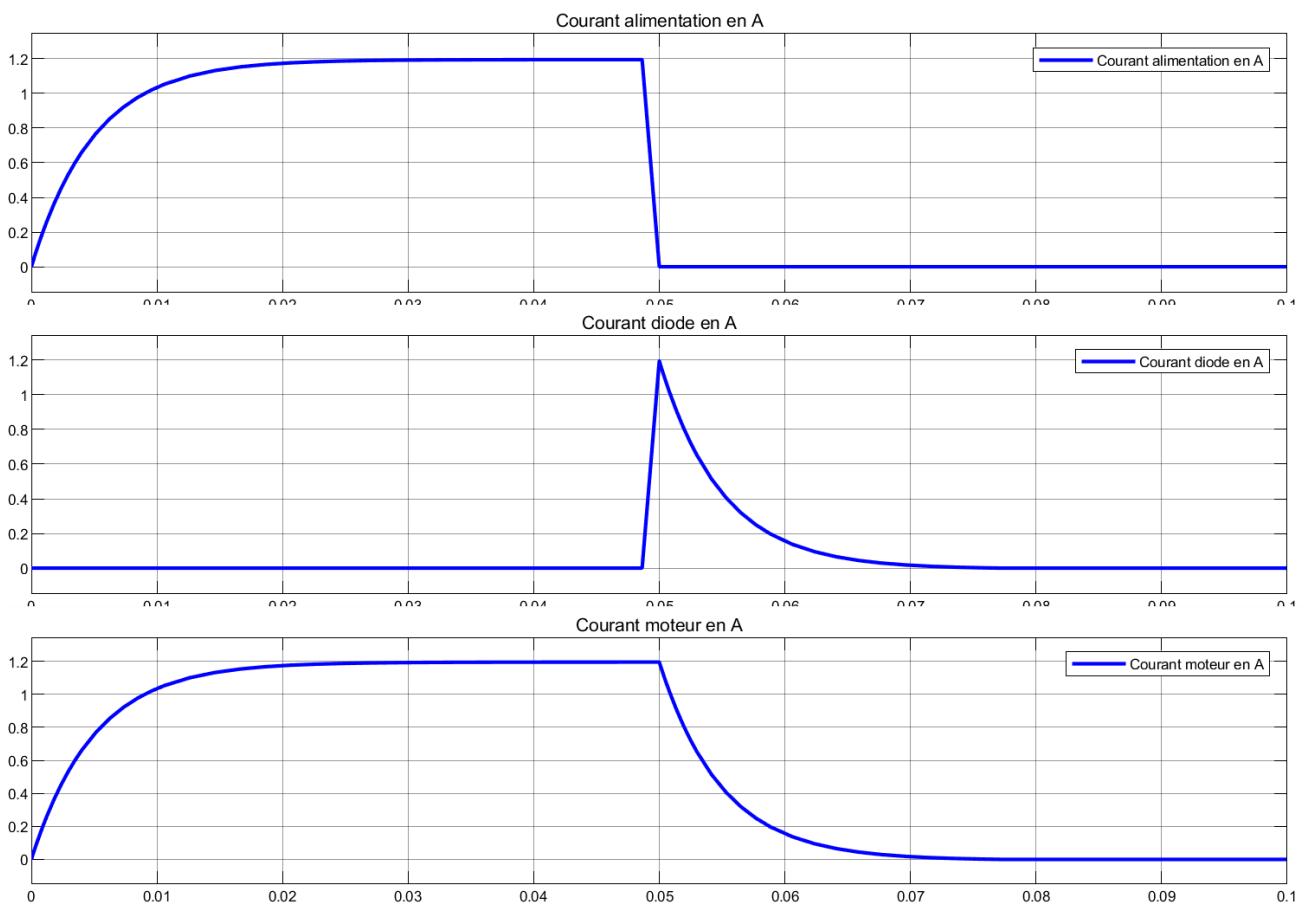


Figure 349 : évolution du courant dans les trois branches du circuit

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

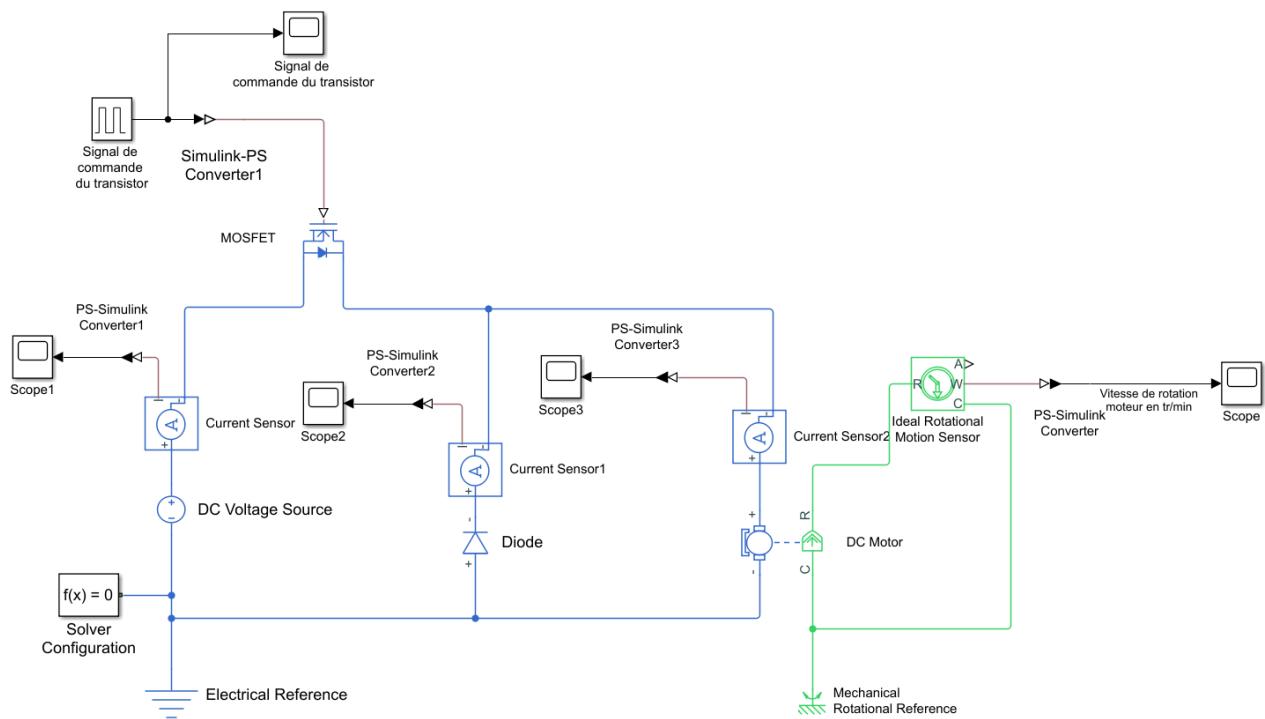


Figure 350 : le modèle non didactisé

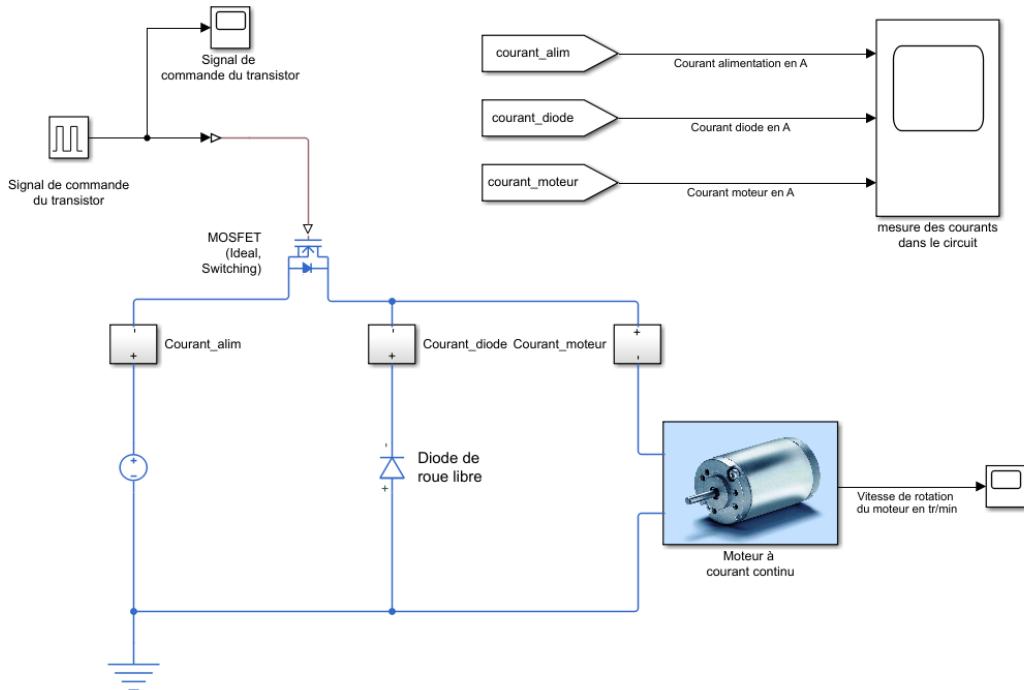


Figure 351 : le modèle didactisé

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 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 352).

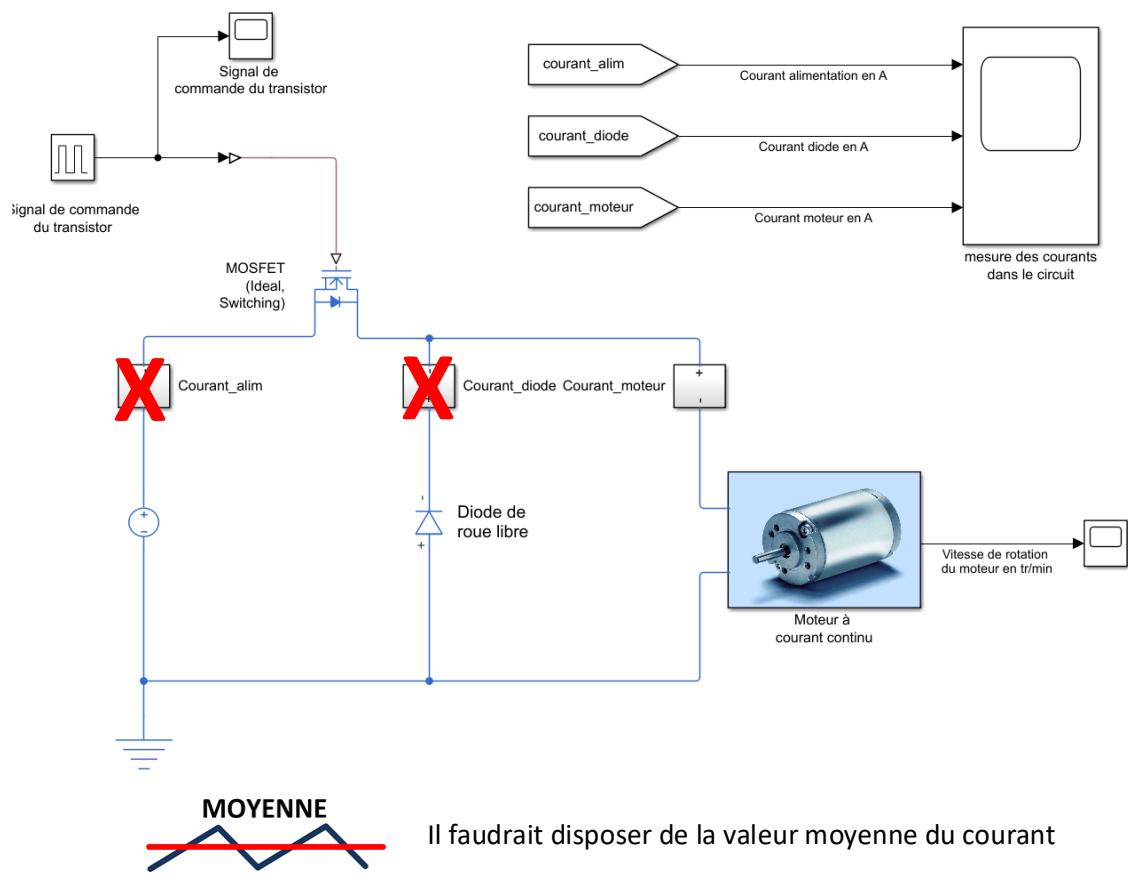


Figure 352 : amélioration de la didactisation du modèle

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

Fonction du composant	Représentation	Bibliothèque
Calcul de la valeur moyenne		Simscape/Electrical/Specialized Power Systems/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 353).

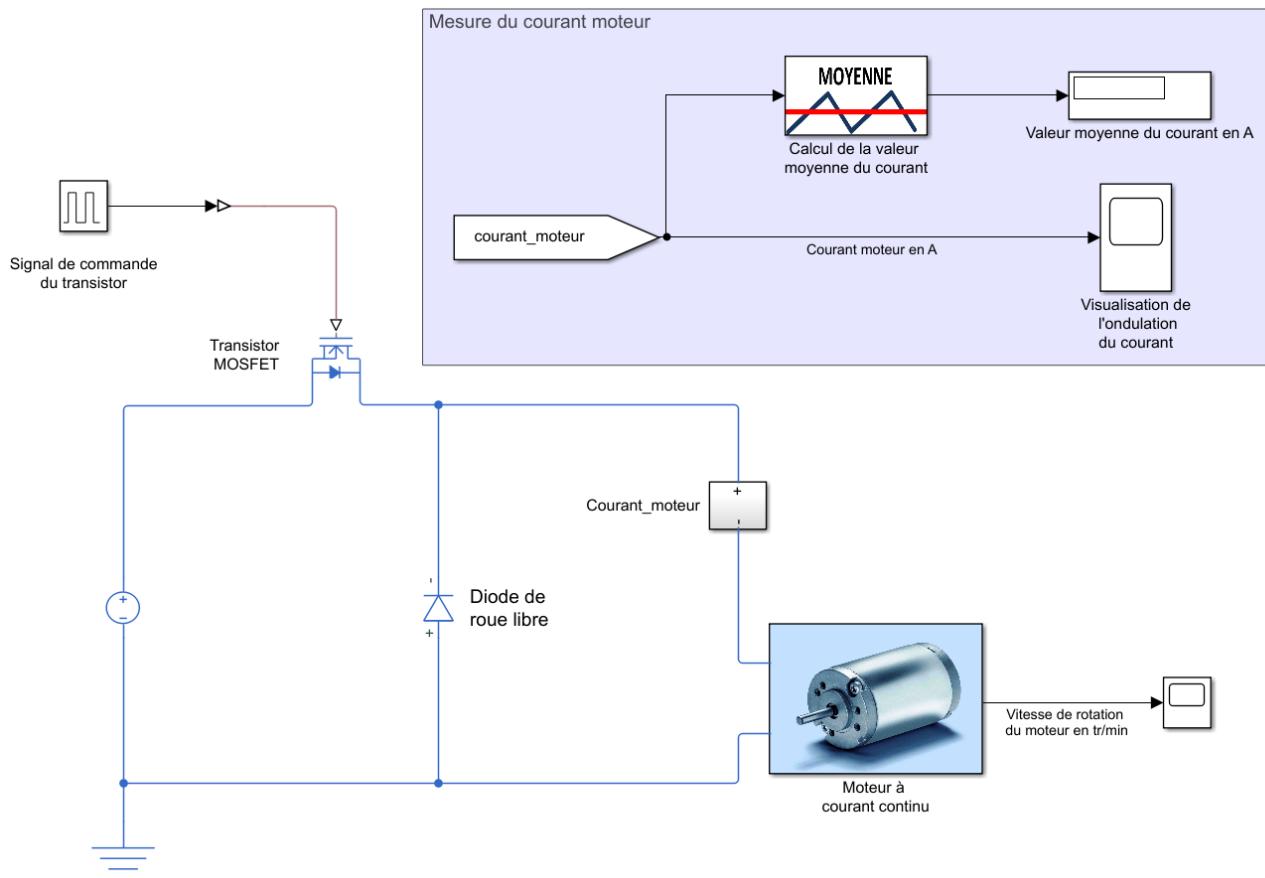


Figure 353 : le modèle didactisé et optimisé

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**.

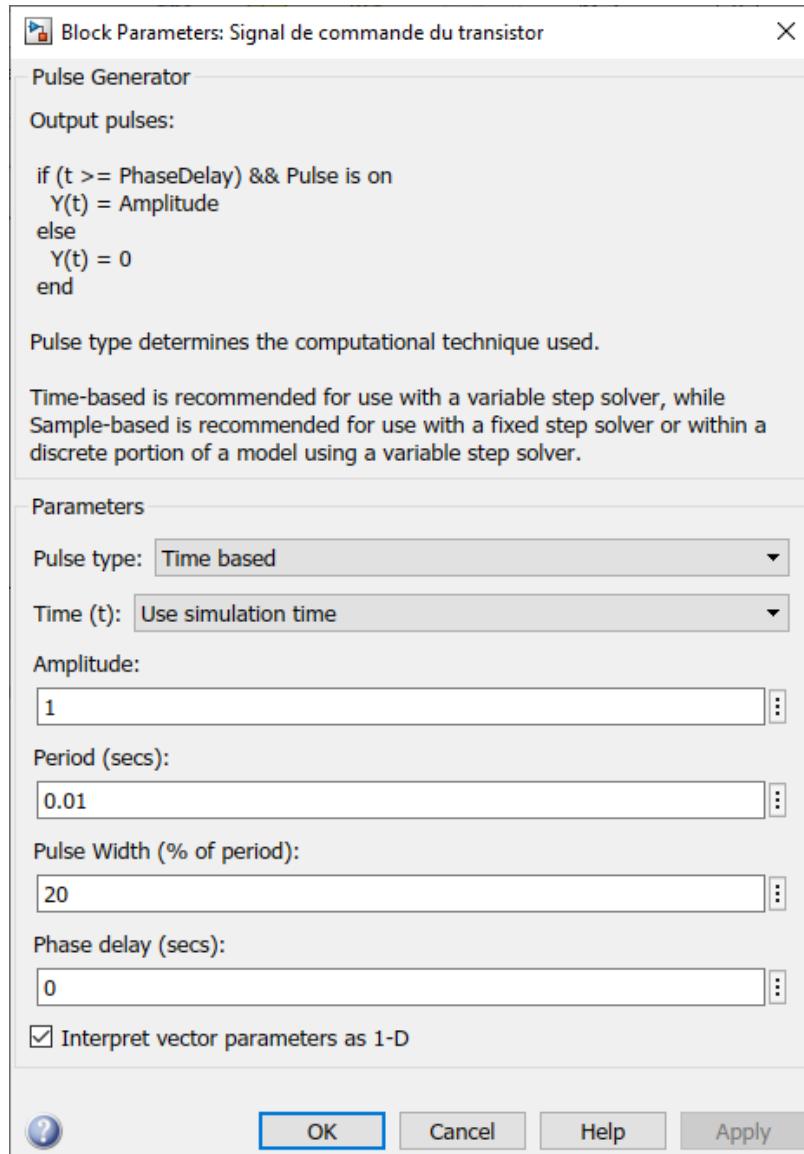
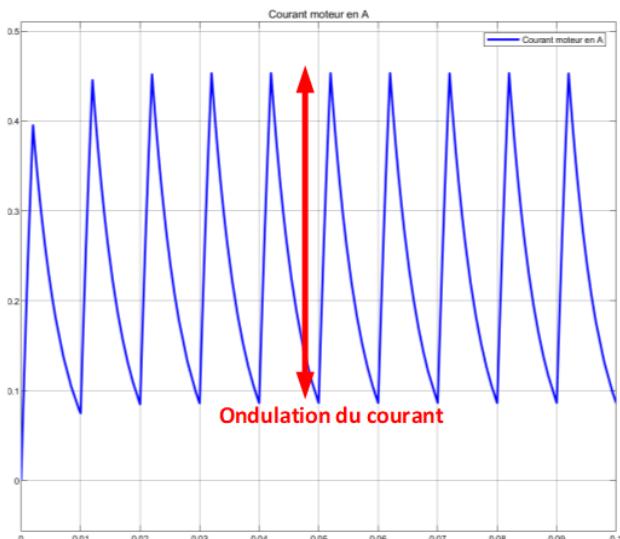
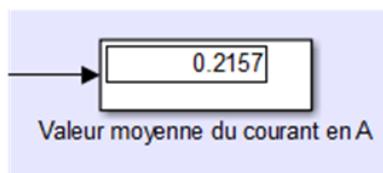


Figure 354 : réglage de la période de commutation du transistor

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 355).



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



Valeur moyenne du courant moteur

Figure 355 : 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 356).

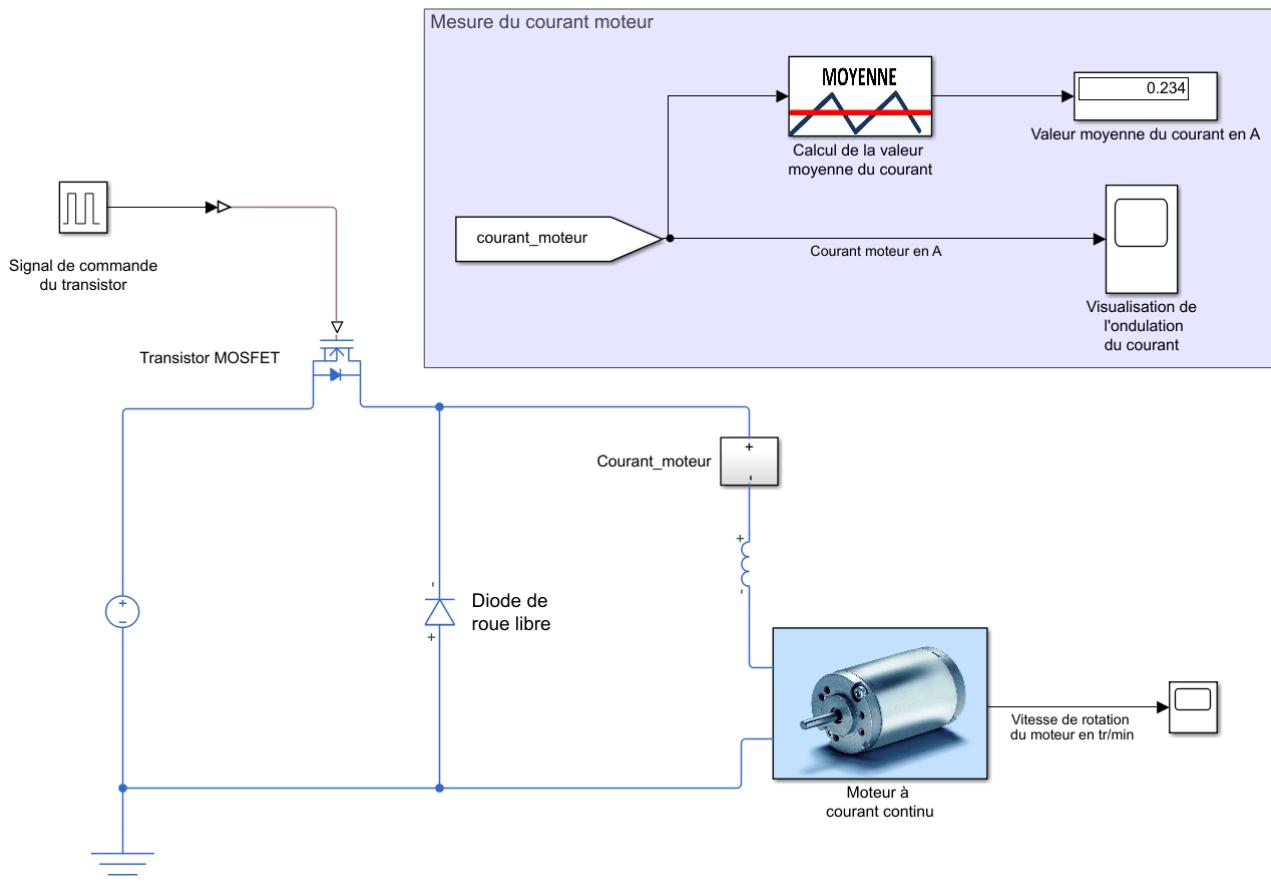


Figure 356 : ajout d'une inductance de lissage en série avec le moteur

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 357) 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

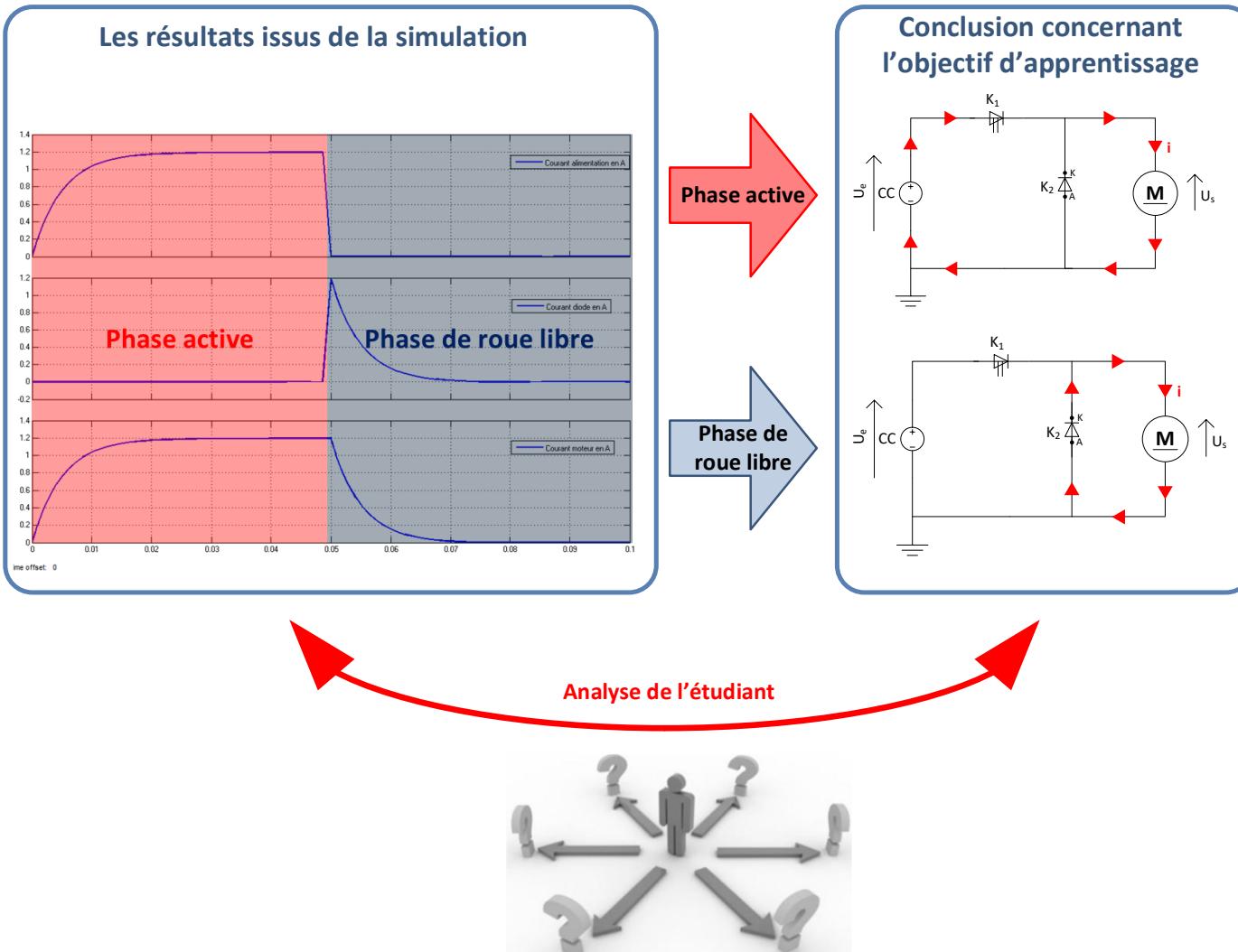


Figure 357 : exploitation et visualisation de la circulation du courant dans le hacheur série

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

La simulation du modèle « **hacheur_serie_3.slx** » (Figure 359) 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 $\alpha = 50\%$ (Figure 358)

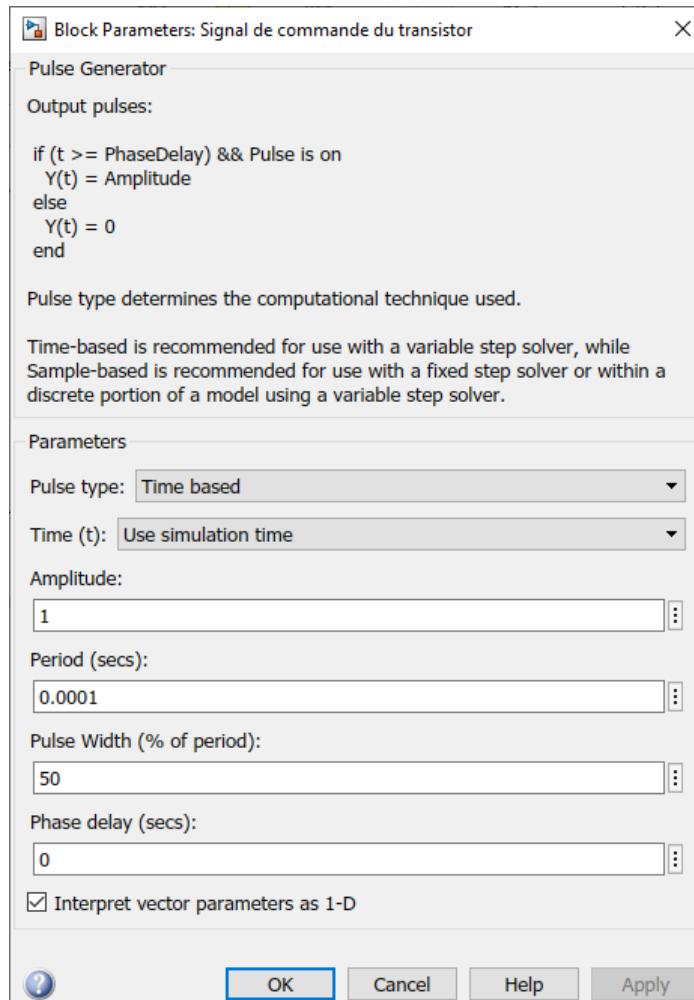
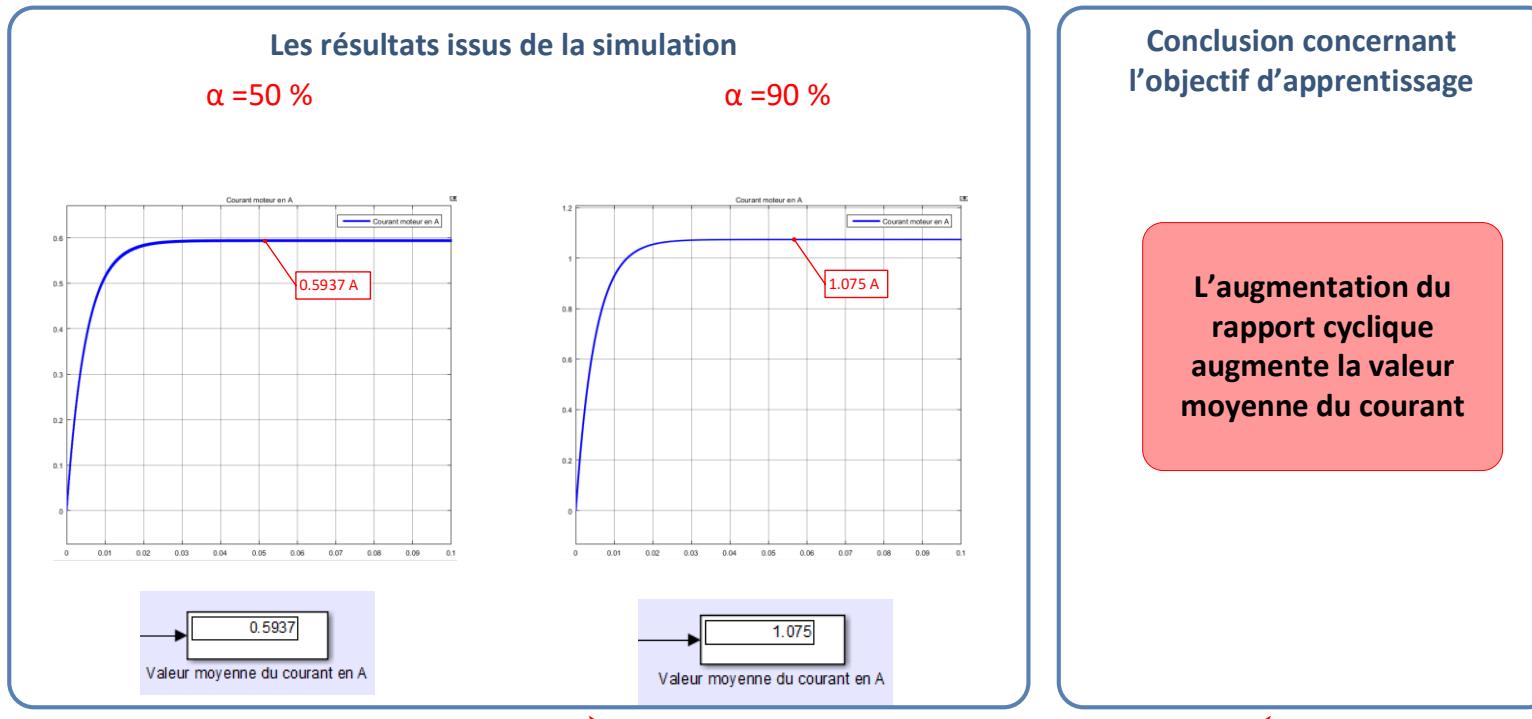


Figure 358 : réglages des paramètres de la simulation

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 $\alpha = 90\%$.

Influence du rapport cyclique



→ Analyse de l'étudiant



Figure 359 : 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 361) 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 **$\alpha = 50\%$**

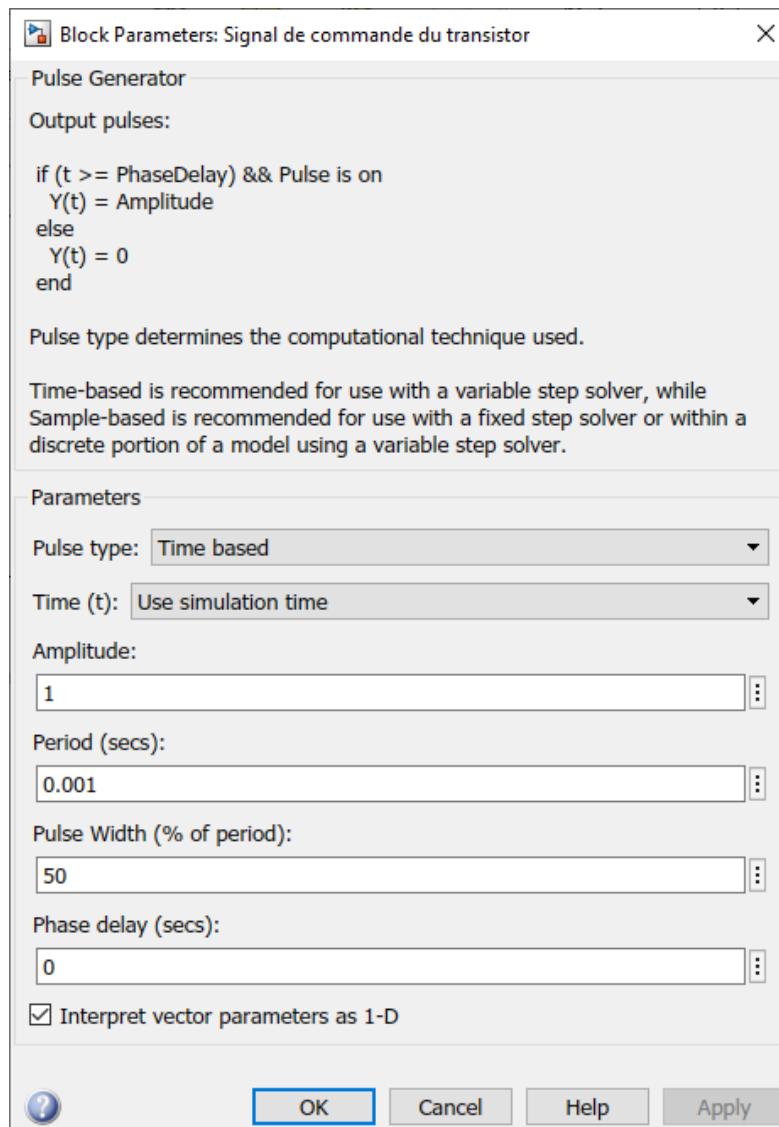
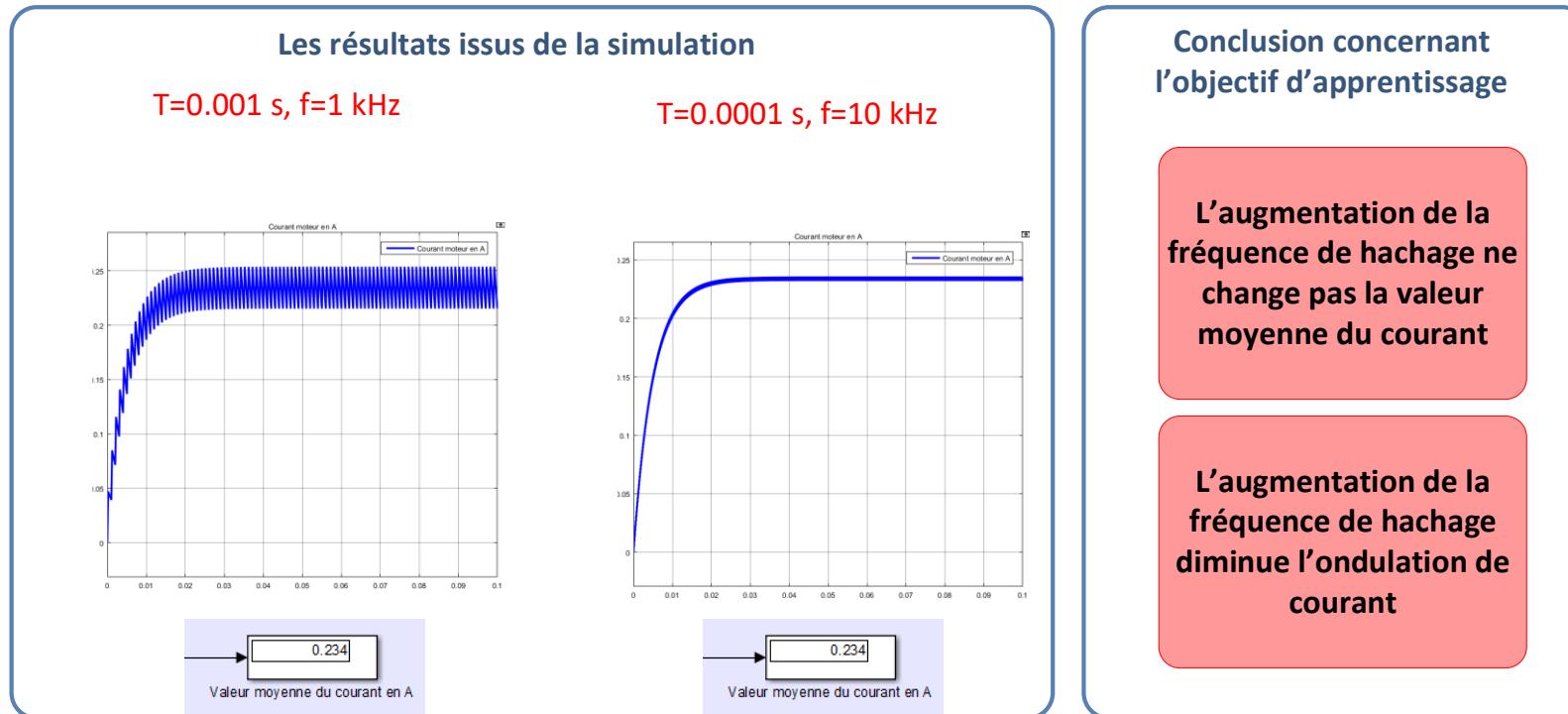


Figure 360 : réglage des paramètres de la simulation

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



Analyse de l'étudiant



Figure 361 : 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 364) permet de mettre en évidence l'influence de l'inductance de la charge.

Paramétrier le bloc **Pulse Generator** conformément à la Figure 362.

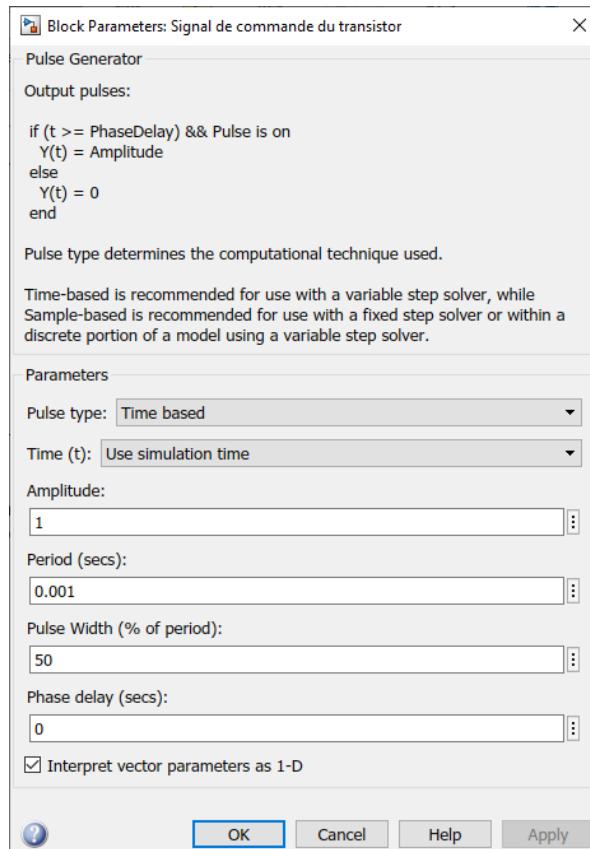


Figure 362 : paramétrage du bloc Pulse Generator

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

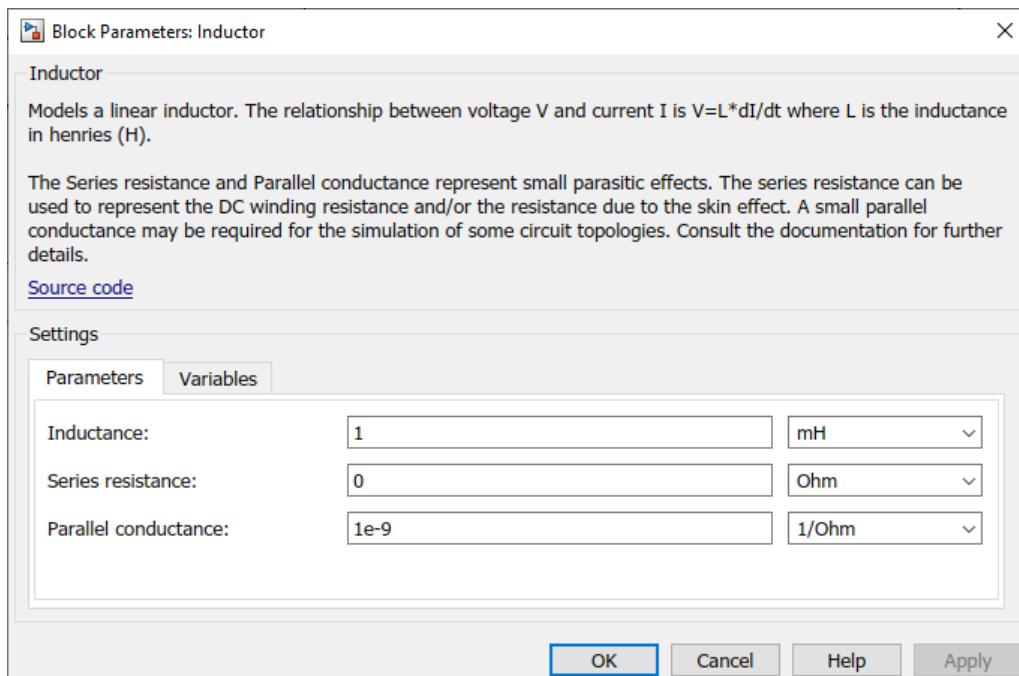
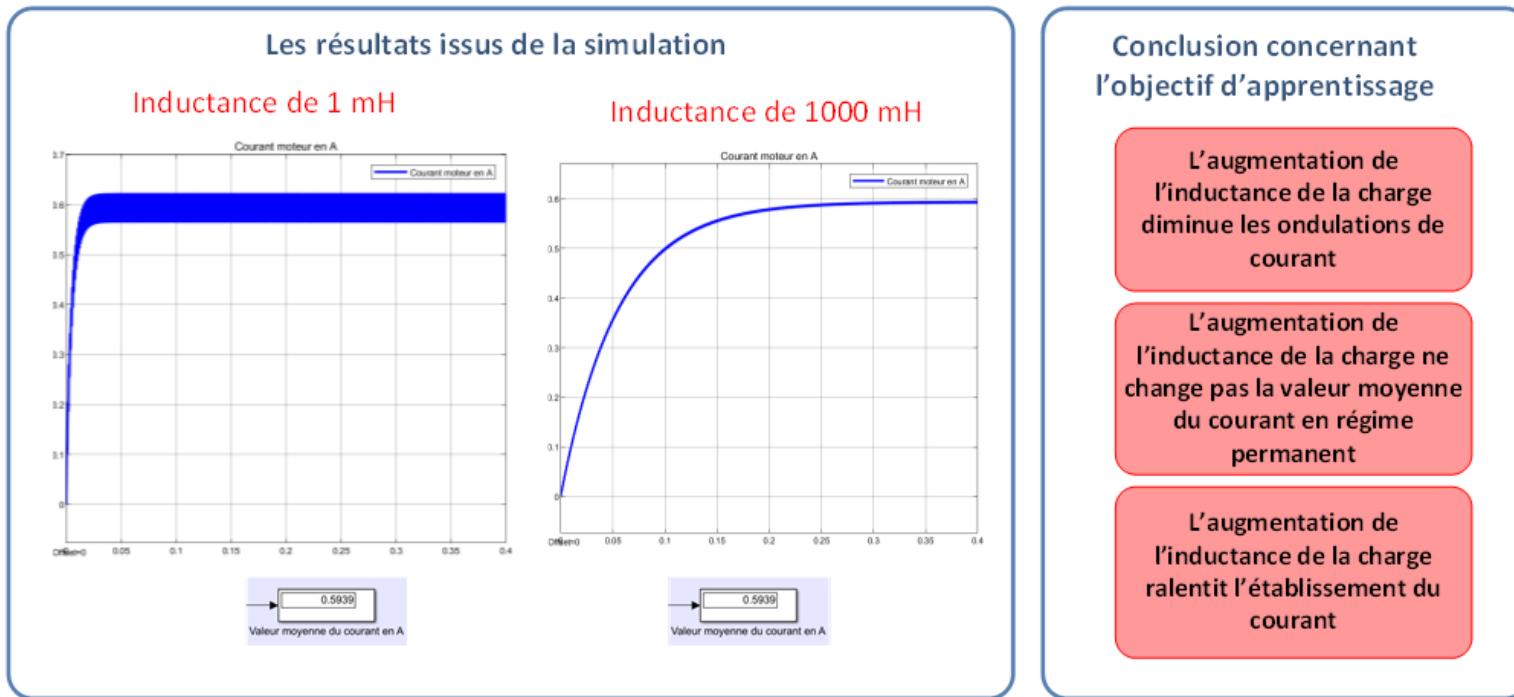


Figure 363 : réglage des paramètres de la simulation

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 **1000 mH**

Influence de l'inductance de la charge



Analyse de l'étudiant



Figure 364 : 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 5 : 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 365.

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

Figure 365 : 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 *linspace* 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.

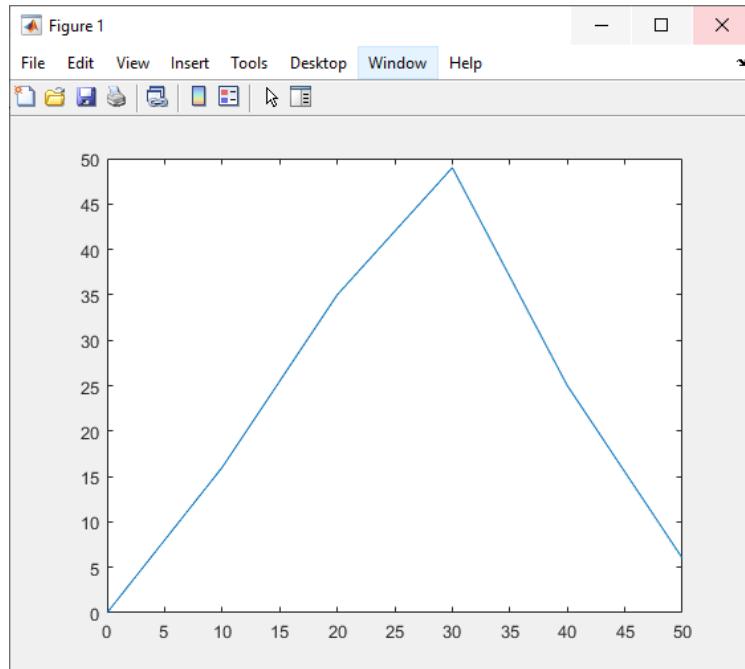


Figure 366 : tracé de deux vecteurs

On peut tracer le vecteur y_2 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 y_2 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*.

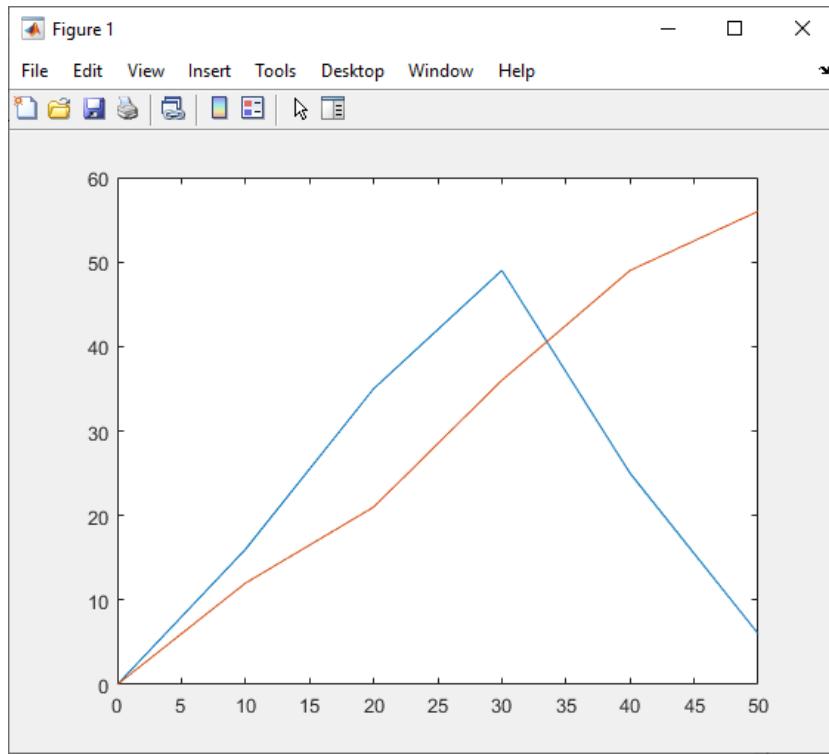


Figure 367 : tracé de deux courbes dans la même fenêtre graphique

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

```
>>grid on
```

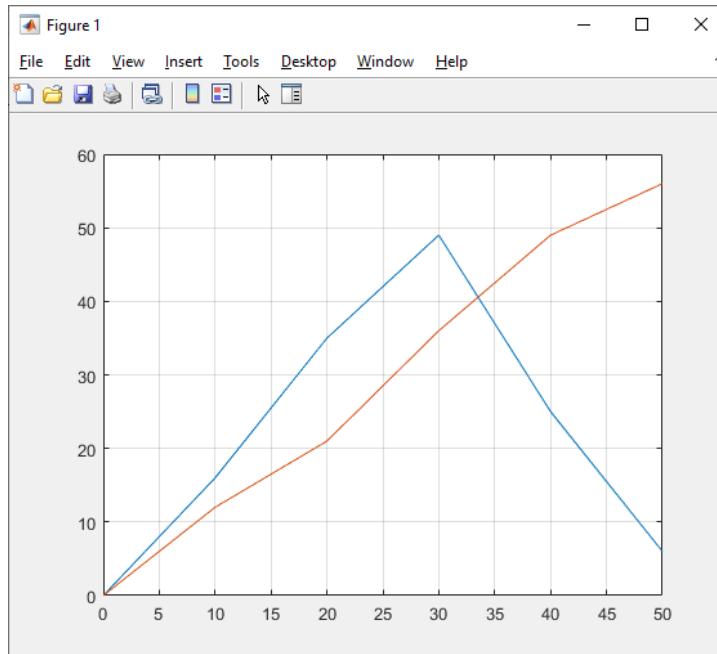


Figure 368 : affichage de la grille sur un graphique

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 369).

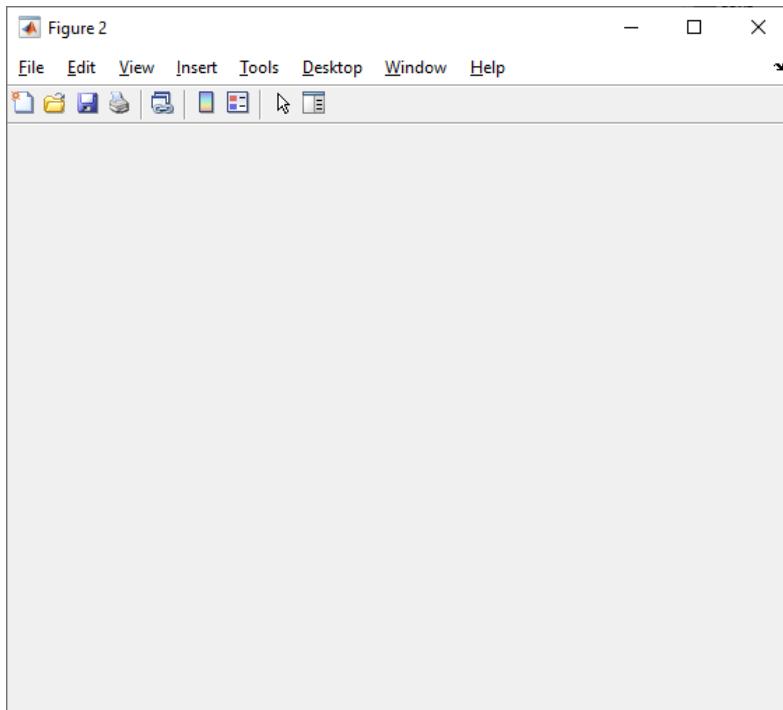


Figure 369 : ouverture d'une nouvelle fenêtre graphique

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 370 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	Point	- Continu
g	Cercle	-- Trait
r	Croix diagonale	: Pointillés
c	Croix verticale	-. Trait-point
m	Etoile	
y	Carré	
k	Losange	
w	Triangle vers le bas	
	Triangle vers le haut	
	Triangle à gauche	
	Triangle à droite	
	Pentagramme	
	hexagramme	

Figure 370 : 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

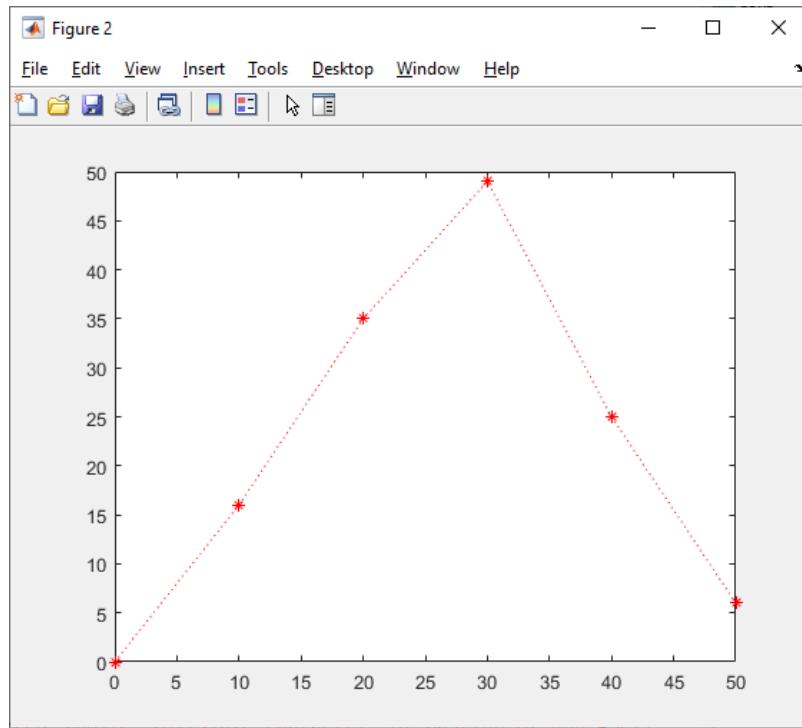


Figure 371 : mise en forme d'une courbe : modification de la couleur et du style de ligne

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

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

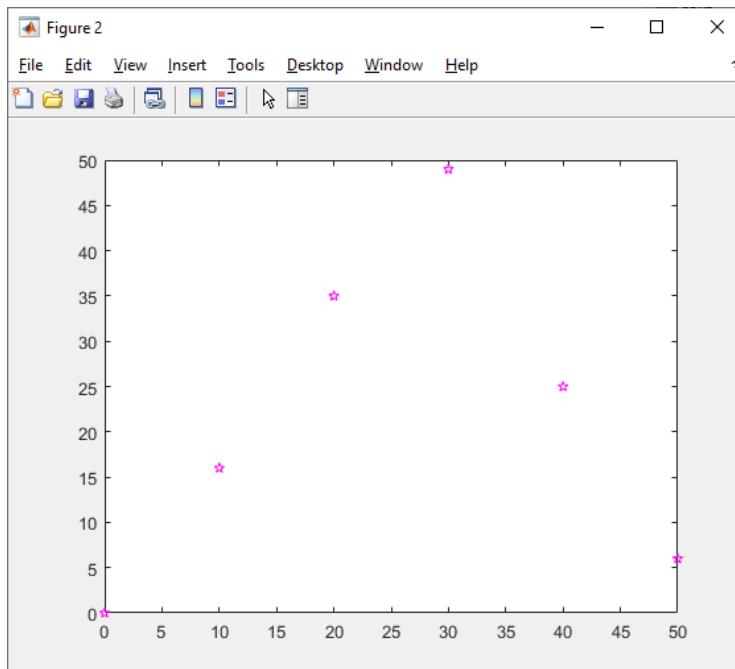


Figure 372 : mise en forme d'une courbe : ajout de marqueurs

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

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

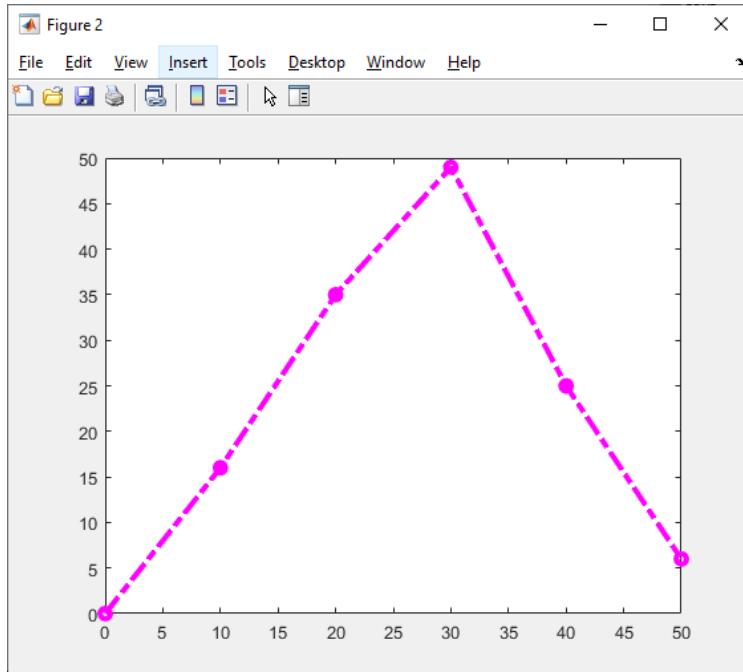


Figure 373 : mise en forme d'une courbe : choix de l'épaisseur de trait

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
```

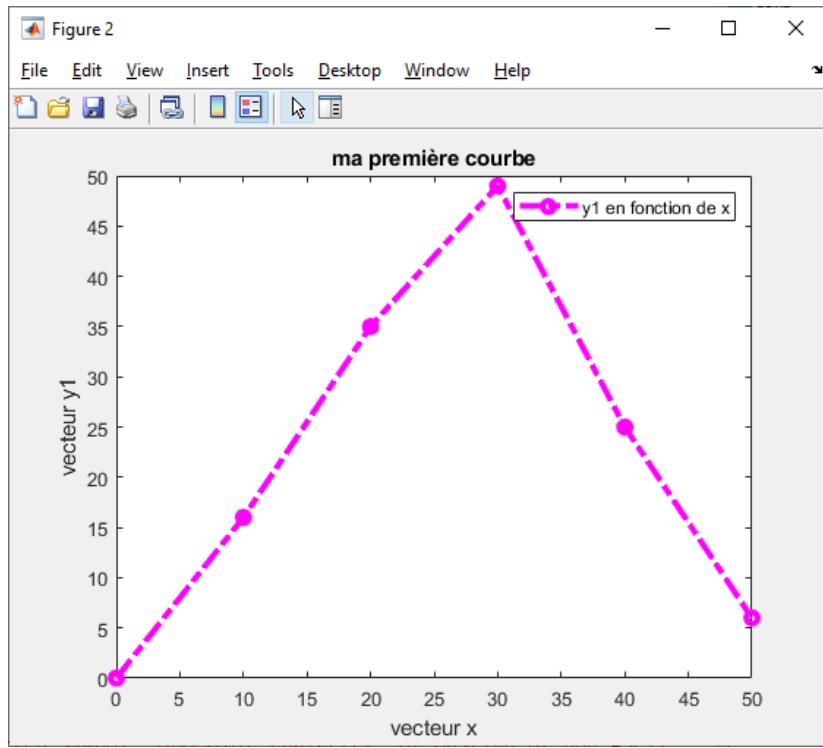


Figure 374 : mise en forme d'une courbe : légende et annotation des axes

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

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

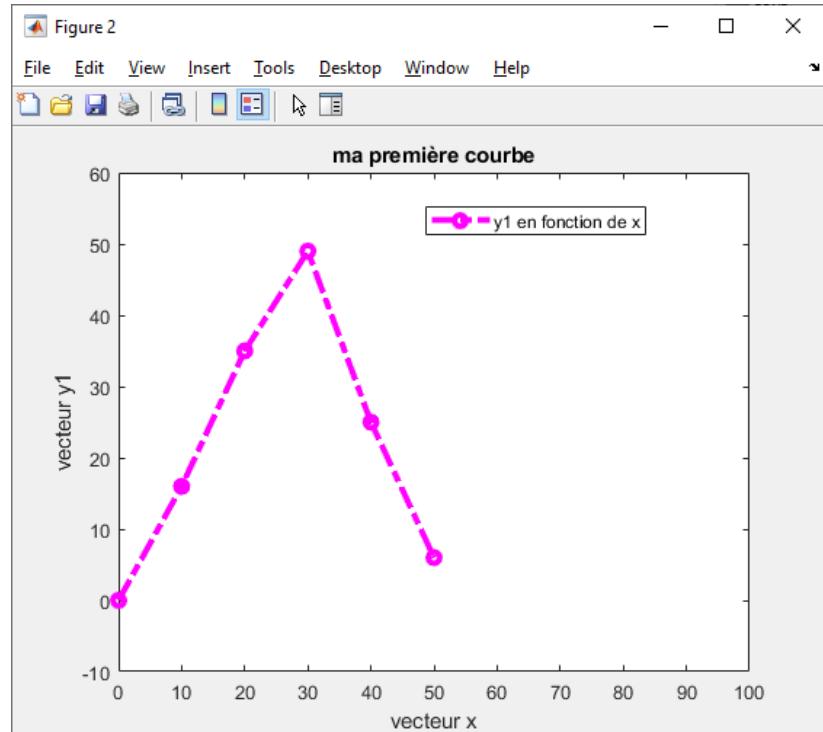


Figure 375 : mise en forme d'une courbe : modification des échelles des axes

G. Création de matrices

La manipulation des données peut également se faire en stockant des nombres dans des matrices.

Pour créer la variable M contenant la matrice $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$:

```
>> M = [1 2 3 ; 4 5 6 ; 7 8 9]
```

M =

```
1 2 3  
4 5 6  
7 8 9
```

Pour accéder à un élément de la matrice on utilise le système d'indexation $M(n_{\text{ligne}}, n_{\text{colonne}})$. Les lignes et les colonnes sont numérotées de 1 à n (Figure 376).

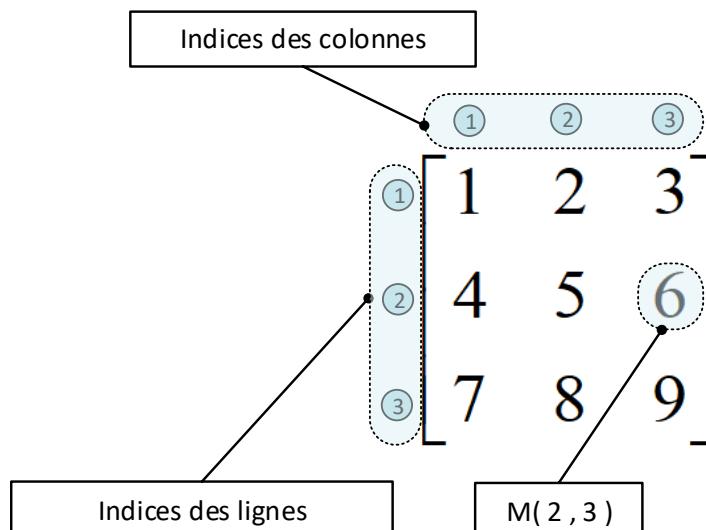


Figure 376 : indexation des lignes et des colonnes d'une matrice

```
>> M(2,3)  
  
ans =  
  
6  
  
>> M(1,3)  
  
ans =  
  
3
```

Il est également possible d'accéder à un élément d'une matrice en utilisant une indexation à un seul indice. Les éléments sont numérotés en partant du premier élément de la première ligne qui porte l'indice 1. Les indices sont ensuite incrémentés en se déplaçant du haut vers le bas dans la colonne. Arrivé en bas

de colonne, l'indexation se poursuit en partant du premier élément en haut de la colonne suivante, et ainsi de suite (Figure 377).

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Figure 377 : indexation des éléments d'une matrice avec un indice unique

```
>> M(1)
```

```
ans =
```

```
1
```

```
>> M(6)
```

```
ans =
```

```
8
```

```
>> M(2)
```

```
ans =
```

```
4
```

Il est ainsi possible d'affecter une valeur à un élément de la matrice de plusieurs manières.

```
>> M(2,3)=666
```

```
M =
```

```
1 2 3
4 5 666
7 8 9
```

```
>> M(4)=777
```

```
M =
```

```
1 777 3
4 5 666
7 8 9
```

Si on affecte une valeur à un élément dont l'indice est supérieur à la taille de la matrice, MATLAB augmente automatiquement la matrice afin de lui donner la bonne dimension en ajoutant des 0.

```
>> M(5,5)=888
```

```
M =
```

```
1 777 3 0 0  
4 5 666 0 0  
7 8 9 0 0  
0 0 0 0 0  
0 0 0 0 888
```

Cette fonctionnalité s'applique également aux vecteurs. Ici le vecteur T n'est pas défini dans le Workspace

```
>> T(6) = 0
```

```
T =
```

```
0 0 0 0 0 12
```

Il est possible d'extraire des parties de la matrice en utilisant la notation suivante : $A = M(2:4, 1:3)$. Cela signifie que la matrice A sera constituée des lignes 2 à 4 et des colonnes 1 à 3 de la matrice M (Figure 378).

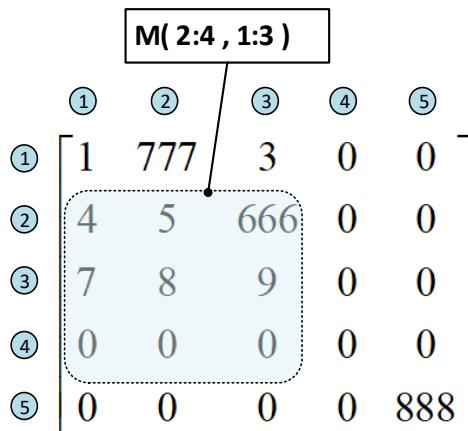


Figure 378 : extraction d'une matrice

```
>> A = M(2:4,1:3)
```

```
A =
```

```
4 5 666  
7 8 9  
0 0 0
```

Il est également possible d'extraire simplement des lignes ou des colonnes entières en utilisant la notation « : » qui signifie « **tous les indices de la ligne ou de la colonne** » (Figure 379).

$M(:, 3)$ permettra d'extraire tous les éléments de la troisième colonne.

$M(2, :)$ permettra d'extraire tous les éléments de la deuxième ligne.

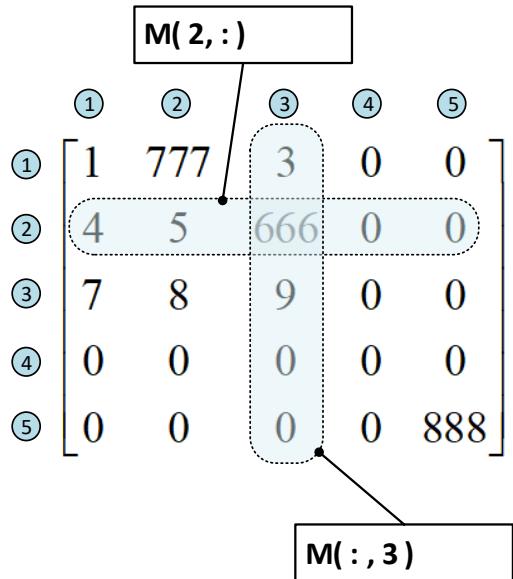


Figure 379 : extraction d'une ligne entière ou d'une colonne entière

```
>> M(:,3)
ans =
3
666
9
0
0

>> M(2,:)
ans =
4 5 666 0 0
```

Pour obtenir la transposée d'une matrice on utilise le signe « ' »

```
>> M'
ans =
1 4 7 0 0
777 5 8 0 0
3 666 9 0 0
0 0 0 0 0
0 0 0 0 888

>> C = M(:,2)
C =
777
```

```

5
8
0
0

>> C'

ans =

 777   5   8   0   0

```

Pour créer des matrices des 0, on utilise la fonction **zeros**.

```

>> B = zeros(4,4)

B =

 0   0   0   0
 0   0   0   0
 0   0   0   0
 0   0   0   0

```

Pour créer des matrices de 1, on utilise la fonction **ones**.

```

>> D = ones (4,4)

D =

 1   1   1   1
 1   1   1   1
 1   1   1   1
 1   1   1   1

```

Pour créer une matrice identité on utilise la fonction **eye**.

```

>> E = eye(5)

E =

 1   0   0   0   0
 0   1   0   0   0
 0   0   1   0   0
 0   0   0   1   0
 0   0   0   0   1

```

Pour créer une matrice avec des nombres choisis de manière aléatoire on utilise la fonction **rand**.

```

>> rand(3,3)

ans =

 0.8147   0.9134   0.2785
 0.9058   0.6324   0.5469
 0.1270   0.0975   0.9575

```

La fonction **randi** génère des éléments entiers choisis dans un intervalle spécifié.
`randi(10,3,3)` permettra de générer une matrice 3x3 constitué d'entier compris entre 0 et 10.

```
>> randi(10,3,3)
```

ans =

9	4	4
8	9	10
7	6	9

H. Opération éléments par éléments

MATLAB permet d'effectuer simplement les opération éléments par éléments, cela s'avère très utile.
Si nous souhaitons créer un vecteur contenant les valeurs de la fonction $f(t) = 3\sin(t) + \cos^2(t)$ pour $t \in [0 ; 2\pi]$. Nous devons dans un premier temps créer le vecteur t :

```
>> t = [0:0.01:2*pi];
```

t est donc un vecteur contenant les éléments espacées de 0.01 compris entre 0 et pi.

Nous devons maintenant créer le vecteur f défini par la fonction f(t). Taper la commande suivante puis validez :

```
>> f = 3*sin(t) + cos(t)^2
```

MATLAB va renvoyer un message d'erreur dans la mesure ou la multiplication et la fonction puissance ne sont pas définies comme une opération termes à termes.

Error using ^ (line 51)
Incorrect dimensions for raising a matrix to a power. Check that the
matrix is square and the power is a scalar. To perform elementwise matrix
powers, use '.^'.

La fenêtre de commande vous invite à utiliser les opérateurs « .* » ou « .^ » pour indiquer que vous voulez réaliser ces opérations termes à termes. Tapez alors la commande suivante :

```
>> f = 3.*sin(t) + cos(t).^2;
```

Le vecteur f a maintenant été créé, et il est possible de tracer la fonction f(t) en utilisant la commande plot (Figure 380):

```
>> plot(t,f)
```

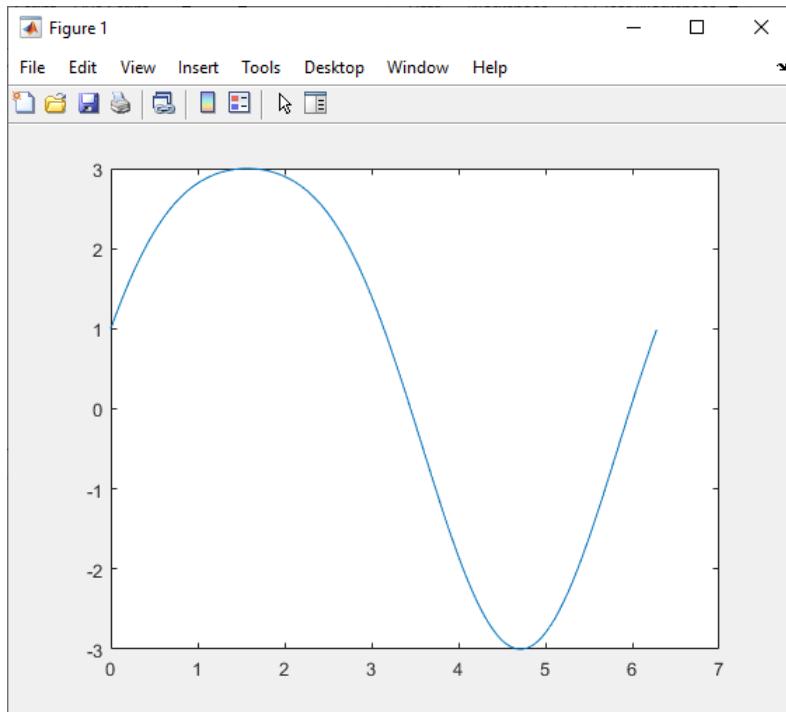


Figure 380 : tracé de la fonction $f(t)$

I. 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 381).

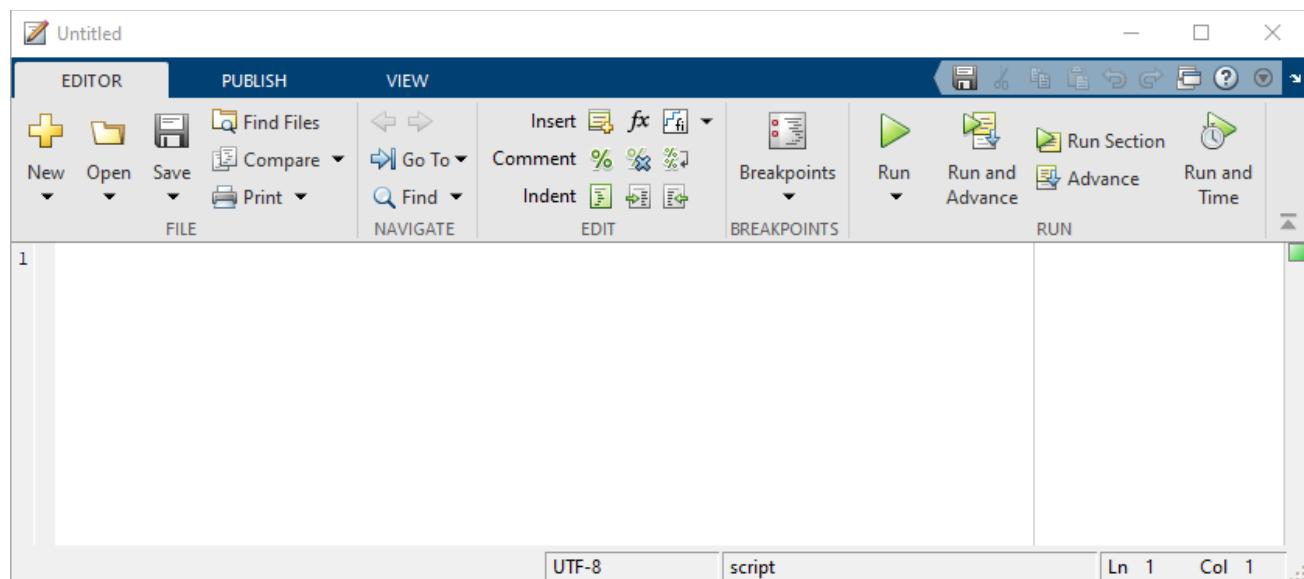
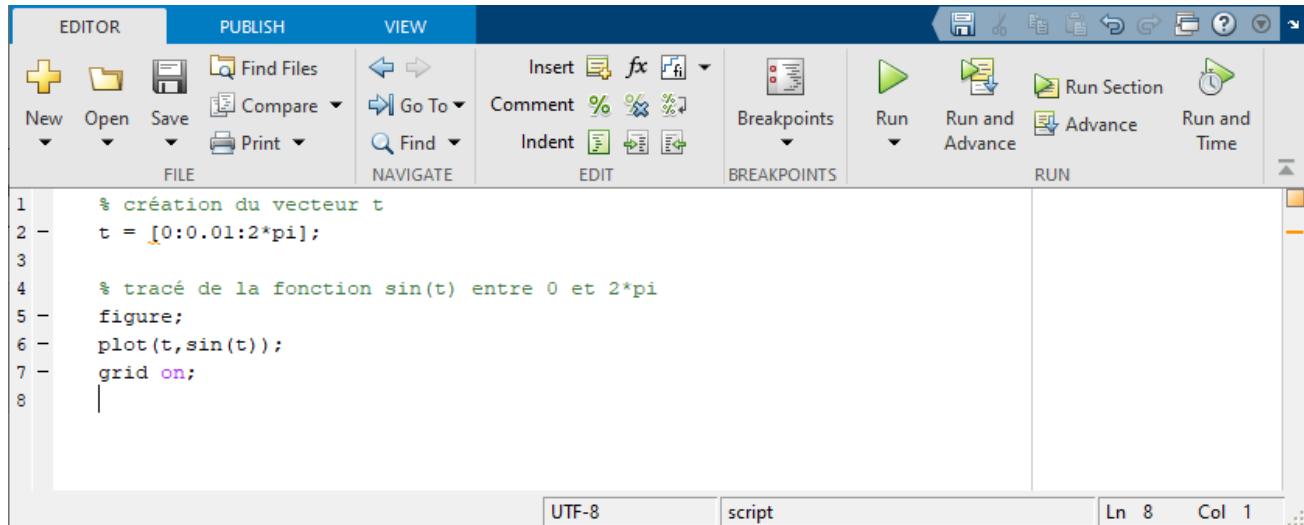


Figure 381 : fenêtre Editor d'édition des scripts

Saisir les lignes de commande de la Figure 382 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.



```
1 % création du vecteur t
2 t = [0:0.01:2*pi];
3
4 % tracé de la fonction sin(t) entre 0 et 2*pi
5 figure;
6 plot(t,sin(t));
7 grid on;
```

The screenshot shows the MATLAB Editor window. The menu bar includes 'EDITOR', 'PUBLISH', and 'VIEW'. The toolbar has icons for New, Open, Save, Print, Find Files, Go To, Comment, Breakpoints, Run, Run Section, Run and Advance, Run and Time, and other editing tools. The main workspace shows the script code. The status bar at the bottom indicates 'UTF-8', 'script', 'Ln 8 Col 1'.

Figure 382 : exemple de script

Exécuter le script en cliquant sur  dans la barre de commande.

Si la fenêtre de la Figure 383 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**

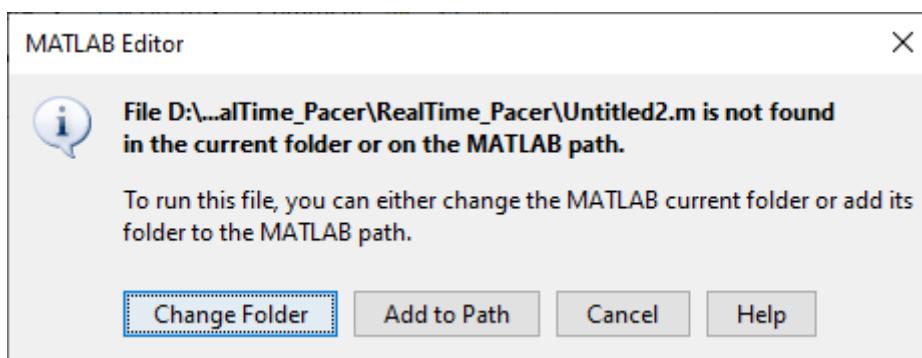
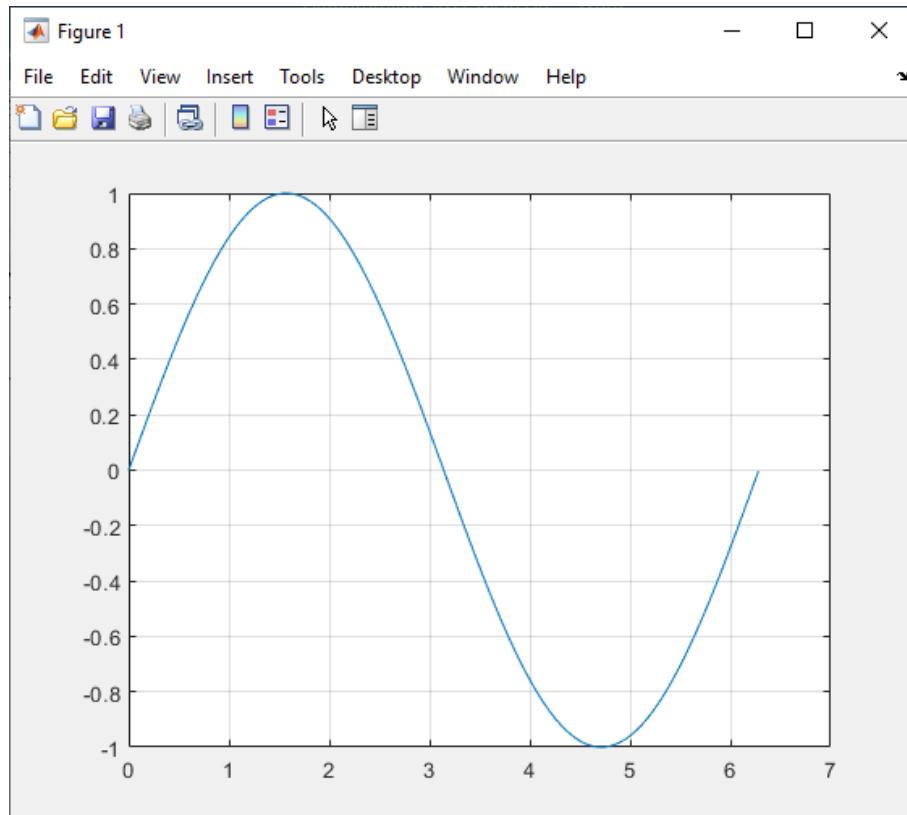


Figure 383 : fenêtre d'ajout automatique de dossier au « **path** » de MATLAB

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 384.

```

1 %création du vecteur t
2 t=[0:0.01:2*pi];
3 % la fonction hold all est identique à la fonction hold on mais permet
4 % de tracer chaque courbes avec une couleur différentes
5 hold all
6 % pour a prenant les valeurs comprises entre 1 et 4 avec un pas de 1
7
8 for a=[1:4]
9 %tracé de la fonction f(t)=sin(a*t) entre 0 et 2*pi
10 plot(t,sin(a*t),'LineWidth',2);
11 end
12 grid on
13 title('Fonction Sinus')
14 %MATLAB possède un interpréteur TeX de base, il est possible d'y faire
15 %figurer des caractères spéciaux ou des équations
16 xlabel('angle \theta en radians')
17 ylabel('f(\theta)=sin(\theta)')
18 axis([-1 7 -1.5 1.5])

```

Figure 384 : 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.

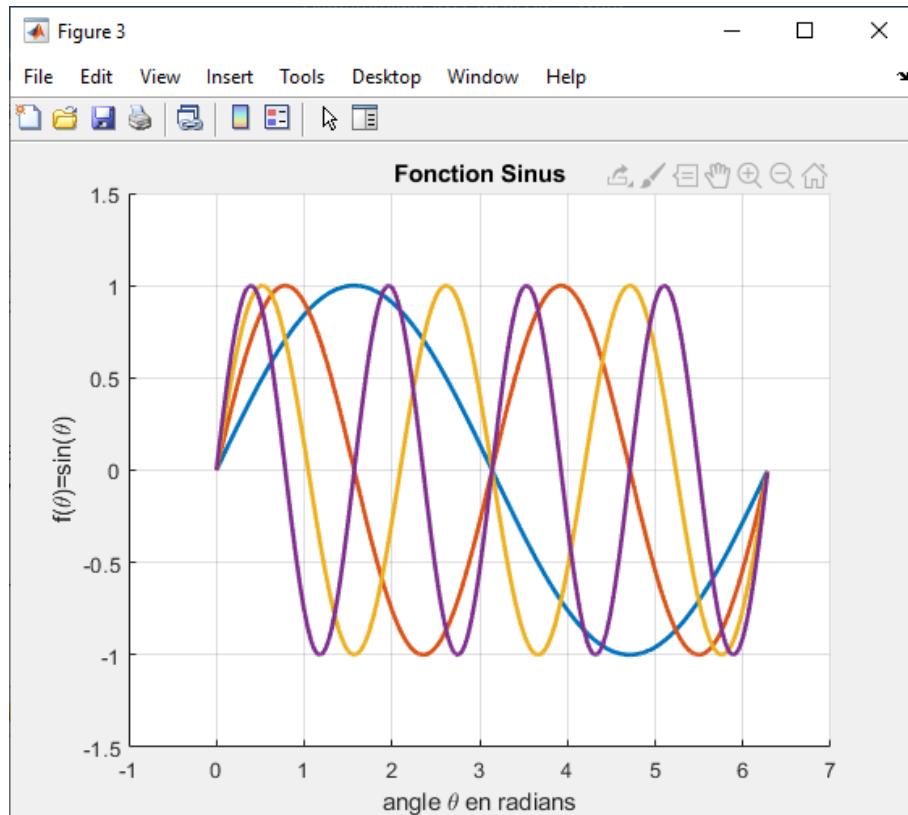


Figure 385 : résultats du script de tracé de plusieurs sinus

J. 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		$==$
Different de		\neq
Supérieur à		$>$
Supérieur ou égal à		\geq
Inférieur à		$<$
Inférieur ou égal à		\leq
Ou(or)		$ $
Et(and)		$\&$

Figure 386 : les opérateurs logiques et de comparaison de MATLAB

K. Les structures 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

Le chapitre « **Ingénierie numérique en langage MATLAB** » présente les algorithmes les plus utiles en calcul numérique. Quelques exemples sont donnés ici à titre indicatif.

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
```

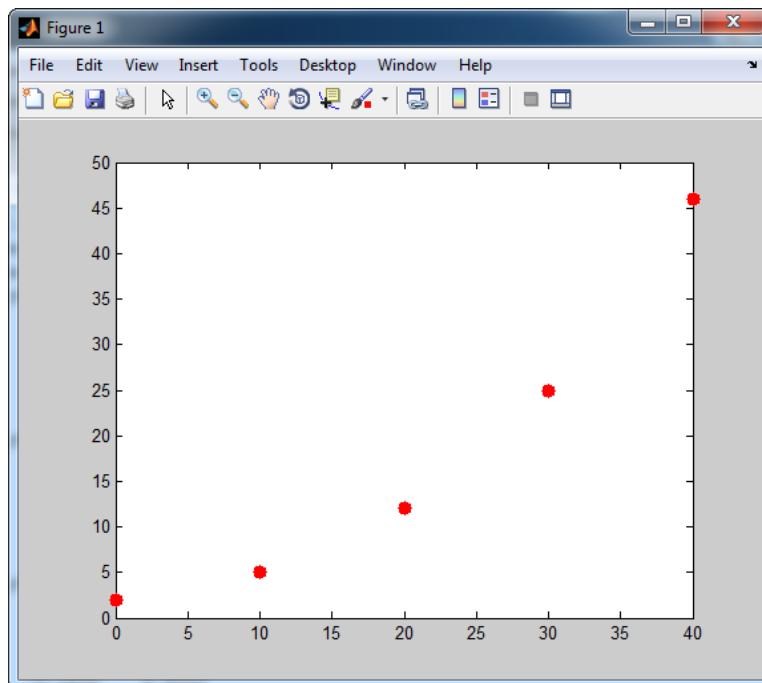


Figure 387 : repartition d'une série de points

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 $p(x) = 0.03x^2 - 0.12x + 2.4$.

Tester le script de la Figure 388 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.

```

% fermeture de toutes les fenêtres graphiques en cours
close all;

% création des vecteurs X et Y
X = [0 10 20 30 40];
Y = [2 5 12 25 46];

% création du vecteur A qui contient les coefficients du polynôme
% d'interpolation de degrés 2 de la série de données
A = polyfit(X,Y,2);

% Création du vecteur t contenant 100 composantes prises entre X(1) et
% X(end)
t = linspace(X(1),X(end),100);

% création du vecteur U qui contient 100 éléments correspondants aux
% valeurs du polynôme d'interpolation pour les 100 éléments du vecteur t
U = polyval(A,t);

% tracé du polynôme d'interpolation
plot(t,U,'g','LineWidth',3);
hold on;

% tracé des points Y en fonction de X, la commande MarkerSize permet de
% spécifier la taille des marqueurs
plot(X,Y,'r.','MarkerSize',25);
grid on;

% titre
title('Interpolation d''une série de données');

```

Figure 388 : script d'interpolation d'une série de données

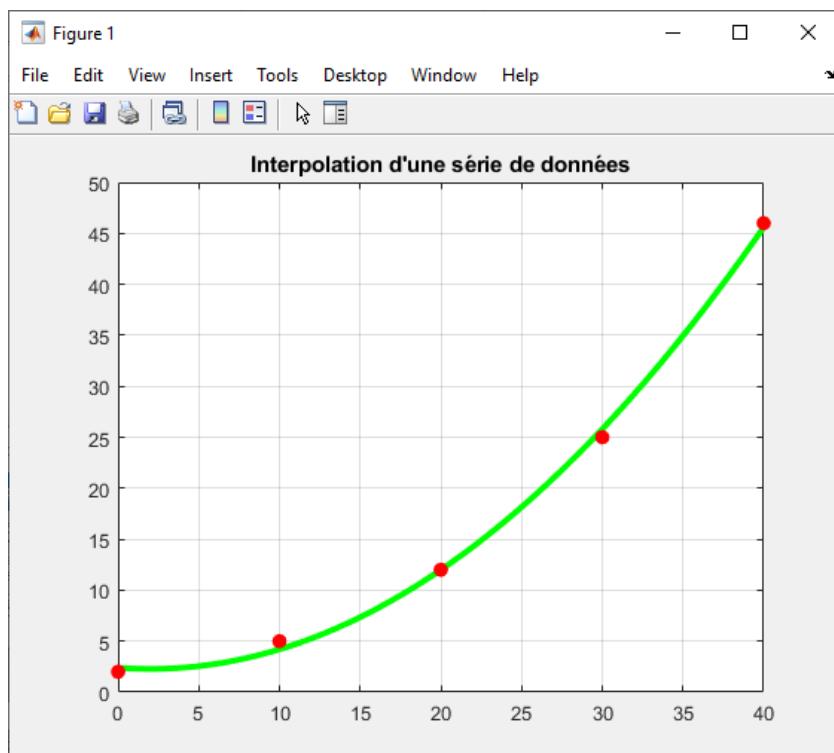


Figure 389 : graphique de l'interpolation d'une série de données

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** » :

```
%définition de x comme une variable symbolique
syms x
%% %résolution d'une équation algébrique

%définition de l'équation à résoudre
equation= x^2+2*x-3==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 390 : 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 équation algébrique (racines 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 391 : 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 doit 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 392 : 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 : $f(t) = \sin(t) + 3t^3$.

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** » :

```
%% dériver une fonction
%définition de t comme une variable symbolique
syms t
%définition de la fonction à dériver
f=sin(t)+3*t^3;

%calcul de la dérivée première
df=diff(f,t)

%calcul de la dérivée cinquième
d5f=diff(f,t,5)
```

Figure 393 : 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 : $f(t) = \sin(t) + 3t^3$.

Nous souhaitons calculer la primitive de $f(t)$ et l'intégrale suivante : $I = \int_{-\pi/3}^{\pi} f(t) dt$

Saisir le script suivant ou ouvrir le script « **integarer_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 394 : script permettant d'intégrer une fonction et de calculer une intégrale définie

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

```
>> integarer_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 = ∫ f(t) dt
                             -π
                             3

```

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 395 : 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(ωt)

ans =
s/(s^2 + w^2)           calcul de la transformée de Laplace de cos(ωt)

ans =
(a + s)/((a + s)^2 + w^2)   calcul de la transformée de Laplace de e^-at cos(ωt)

ans =
w/((a + s)^2 + w^2)       calcul de la transformée de Laplace de e^-at sin(ωt)

```

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** » :

```
%% Transformée inverse de Laplace d'une expression
syms t;
syms w;
syms a;

%% Transformée inverse de Laplace d'une expression
syms s
ilaplace(s/(s^2 + w^2))
ilaplace(3/(2+s)-5/(3+s)^2)
```

Figure 396 : 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  $\frac{s}{s^2 + \omega^2}$ 
ans =
3*exp(-2*t) - 5*t*exp(-3*t)           calcul de la transformée inverse de Laplace de  $\frac{3}{2+s} - \frac{5}{3s^2}$ 
```

7. Décomposition en éléments simples

La commande *residue* permet d'obtenir la décomposition en éléments simples d'une fraction rationnelle.

Considérons la fonction $H(s) = \frac{4s+5}{s^2+6s+8}$ 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 397 : 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'ajouteraient 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 :

$$5 \frac{d^2 f(t)}{dt^2} + 3 \frac{df(t)}{dt} + f(t) = 2 \quad \text{avec} \quad f(0)=0 \text{ et } \frac{df(0)}{dt}=0 \quad (\text{équation 1})$$

$$5 \frac{d^2 f(t)}{dt^2} + \frac{df(t)}{dt} + f(t) = 2 \quad \text{avec} \quad f(0)=0 \text{ et } \frac{df(0)}{dt}=0 \quad (\text{équation 2})$$

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 398 : 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 sol2 sont exprimées sous forme symbolique et sont tracées à l'aide de la commande *ezplot* (Figure 399 et Figure 400)

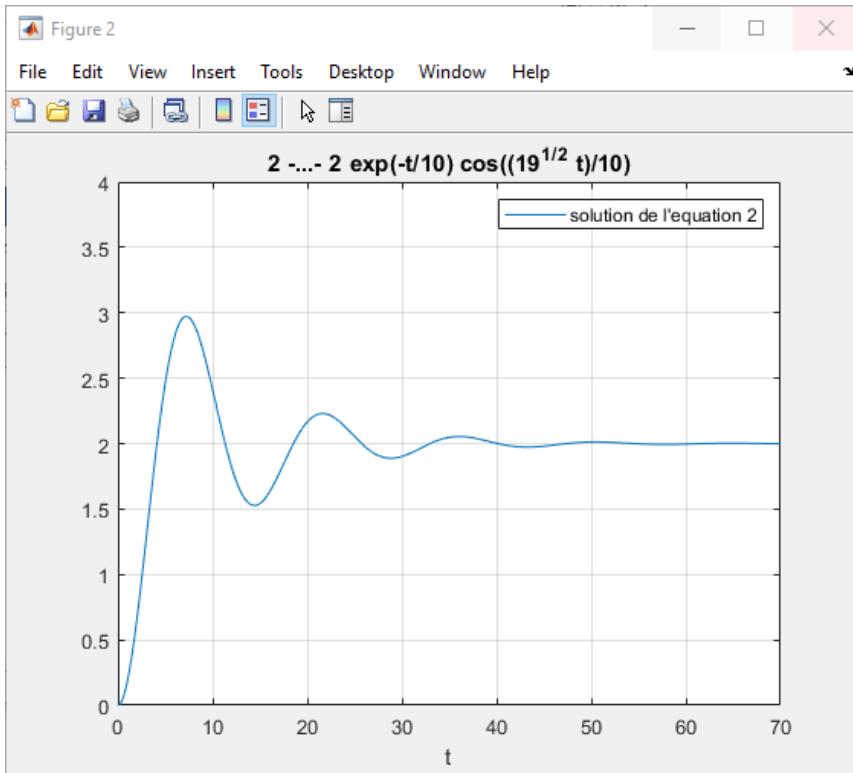


Figure 399 : représentation de la solution 1 de l'équation différentielle d'ordre 2

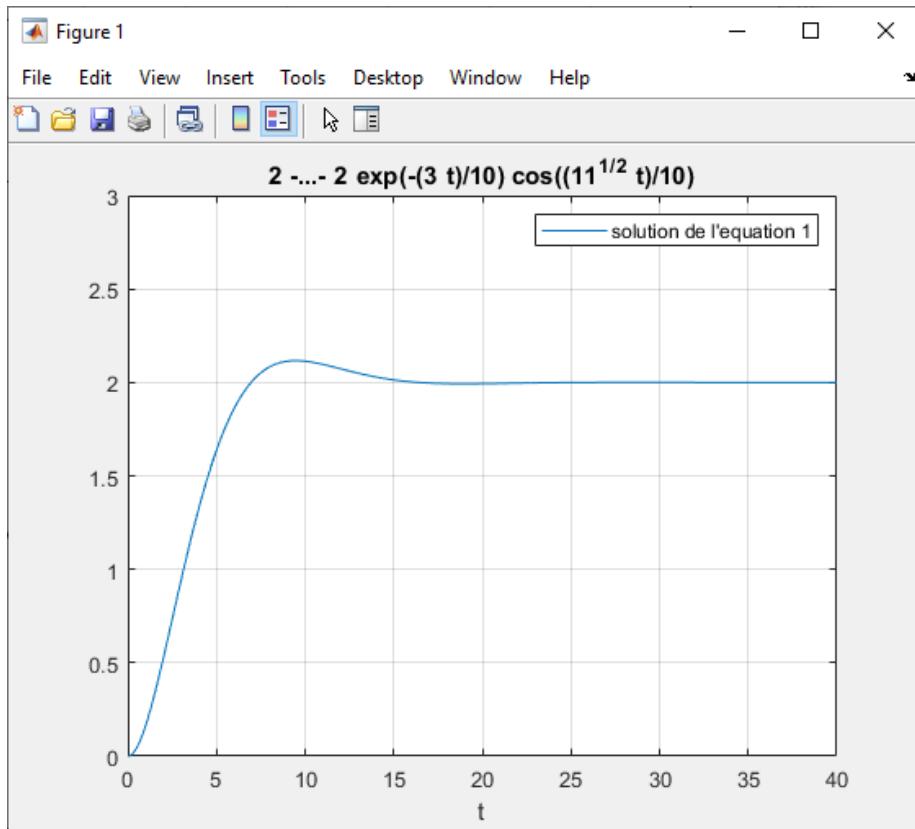


Figure 400 : représentation de la solution 2 de l'équation différentielle d'ordre 2

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 :

$$\begin{cases} H(p) = \frac{4s+1}{2s^2 + 3s + 6} \\ G(p) = \frac{100(s+1)(s+2)}{(s+0.1)(s+4)(s+10)} \end{cases}$$

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 `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`.

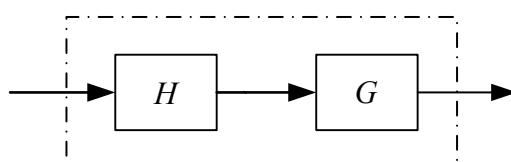


Figure 401 : fonctions de transfert en série

```
>> 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*

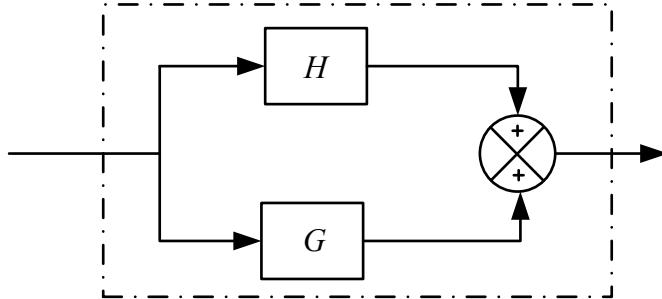


Figure 402 : fonction de transfert en parallèle

```
>> 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*.

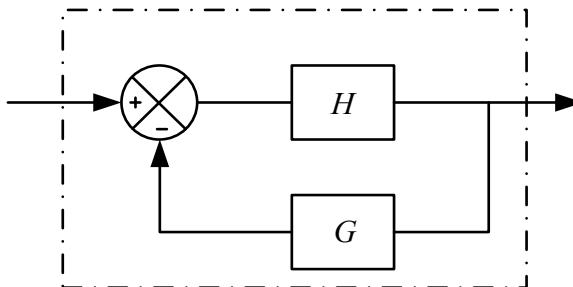


Figure 403 : fonction de transfert en boucle fermée

```
>> 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.

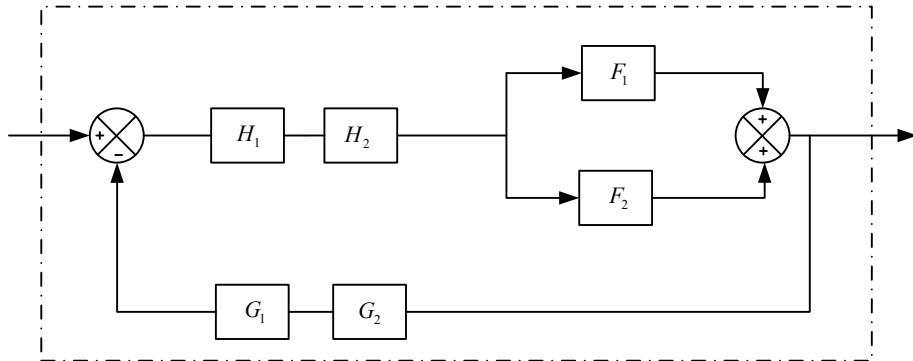


Figure 404 : fonction de transfert d'un système quelconque

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)`

`ans =`

$$\frac{1}{K_p + K_i \cdot \text{---} + K_d \cdot s}$$

with $K_p = 50$, $K_i = 0.1$, $K_d = 70$

Continuous-time PID controller in parallel form.

Les trois arguments représentent les coefficients K_p , K_i et K_d du correcteur : `pid(Kp,Ki,Kd)`

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`

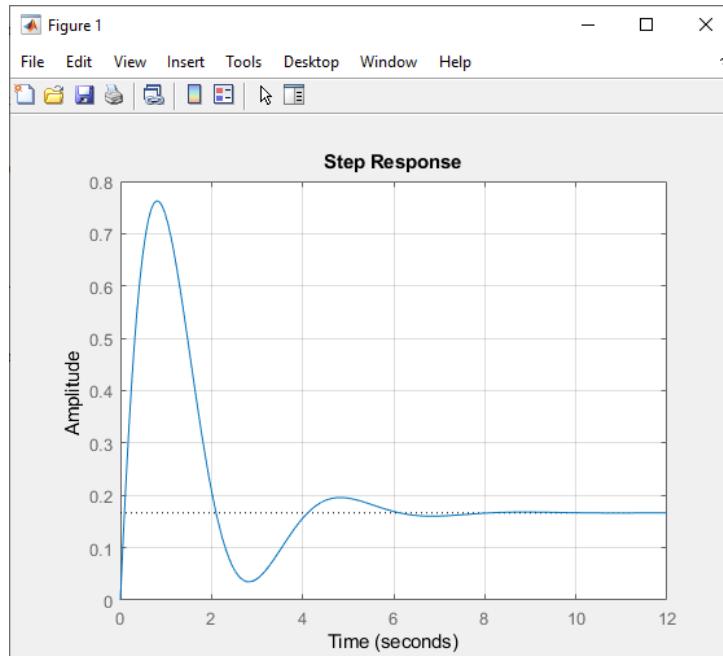


Figure 405 : réponse indicielle d'un système

Pour obtenir la réponse impulsionale d'un système, on peut utiliser la commande *impulse*
`>>impulse(H) ; grid on`

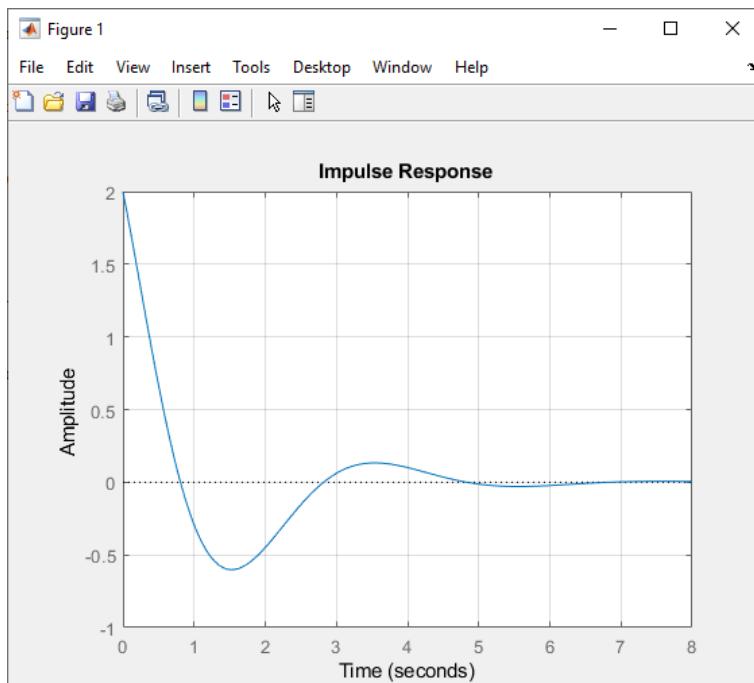


Figure 406 : réponse impulsionale d'un système

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`

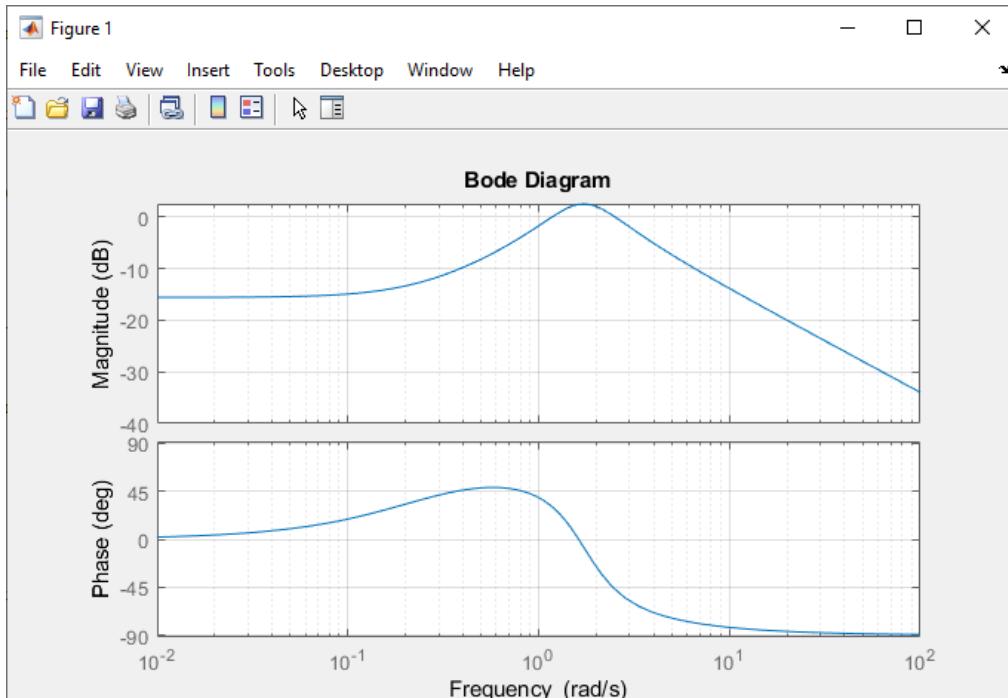


Figure 407 : diagramme de Bode d'un système

Pour obtenir le diagramme de Black-Nichols d'un système, on peut utiliser la commande *nichols*
`>>nichols(H) ; grid on`

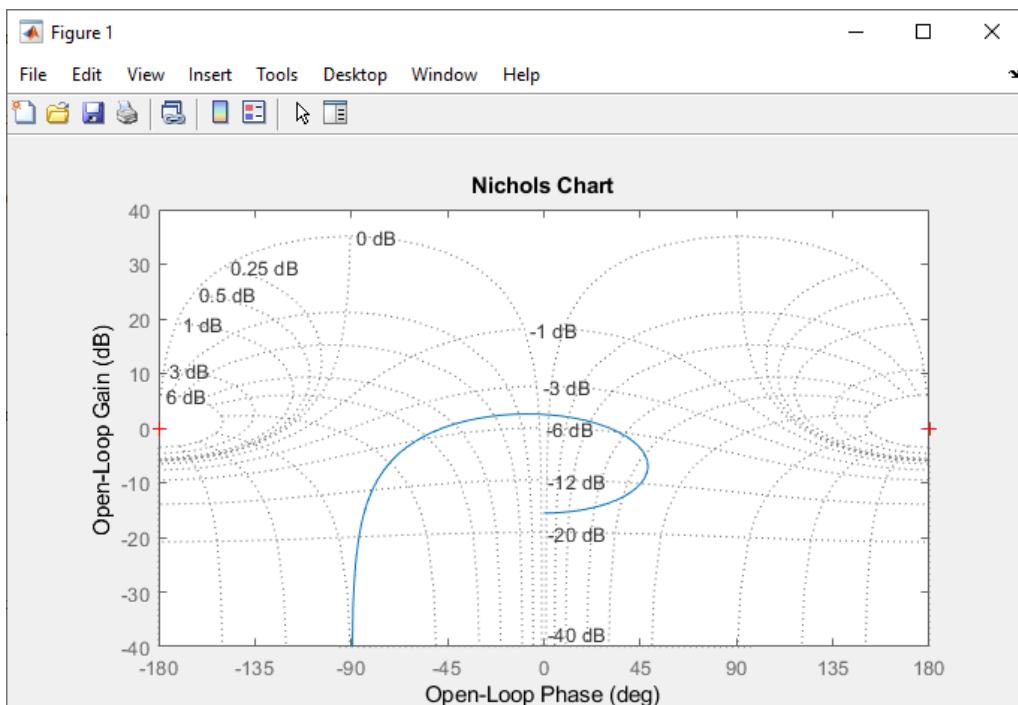


Figure 408 : diagramme de Black-Nichols d'un système

Pour obtenir le diagramme de Nyquist d'un système, on peut utiliser la commande *nyquist*
`>>nyquist(H) ; grid on`

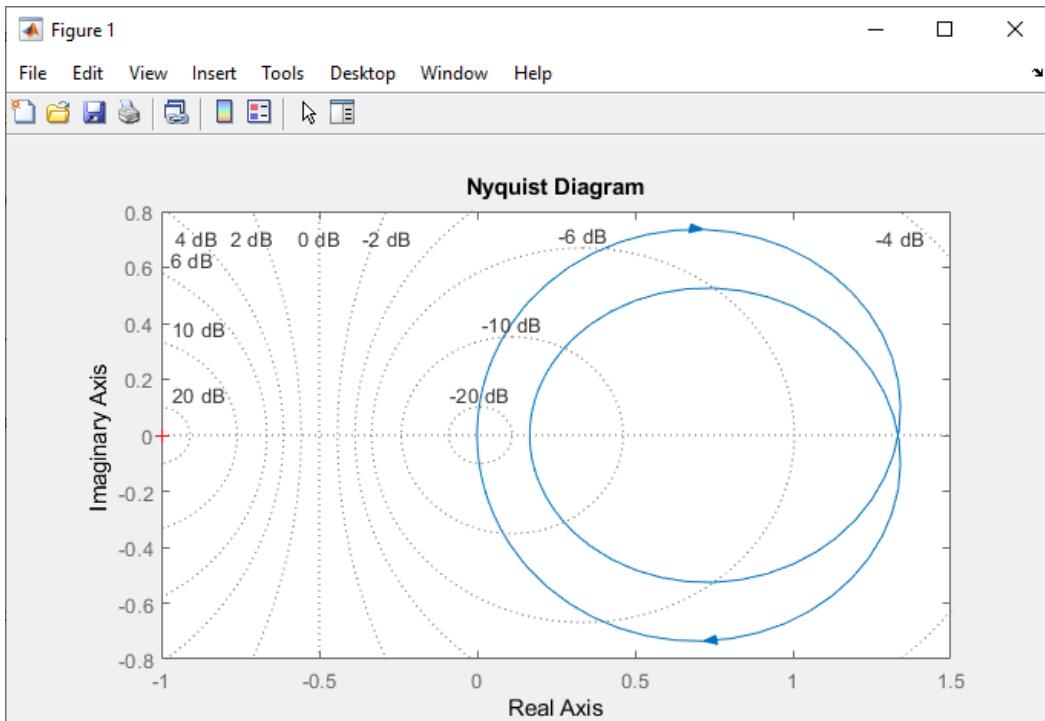


Figure 409 : diagramme de Nyquist d'un système

On peut également tracer deux diagrammes de Bode sur le même graphique.
`>>bode(H,G) ; grid on`

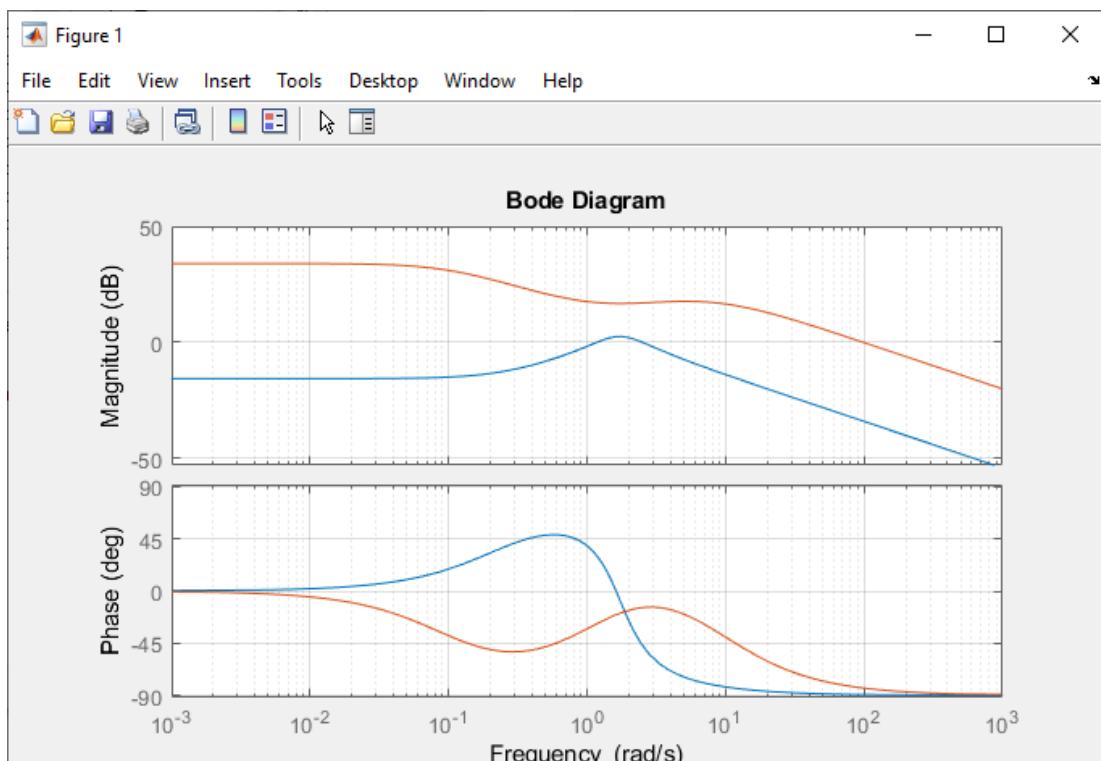


Figure 410 : tracé de deux diagrammes de Bode sur le même graphique

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 boucle 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); grid on;
```

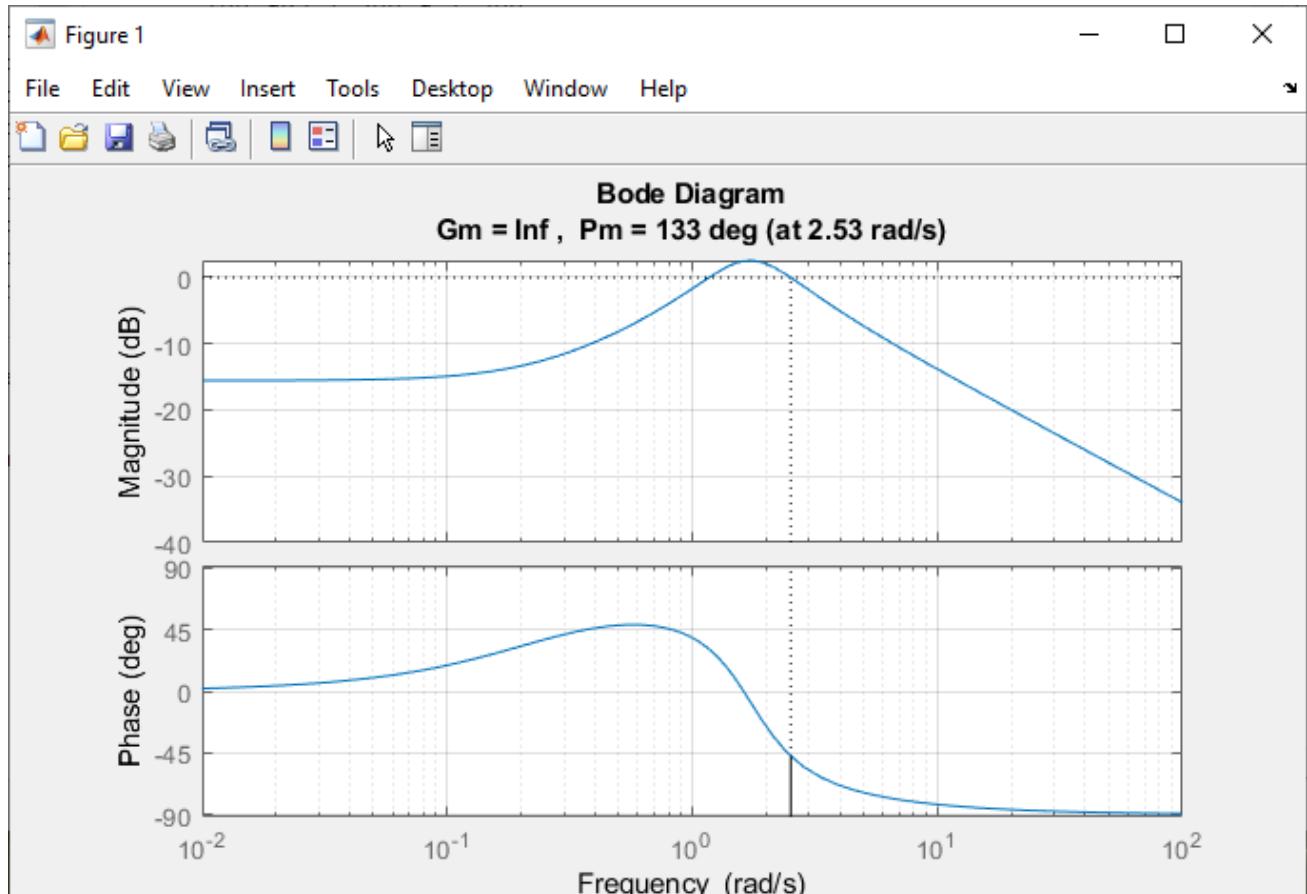


Figure 411 : diagramme de Bode avec indication des marges de gain et de phase

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	<code>syms x</code>
Résolution d'une équation algébrique	<code>solve(equation,x)</code>
Factoriser une expression	<code>factor (x^3-4*x^2-28*x-32)</code>
Développer une expression	<code>expand ((x-8)*(x+2)^2)</code>
Dériver une fonction	<code>diff (f,t)</code>
Intégrer une fonction	<code>int (f,t)</code>
Résolution d'une équation différentielle	<code>dsolve(equation)</code>
Calculer une intégrale définie	<code>int(f,t,-pi/3,pi)</code>
Calculer la transformée de Laplace d'une fonction	<code>laplace(sin(w*t))</code>
Calculer la transformée inverse de Laplace d'une fonction	<code>ilaplace(3/(2+s)-5/(3*s)^2)</code>
Décomposer en éléments simples une fraction rationnelle	<code>[r p k]=residue (a,b)</code>
Création d'une fonction de transfert à partir de la définition des coefficients des polynômes	<code>H=tf([4 1],[2 3 6])</code>
Création d'une fonction de transfert à partir de la connaissance du gain, des pôles et des zéros	<code>G=zpk([-1,-2],[-0.1,-4,-10],100)</code>
Indiquer que s est la variable de Laplace	<code>s=tf('s')</code>
Création directe d'une fonction de transfert après utilisation de la commande « <code>s=tf('s')</code> »	<code>G=100*((s+1)*(s+2))/((s+0.1)*(s+4)*(s+10))</code>
Création de la fonction de transfert d'un correcteur PID	<code>pid(Kp,Ki,Kd)</code>
Fonctions de transfert en série	
	<code>series(H,G)</code>
Fonctions de transfert en parallèle	
	<code>parallel(H,G)</code>
Fonction de transfert en boucle fermée	
	<code>feedback(H,G)</code>
Inverser une fonction de transfert	<code>inv(H)</code>
Obtenir les pôles d'une fonction de transfert	<code>pole(H)</code>
Obtenir les zéros d'une fonction de transfert	<code>zero(H)</code>
Réponse indicielle	<code>step(H)</code>
Réponse impulsionale	<code>impulse(H)</code>

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	<i>permet d'ouvrir la fenêtre d'aide de la commande « bode »</i>
------------	--

Chapitre 6 : 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 413 et explorer les différents blocs du modèle. Ce modèle **Simulink** représente l'asservissement en température d'un four. Les éléments modélisés par les blocs sont indiqués sur la Figure 413.

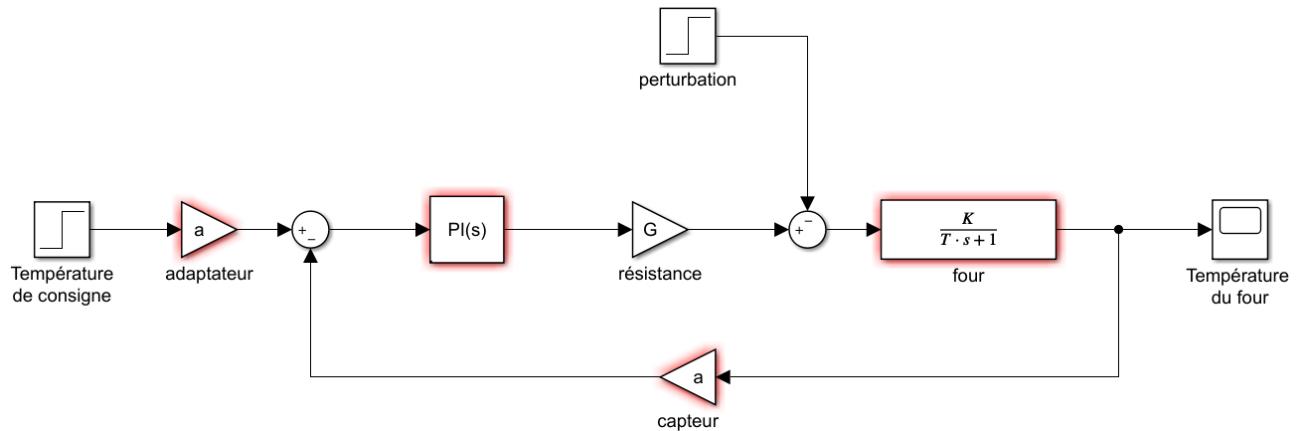


Figure 412 : ouverture du modèle Simulink du four

Nous constatons que certains blocs apparaissent encadrés en rouge. Cela signifie qu'ils sont paramétrés avec des variables qui ne sont pas encore définies dans le Workspace. Le modèle ne peut donc pas être exécuté pour l'instant sans générer un message d'erreur. Lorsque vous construisez un modèle Simulink, vous pouvez paramétriser les blocs directement avec des valeurs numériques ou au travers de variables que vous définirez dans le Workspace, soit directement dans la fenêtre de commande soit via l'exécution d'un script. 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.

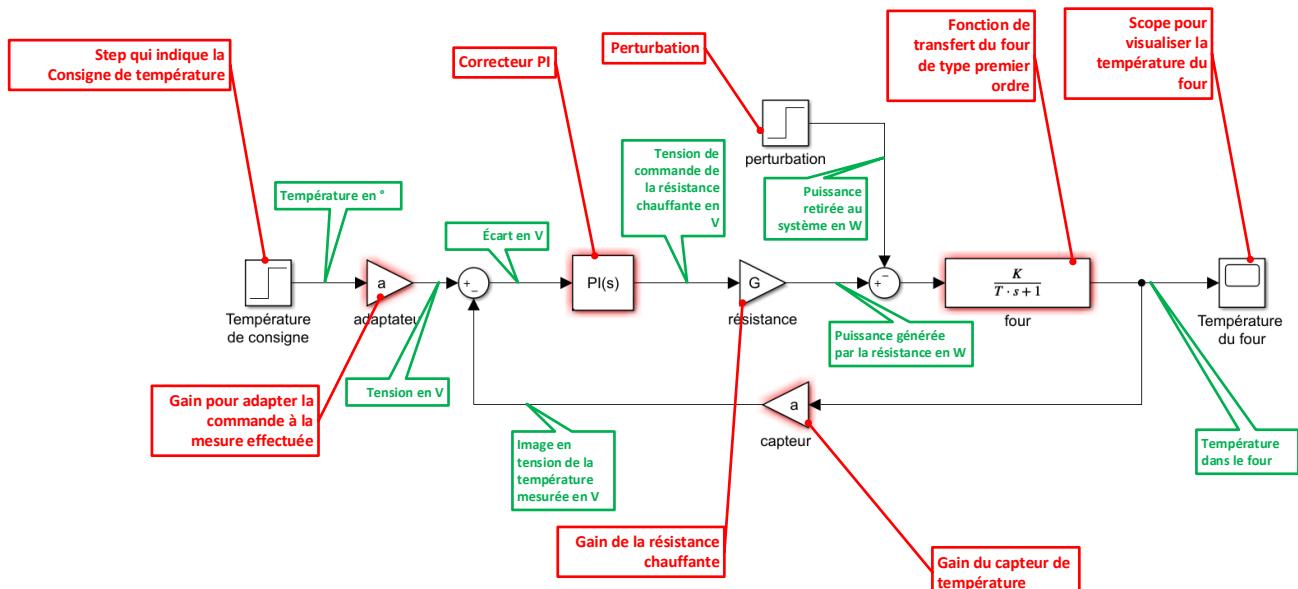


Figure 413 : modélisation de l'asservissement en température d'un four

B. Ouverture du script contenant la définition des variables

Ouvrir le script **parametres_four.m**, en cliquant sur **Open script**



```
%gain du capteur
a=0.5;
%tension maxi aux bornes de la résistance
Vsat=100;
%gain de la resistance chauffante
G=0.8;
%paramètres de la fonction de transfert du four
K=15;
T=2500;
%paramètres du correcteur
Kp=10;
Ki=0.01;
```

Figure 414 : script contenant les paramètres de la simulation

Exécuter le script, en cliquant sur **Run**



dans la barre de commande.

Nous constatons que les variables ont été créés et sont visible dans le Workspace de **MATLAB** (Figure 415)

Name	Value	Class
a	0.5000	double
G	0.8000	double
K	15	double
Ki	0.0100	double
Kp	10	double
T	2500	double
Vsat	100	double

Figure 415 : visualisation des variables créées dans le Workspace

La simulation peut maintenant être lancée.

C. Lancement de la simulation

Lancer la simulation du modèle Simulink et visualiser la variation de la température à l'intérieur du four (Figure 416).

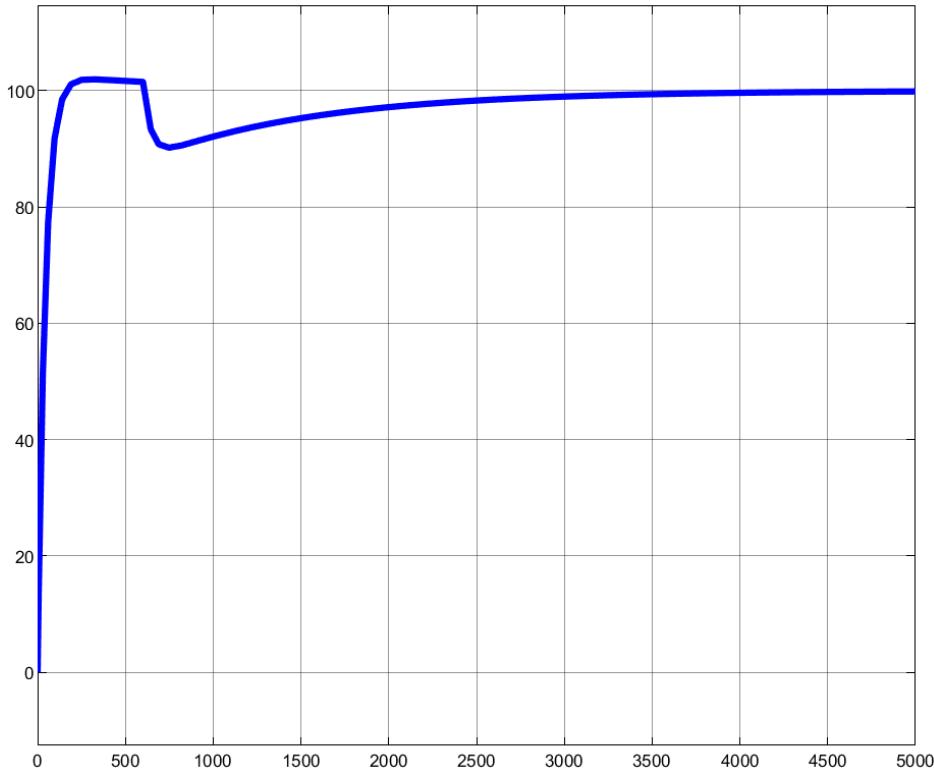


Figure 416 : réponse en température du four

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 417 :

- Entrée
- Sortie
- Ecart
- Retour

Pour nommer un signal **cliquer avec le bouton droit** sur le signal puis choisir **Properties**. La boîte de dialogue **Signal Properties** s'ouvre. Le nom du signal est à spécifier dans **Signal Name**.

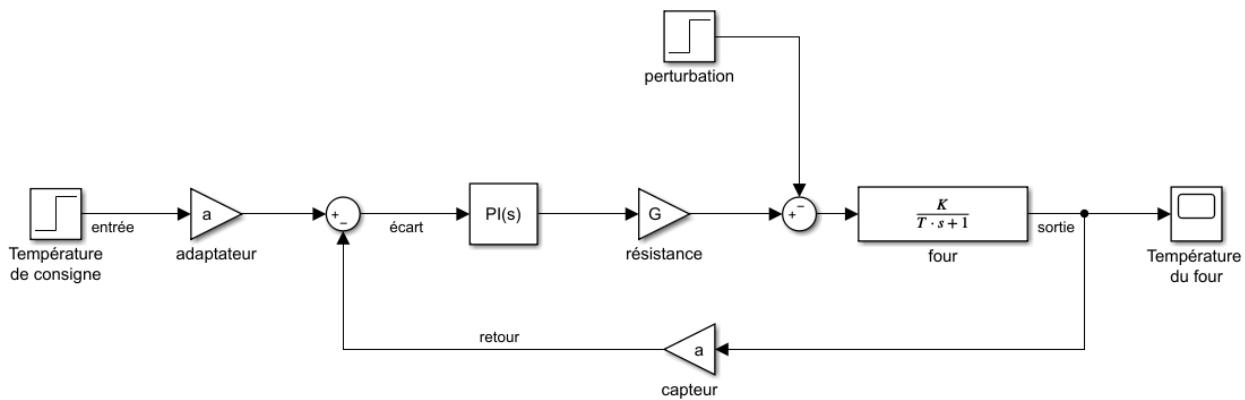


Figure 417 : nommer un signal Simulink

1. Tracer un diagramme 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)

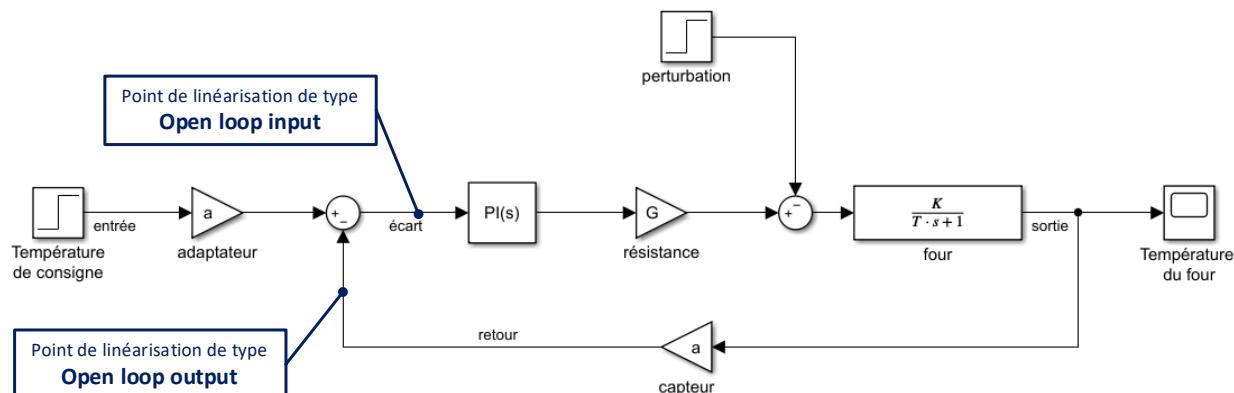


Figure 418 : placement des points de linéarisation pour tracer un diagramme de Bode en boucle ouverte

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**.

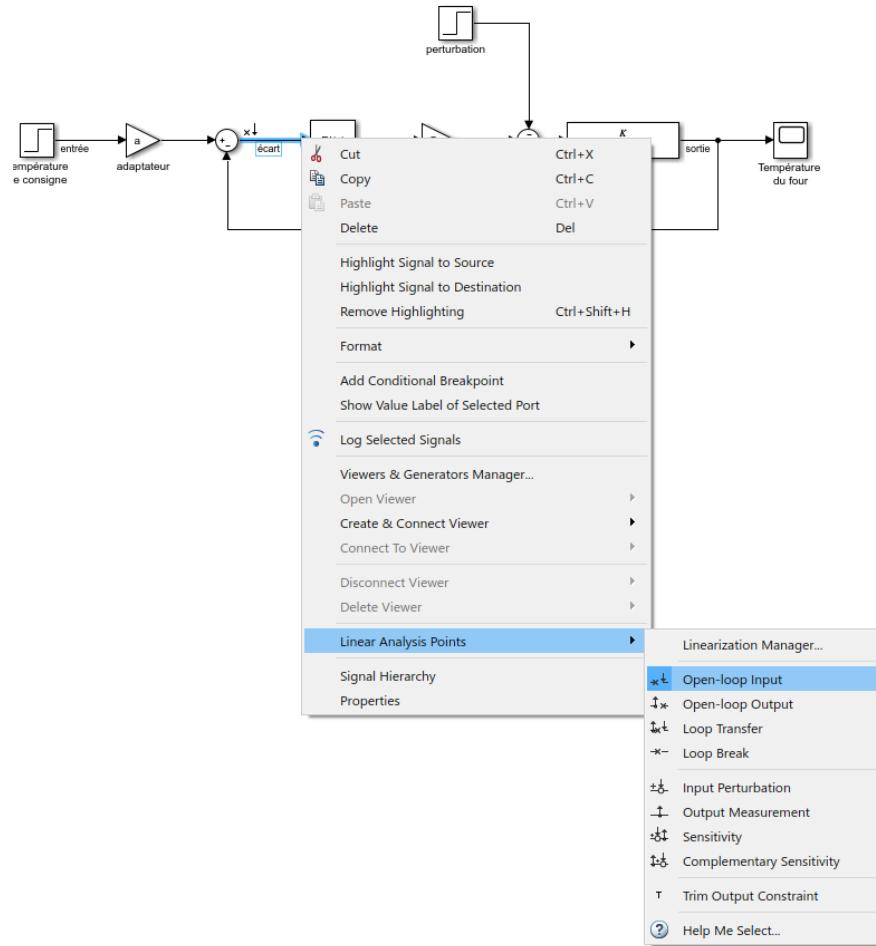


Figure 419 : placement d'un point de linéarisation de type « Open loop input »

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**.

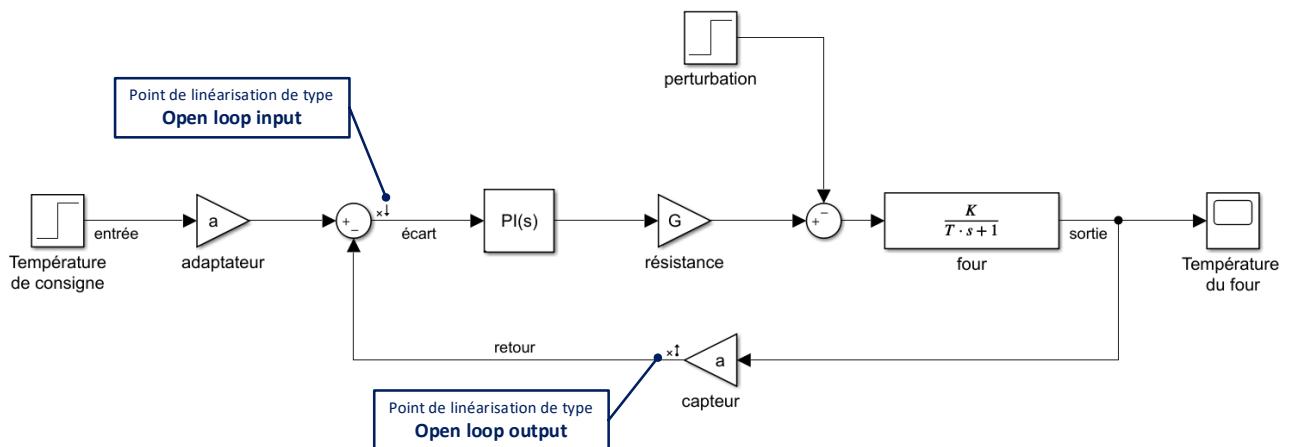
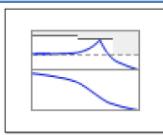


Figure 420 : représentation des points de linéarisation

Fonction du composant	Représentation	Bibliothèque
Tracer d'un diagramme de Bode	 Bode Plot	Simulink/Simulink Control Design/Linear Analysis Plot

Insérer dans votre modèle un bloc **Bode Plot**, renommer ce bloc « **boucle ouverte** ».

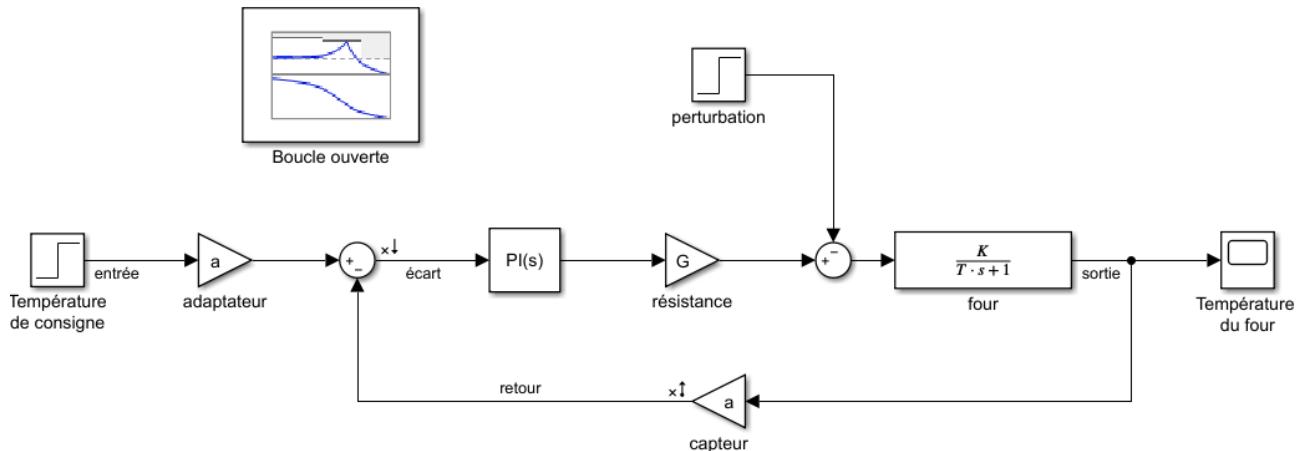


Figure 421 : insertion d'un bloc Bode Plot

Double cliquer sur le bloc pour l'ouvrir.

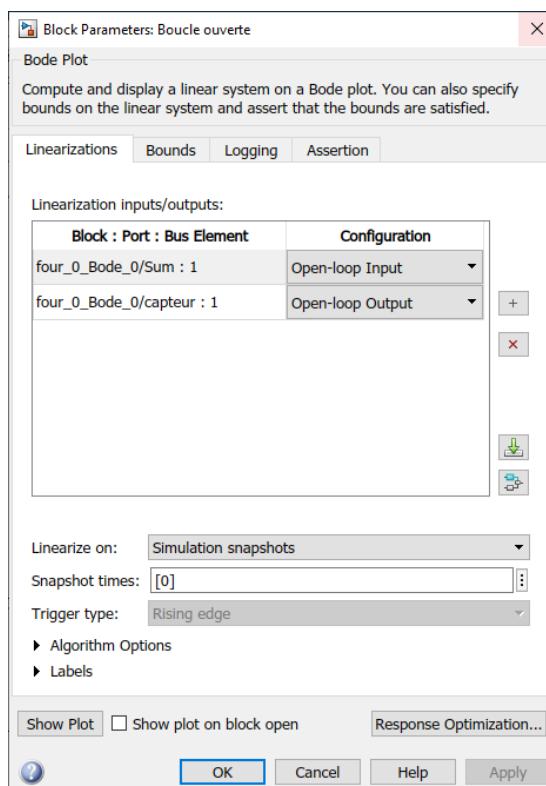


Figure 422 : fenêtre de paramétrage du bloc Bode Plot

Par défaut les deux points de linéarisation créés apparaissent.
 Cliquer sur **Show Plot** pour faire apparaître la fenêtre de tracé.

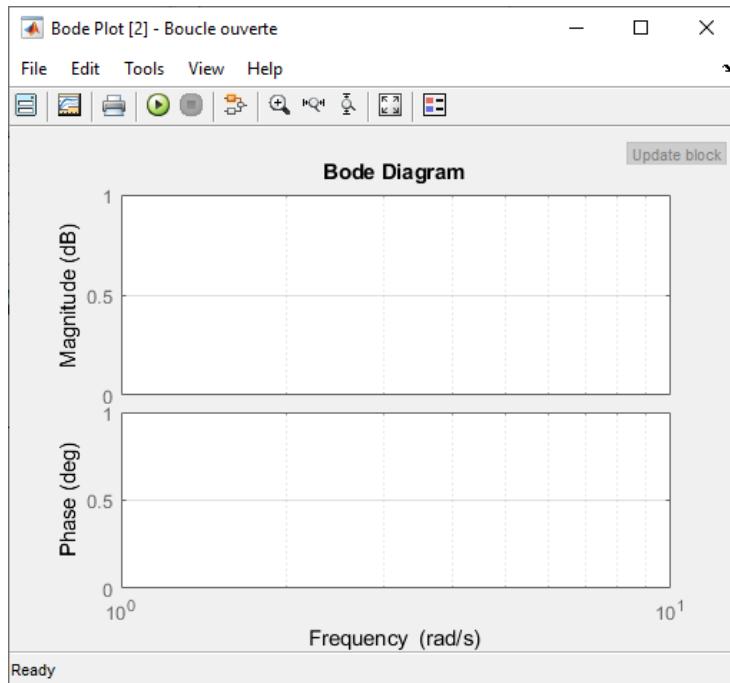


Figure 423 : fenêtre de tracé du diagramme de Bode

Cliquer sur Run pour linéariser le modèle. Le diagramme de Bode se trace automatiquement.

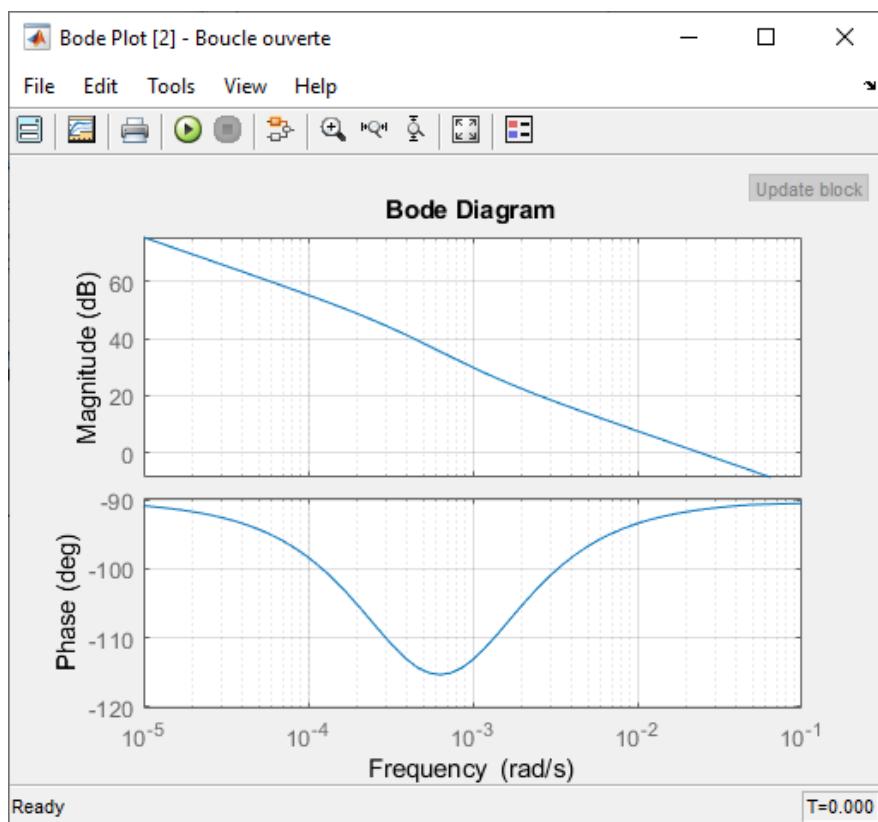


Figure 424 : diagramme de Bode de la boucle ouverte

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 425.

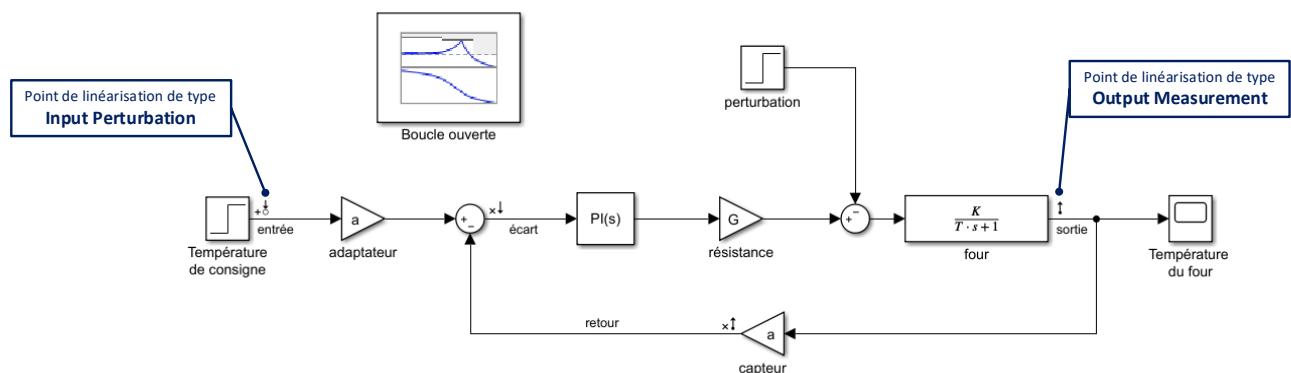


Figure 425: représentation des points de linéarisation

Insérer dans votre modèle un second bloc **Bode Plot**, renommer ce bloc « **boucle fermée** ».

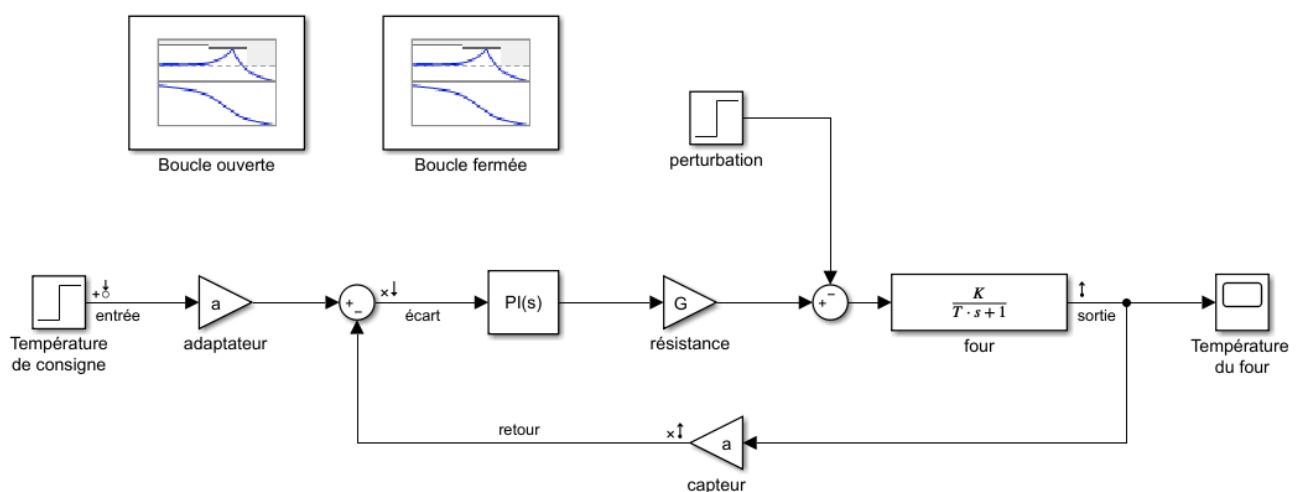


Figure 426 : insertion d'un second bloc Bode plot

Double cliquer sur le bloc pour l'ouvrir.

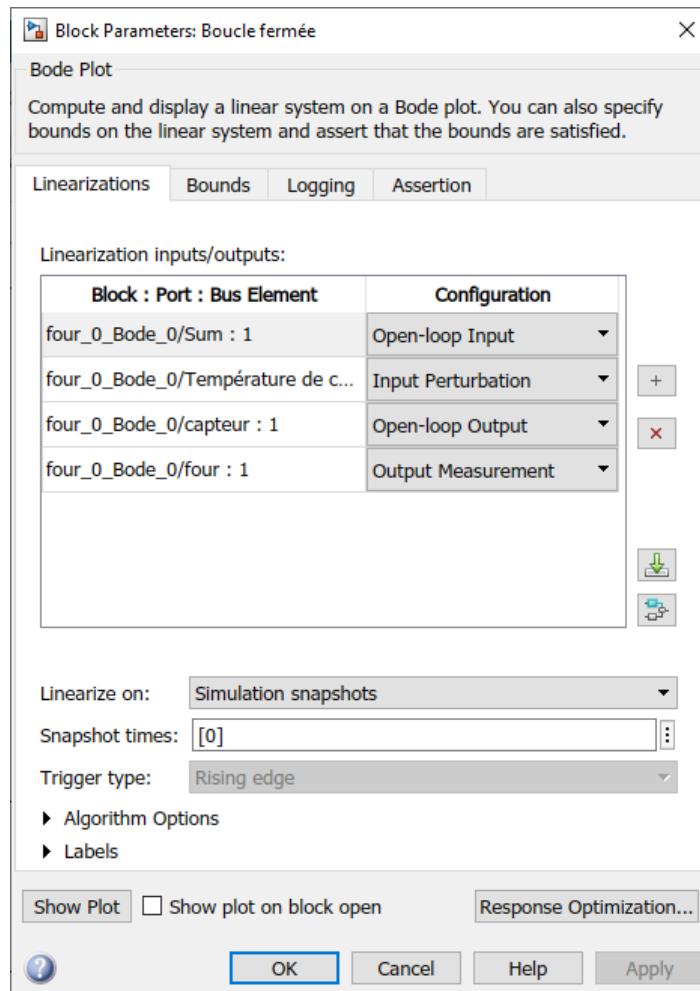


Figure 427: fenêtre de paramétrage du bloc Bode Plot

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 428.

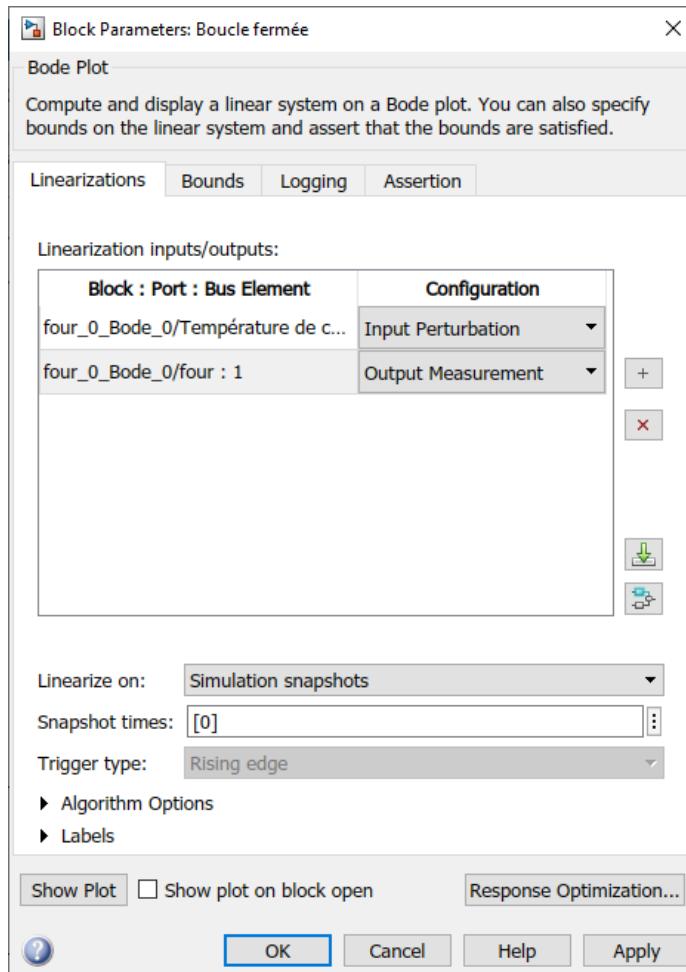


Figure 428 : paramétrage des points de linéarisation pour la boucle fermée

Cliquer sur **Show Plot** puis cliquer sur **Run** pour faire apparaître le diagramme de Bode

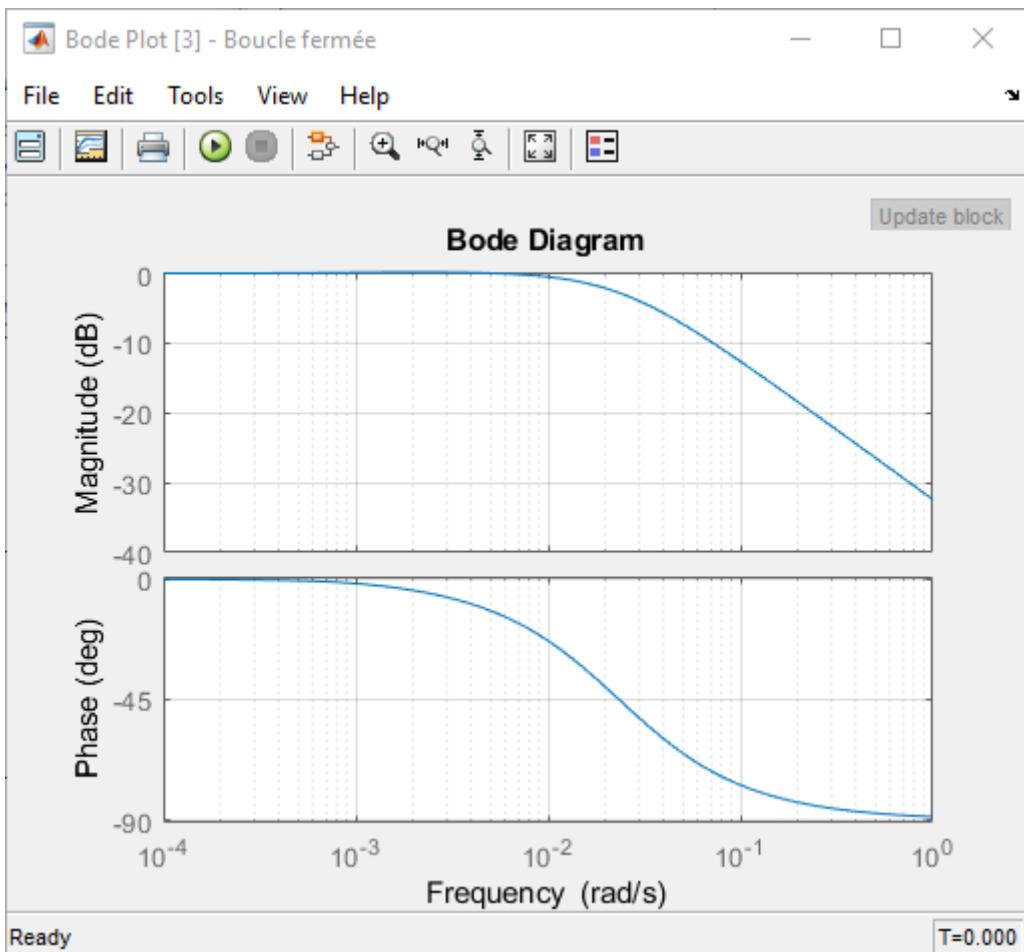
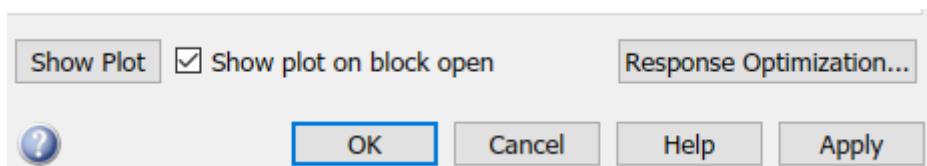


Figure 429 : diagramme de Bode de la boucle fermée

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 facilement et visualiser très rapidement l'allure des diagrammes de Bode. A chaque modification du modèle, il faudra relancer la linéarisation du modèle pour obtenir le nouveau diagramme 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**.

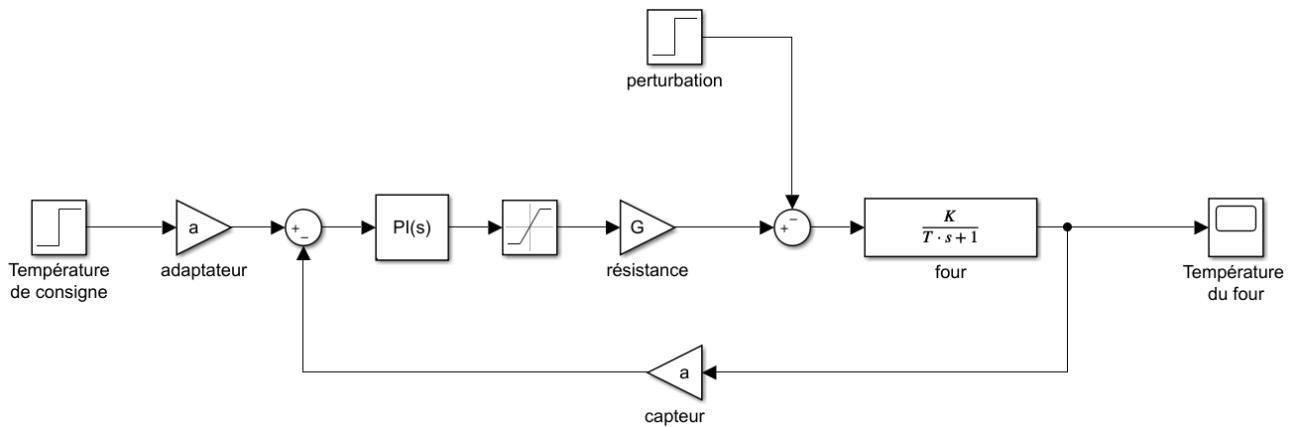


Figure 430 : asservissement en température d'un four avec saturation

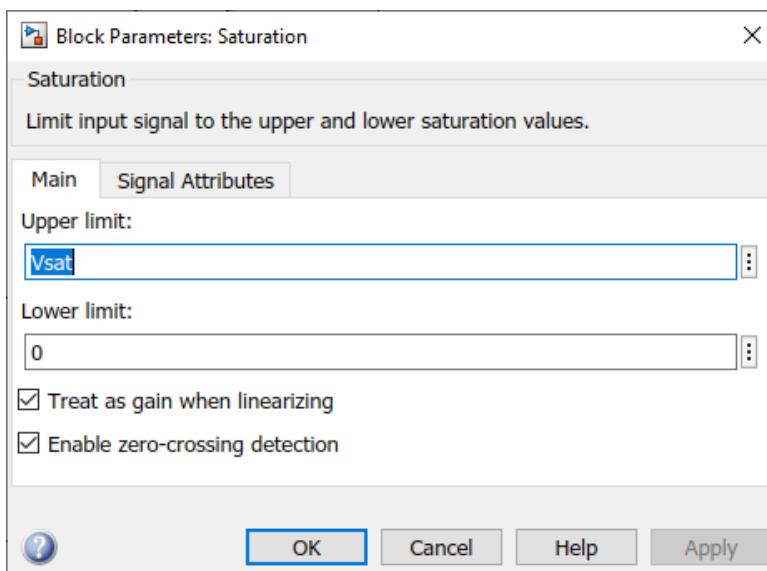
Paramétrage

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.

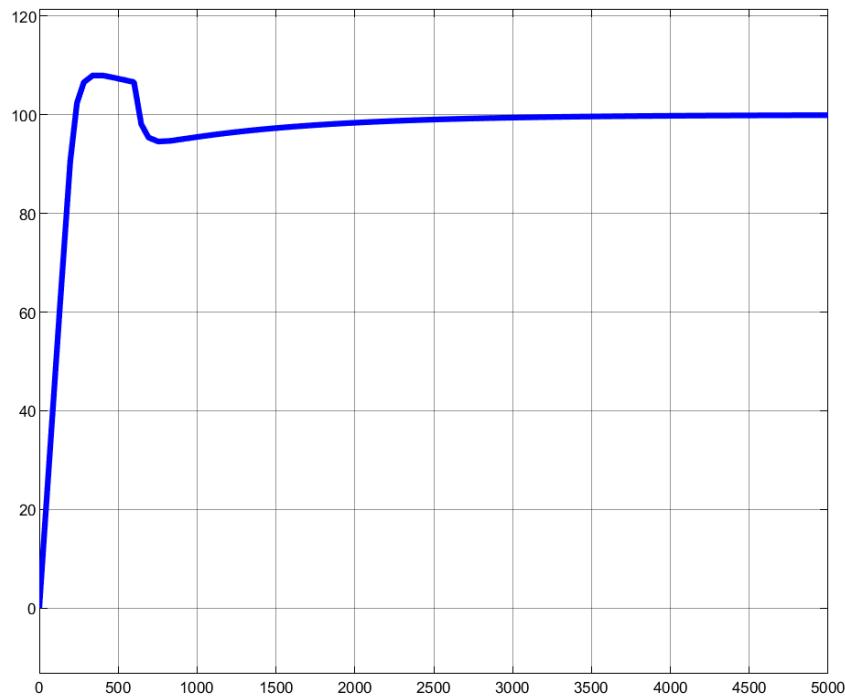
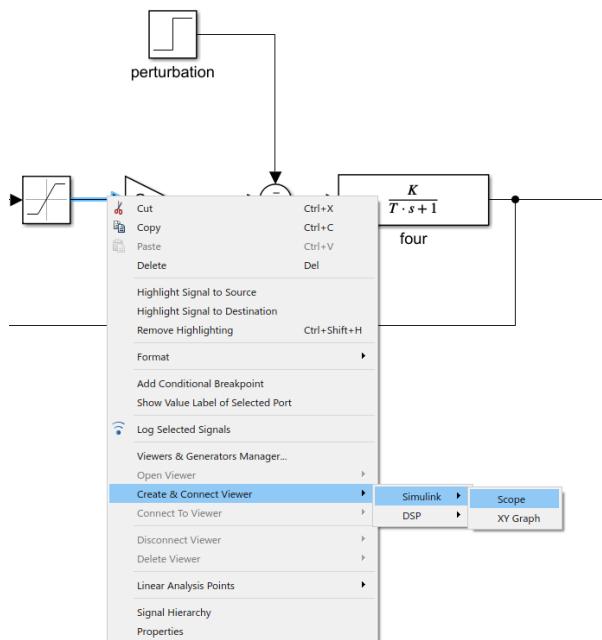


Figure 431 : réponse en température de four avec saturation

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.

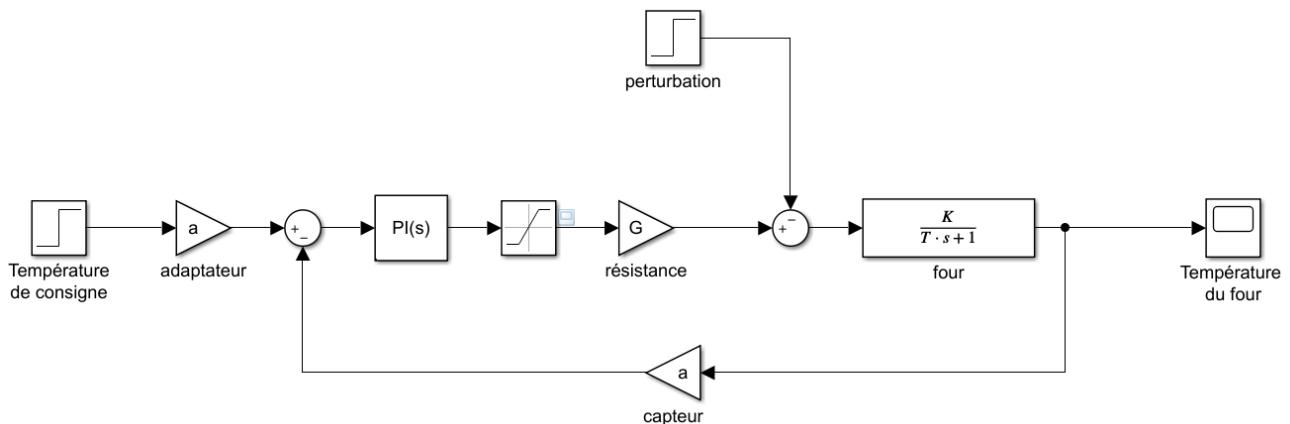


Figure 432 : mise en place d'un scope sur un signal

Relancer la simulation et double cliquer sur le scope du saturateur pour visualiser la tension en sortie du bloc.

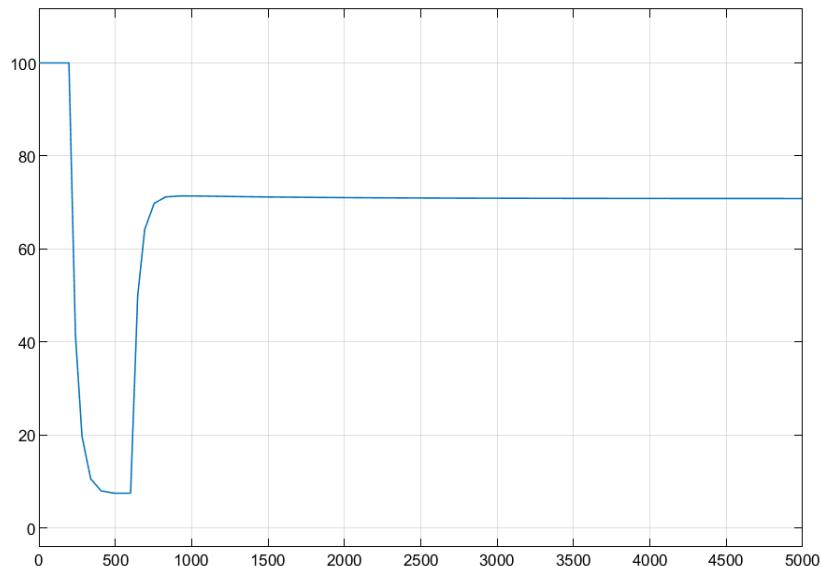


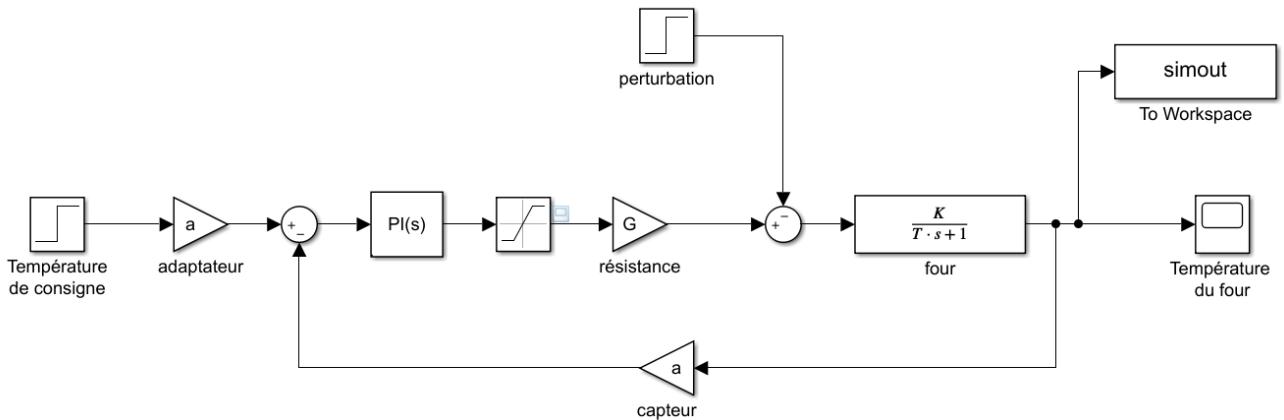
Figure 433 : visualisation de la tension en sortie de saturateur

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

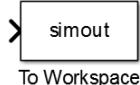
Il est très souvent utile d'exporter des variables qui contiennent le résultat de simulation vers le **Workspace**. Cela permet d'utiliser toutes les fonctions de **MATLAB** pour la présentation des résultats.

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**.



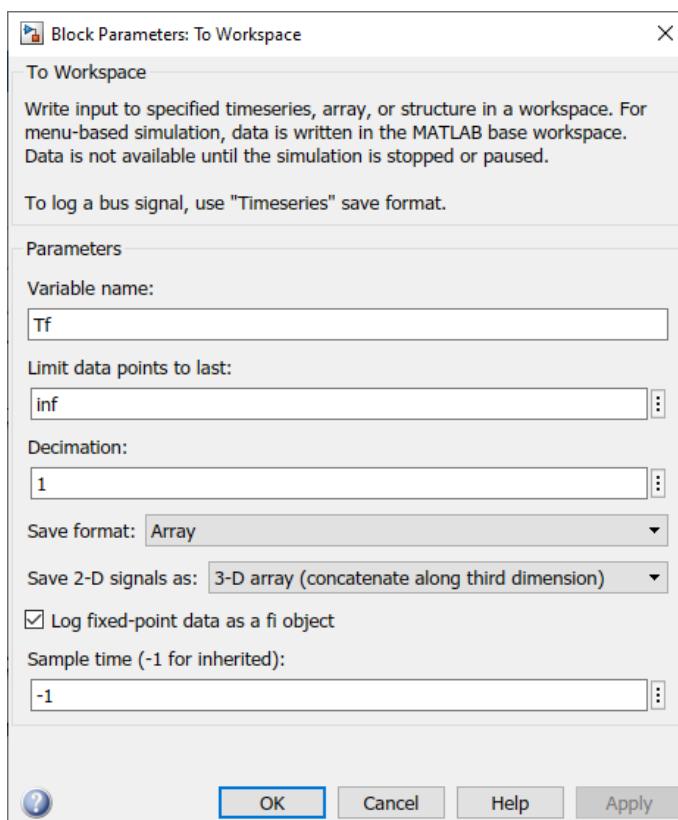
Paramétrage

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**.

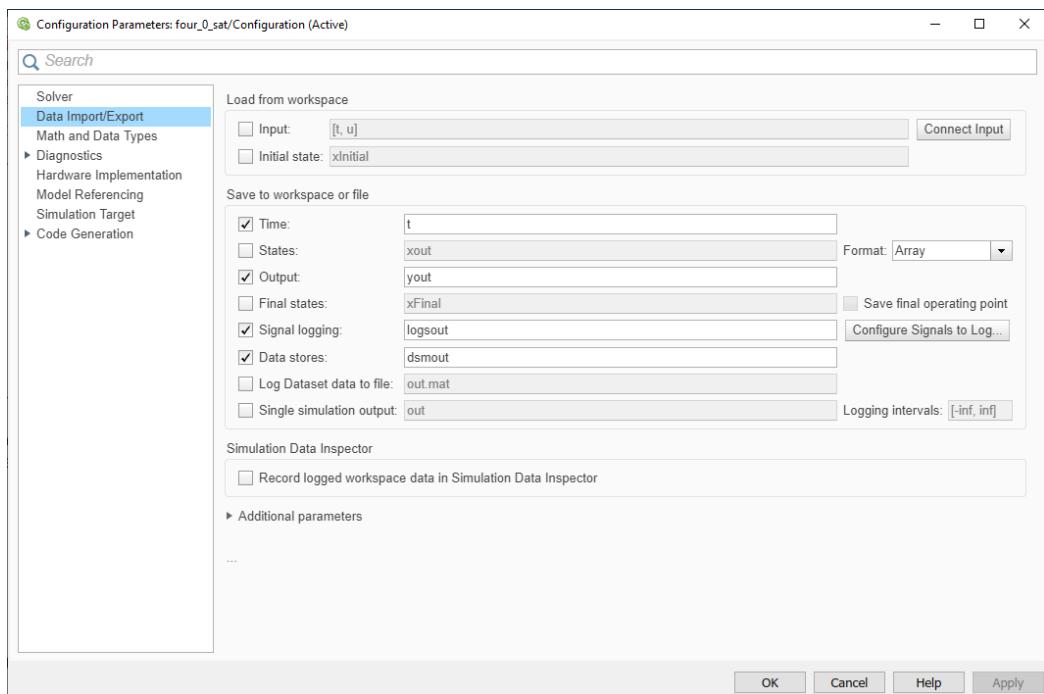


Figure 434 : fenêtre de paramétrage de l'exportation des données vers le Workspace

Enregistrer le modèle sous le nom *four_export_name.slx*

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

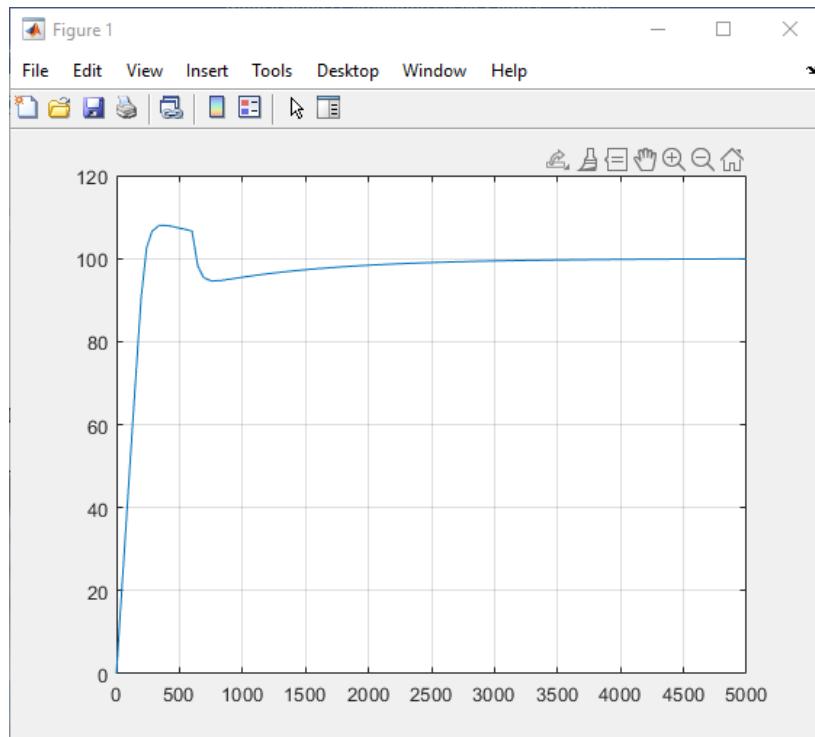
Lancer la simulation.

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.

```
close all;
figure;
hold all;
grid on;
% pour Kp variant de 1 à 10 avec un pas de 1 (valeur par défaut du pas)
% exécute la simulation du fichier simulink four_import
% trace la température en fonction du temps
for Kp=1:10
    sim('four_export');
    plot(t,Tf,'LineWidth',2);
end
```

Figure 435 : script pour tracer une série de courbes

Le script est disponible dans le fichier **script_courbes_Kcor.slx**.

Vous devez obtenir le résultat suivant.

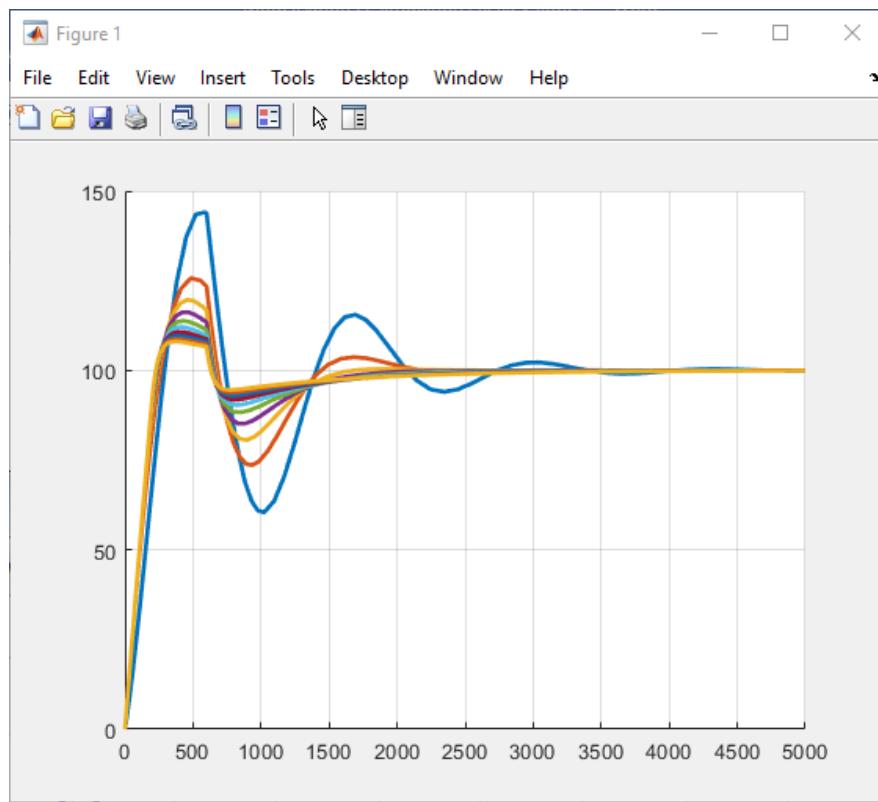


Figure 436 : influence de la correction proportionnelle sur la température du four

H. Linéarisation d'un modèle Simulink à partir d'un script

Il est souvent utile de linéariser un modèle Simulink à partir d'un script pour pouvoir exploiter les résultats.

Ouvrir le fichier **four_linearisation.slx**. Ce modèle contient le modèle du four avec les deux points de linéarisation permettant de définir la fonction de transfert en boucle ouverte (Figure 437).

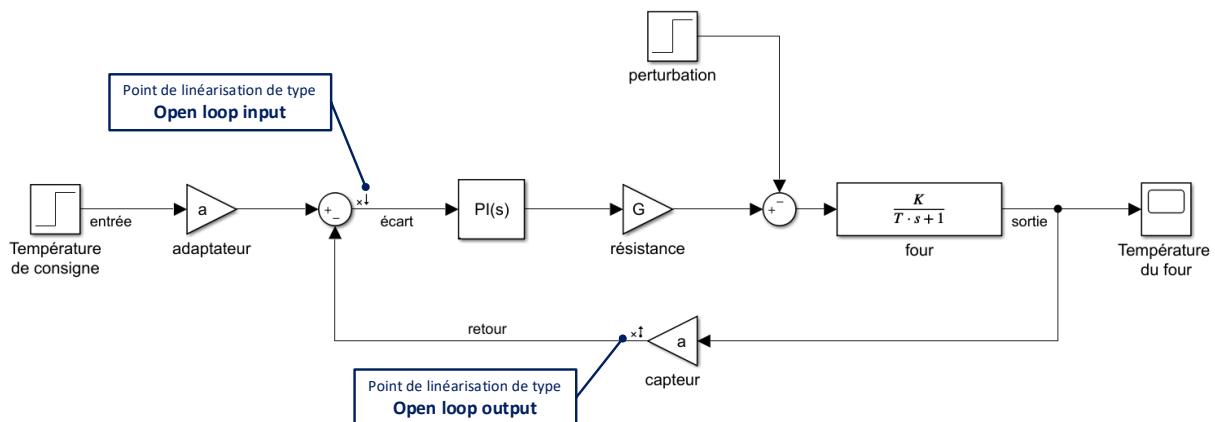


Figure 437 : modèle du four pour la linéarisation

Il est possible à partir du script de la Figure 438 de tracer le diagramme de Bode en boucle ouverte du schéma bloc Simulink (**linearisation_schema_bloc.slx**).

```
% affectation dans une variable du nom du fichier contenant le schéma bloc
% à linéariser
model = 'four_linearisation';
% récupération des points de linéarisation
io = getlinio(model);

% création de la fonction de transfert à partir des points de linéarisation
FTBO = tf(linearize(model,io));

% tracé du diagramme de Bode en boucle ouverte
bode(FTBO);
grid on;
```

Figure 438 : script permettant de linéariser un modèle Simulink

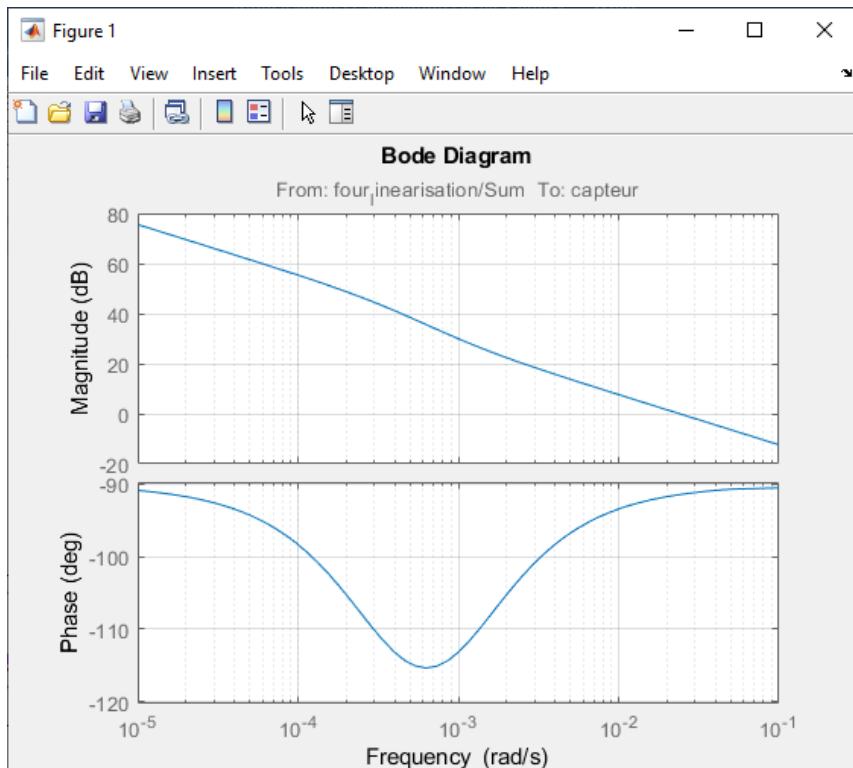


Figure 439 : diagramme de Bode obtenu en linéarisant le modèle Simulink

Le script de la Figure 440 permet de tracer une série de diagramme de Bode en faisant varier le gain du correcteur proportionnel (**linearisation_schema_bloc_courbes_Kcor.slx**).

```
% affectation dans une variable du nom du fichier contenant le schéma bloc
% à linéariser
model = 'four_linearisation';

for Kp=1:10

% récupération des points de linéarisation
io = getlinio(model);

% création de la fonction de transfert à partir des points de linéarisation
FTBO = tf(linearize(model,io));

% tracé du diagramme de Bode en boucle ouverte
bode(FTBO);hold on;
end
```

Figure 440 : script permettant de tracer une série de diagrammes de Bode en faisant varier le gain proportionnel du correcteur.

Le résultatat de l'exécution du script est visible sur la Figure 441.

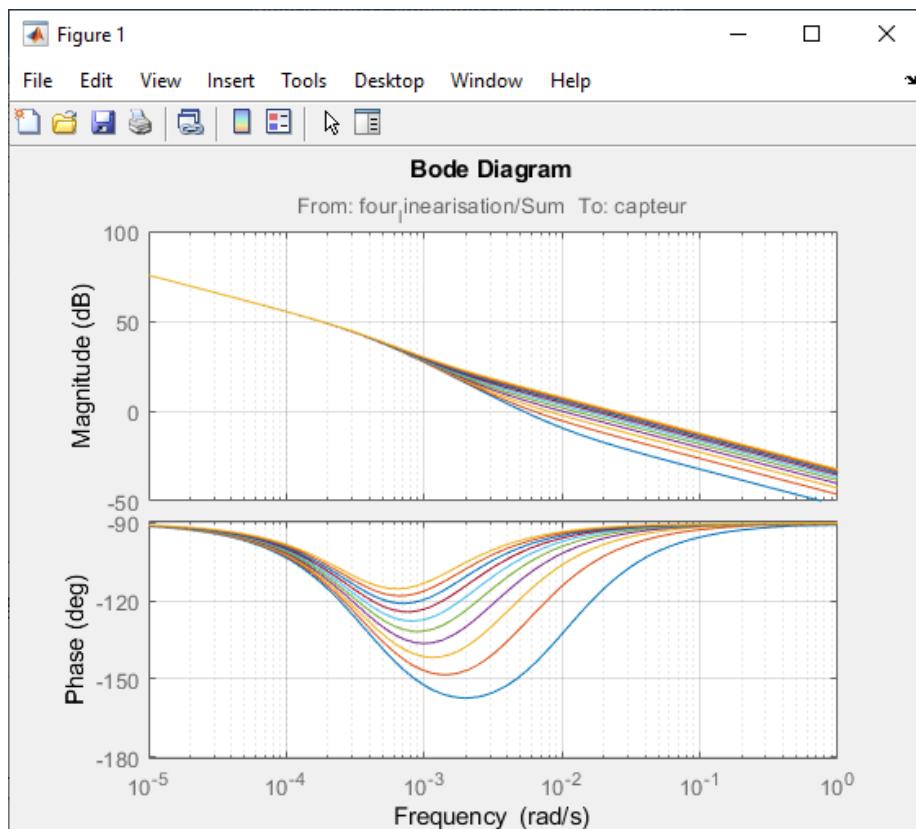


Figure 441 : diagrammes de Bode obtenus en faisant varier le gain du correcteur proportionnel

Chapitre 7 : 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

Afin de ne pas solliciter la batterie du pilote en permanence, la correction de cap ne se fera qu'à partir du moment où l'erreur de cap atteint un seuil de 1° .

Le graphe d'état comporte 3 états distincts :

Etat « Pause »

Le système est dans cet état si l'erreur de cap est inférieure à 0.1° . Le pilote hydraulique maintient la consigne de barre à 0° tant que l'erreur de cap ne dépasse pas 0.1° . 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 0.1° . Si l'erreur de cap se maintient pendant 20s au-dessus de 0.1° , 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 redévie 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 « chaîne d'information » et observer le diagramme à compléter sur la Figure 442.

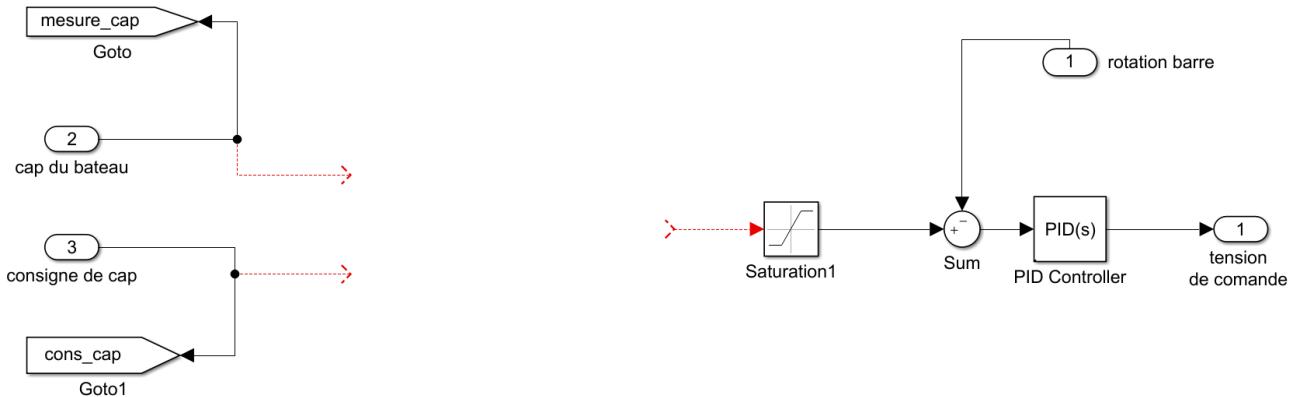


Figure 442 : boucle de cap sans le diagramme d'état

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** (Figure 443).

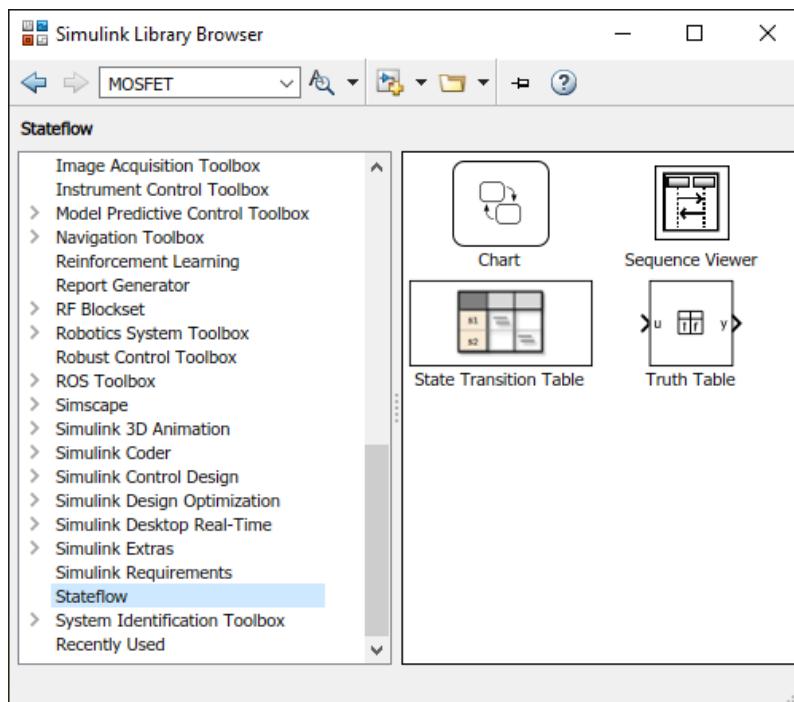
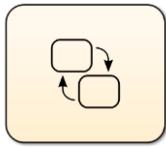


Figure 443 : bibliothèque Statflow

Fonction du composant	Représentation	Bibliothèque
Chart	 Chart	Statflow

Positionner et redimensionner le chart pour obtenir la configuration de la Figure 444.

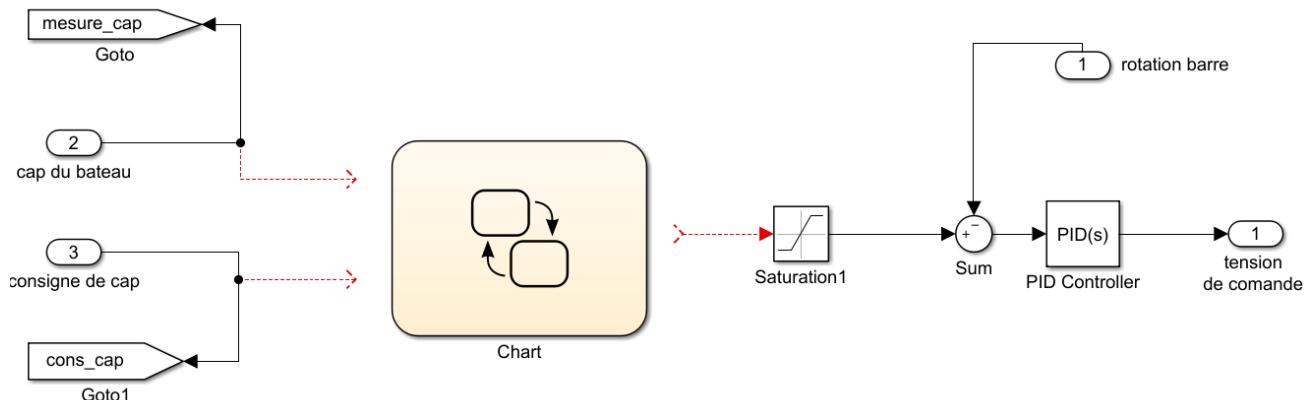


Figure 444 : positionnement d'un chart dans un modèle Simulink

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** apparaît à gauche de la fenêtre avec en particulier les commandes qui permettent de créer un état et une « transition par défaut ».

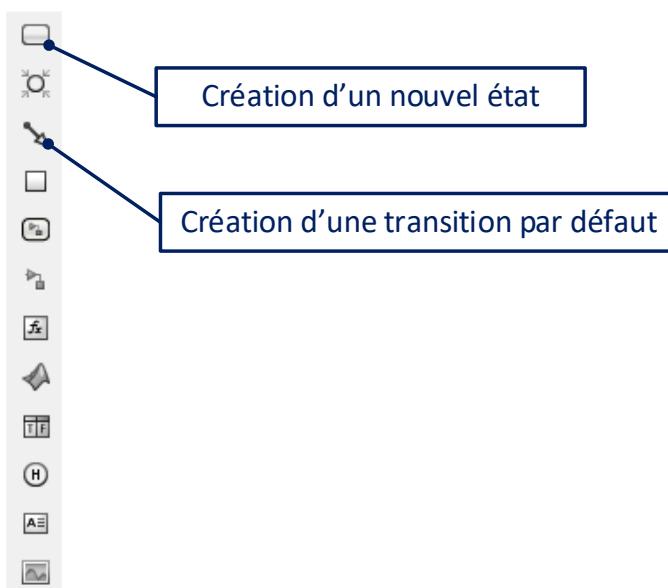


Figure 445 : commande de création d'un état et d'une « transition par défaut »

A partie de l'onglet MODELING, cliquer sur **Symbols Pane** afin de faire apparaître la fenêtre du **Symbols Pane**.

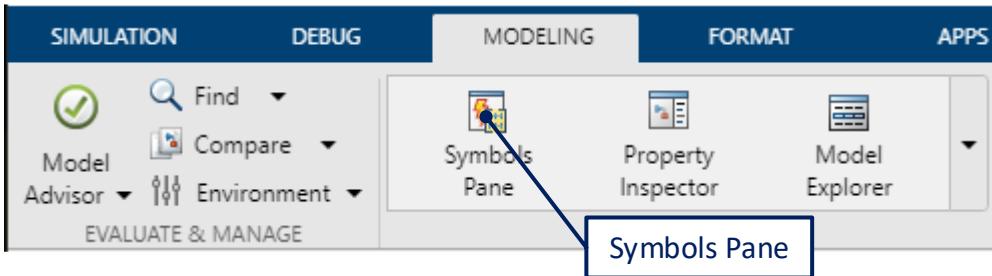


Figure 446 : ouverture de la fenêtre Symbols

La fenêtre **Symbols** apparaît alors sur la partie droite de l'écran. Cette fenêtre permettra de visualiser la nature des variables utilisées dans la conception du graphe d'état avec Stateflow.

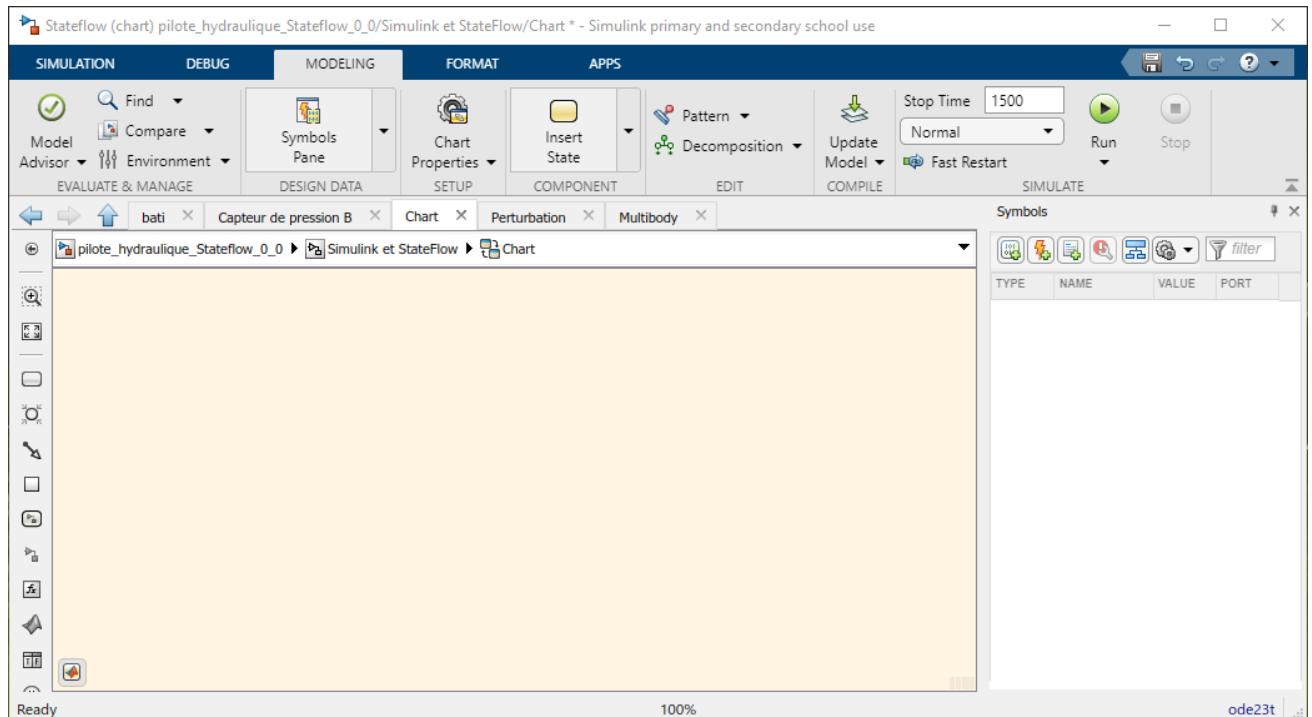


Figure 447 : environnement de travail de Stateflow

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 448. 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. Lors de la création du premier état, une transition par défaut est automatiquement ajoutée.

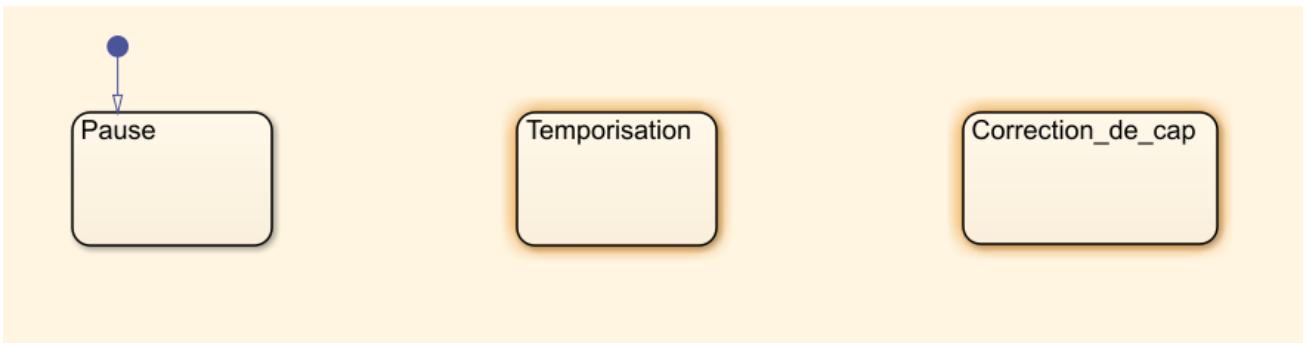


Figure 448 : création des états du système

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 449 : 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 450

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.

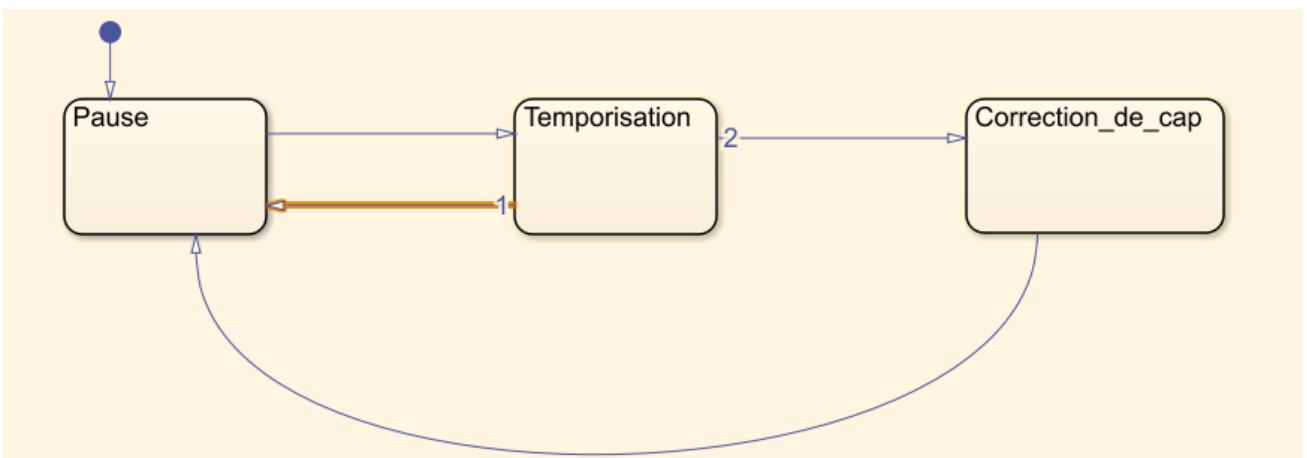


Figure 450 : création des transitions entre les états

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.

Si aucun mot clé n'est spécifié, par défaut l'action sera de type **entry,during** et sera exécutée à la fois lors de l'activation de l'état et à chaque pas de temps tant que l'état est actif.

Compléter les actions des états conformément à la Figure 451.

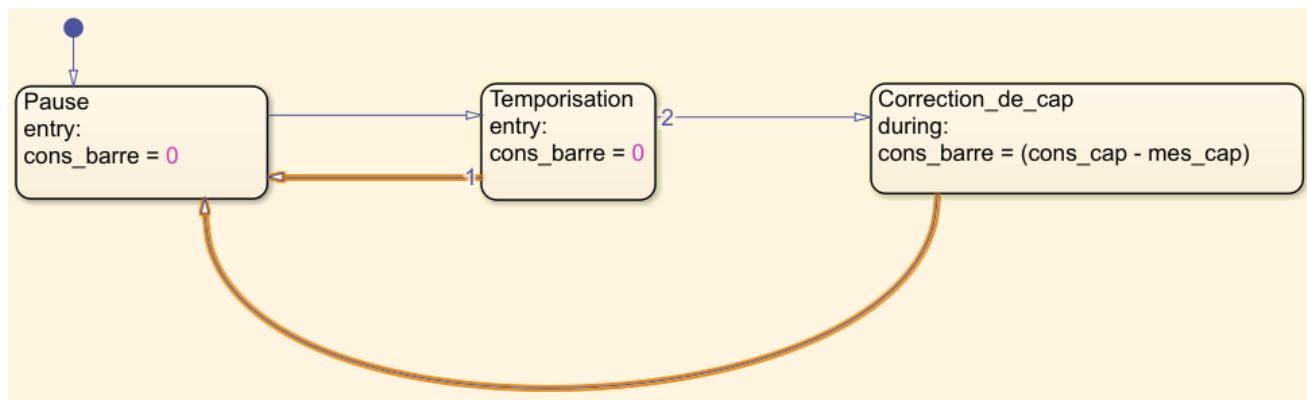


Figure 451 : affectation des actions dans les états

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**.

Symbols			
TYPE	NAME	VALUE	PORT
!	cons_cap		
!	mes_cap		
!	cons_barr		

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 (Figure 452).

Une condition	
Un évènement	
Une action de condition	
Un commentaire	% commentaires

Figure 452 : nature des informations d'une étiquette de transition

Pour créer une étiquette de transition, double cliquer sur la transition pour faire apparaître le menu vous permettant de sélectionner la nature de l'information que vous souhaitez indiquer dans la transition. Cliquer alors sur puisque nous allons saisir des **conditions** sur chacune des transitions.

Pour déplacer une étiquette de transition, utiliser un simple glisser déposer avec le bouton gauche de la souris.

Créer les étiquettes des transitions conformément à la Figure 453.

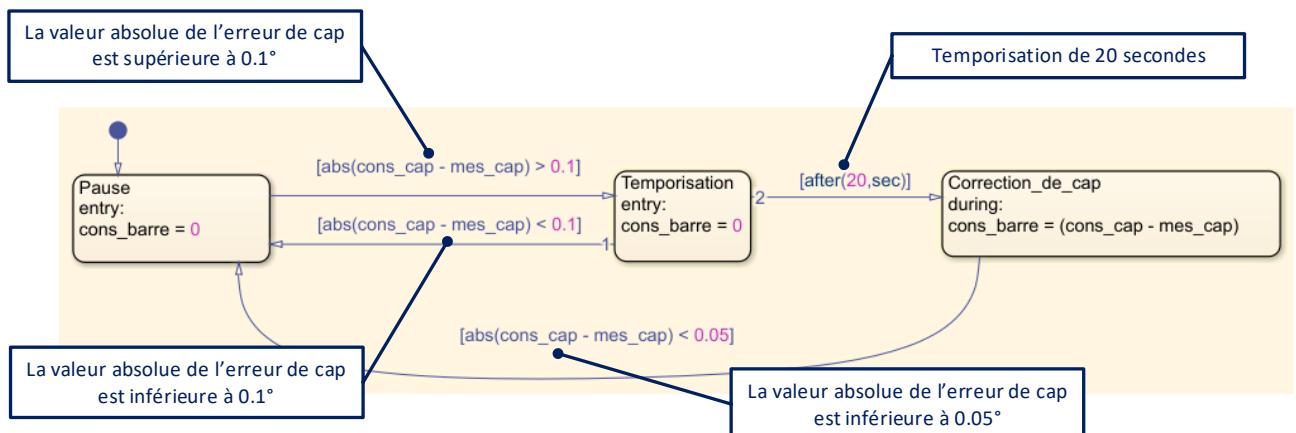


Figure 453 : écriture des étiquettes de transitions dans un diagramme d'états

Les étiquettes de transitions offrent d'autres possibilités qui sont décrites page 368.

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.

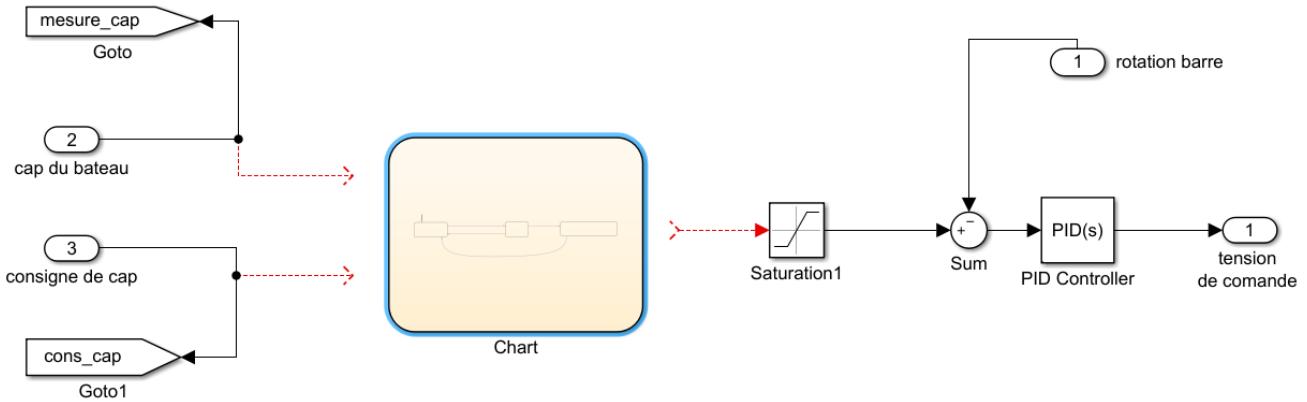


Figure 454 : chart non connecté avec le modèle

Nous pouvons également constater que la fenêtre Symbols montre que la configuration des entrées et sorties de notre système de commande n'est pas encore totalement définie.

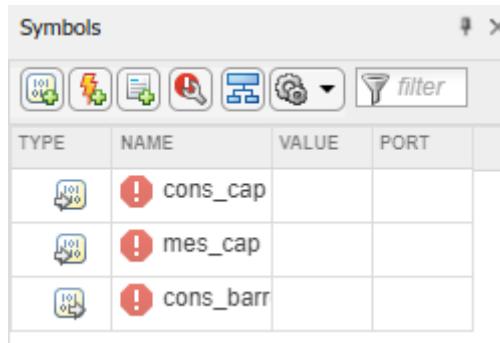


Figure 455 : fenêtre Symbols avec paramétrage incomplet des entrées/sorties

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.

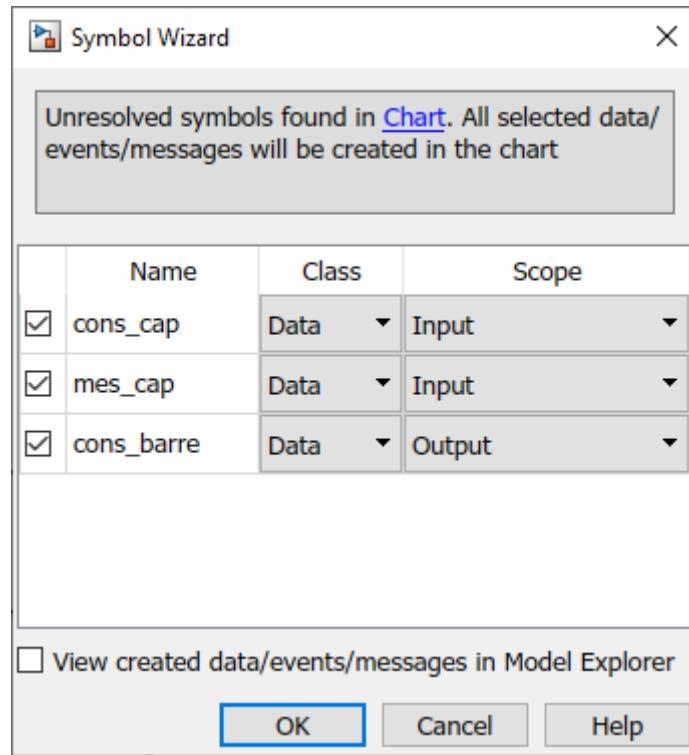


Figure 456 : affectation des variables du diagramme à l'aide du Symbol Wizard

Nous constatons que la fenêtre Symbols indique maintenant que la configuration des entrées et des sorties du modèle Stateflow est finalisée. Cette fenêtre permet de créer rapidement une variable supplémentaire ou un évènement (Figure 457).

TYPE	NAME	VALUE	PORT
101	cons_cap		1
101	mes_cap		2
101	cons_barre		1

Création d'une variable

Création d'un évènement

Figure 457 : la fenêtre Symbols

De manière générale, il est également possible de visualiser en détail, les variables du modèle, à l'aide de la fenêtre du **Model Explorer**. Il est possible d'ouvrir le **Model Explorer** (Figure 458) pour modifier les

caractéristiques d'une variable en cliquant sur  dans la barre de commande principale (onglet MODELING). L'arborescence située sur la partie gauche de la fenêtre permet de naviguer dans le système.

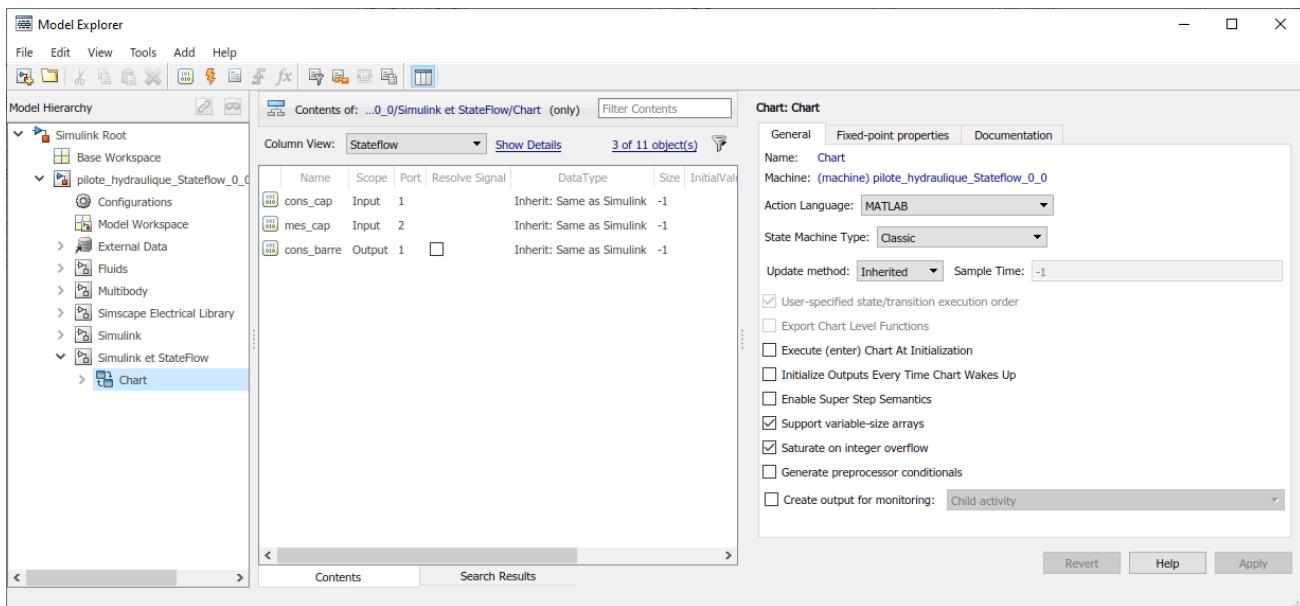


Figure 458 : visualisation des variables dans le Model Explorer

Il est possible de modifier les caractéristiques des différentes variables. Vous pouvez également revenir à tout moment dans le **Model Explorer** pour modifier les caractéristiques d'une variable en cliquant sur



dans la barre de commande principale.

Revenir dans le sous-système représentant la chaîne d'information pour visualiser les connexions qui apparaissent sur le Chart. Une prévisualisation du diagramme d'état est visible par défaut dans le Mask du sous-système.

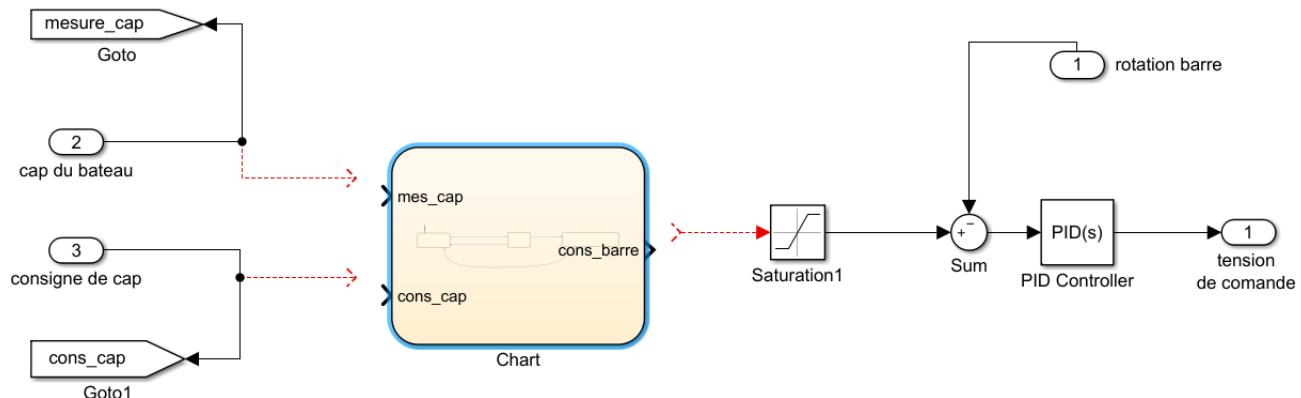


Figure 459 : chart avant connexion avec le système

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, elles peuvent être modifiées dans la fenêtre **Symbols** Figure 460.

Symbols			
TYPE	NAME	VALUE	PORT
	cons_cap		2
	mes_cap	1	
	cons_barre	1	2

Figure 460 : inversions de la position des ports

Cette opération peut également être réalisée à l'aide du **Model Explorer** ou l'on peut également indiquer les bons numéros de port comme indiqué sur la Figure 461.

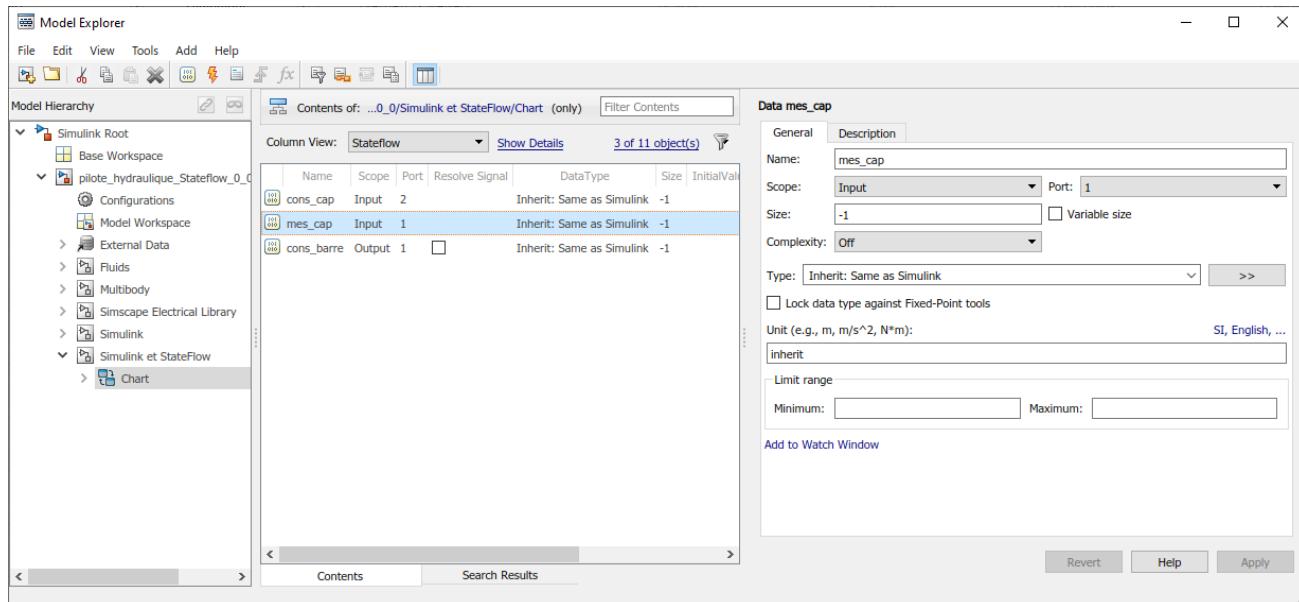


Figure 461 : modification des numéros de port d'un chart

Connecter le chart avec le schéma bloc Simulink conformément à la Figure 462.

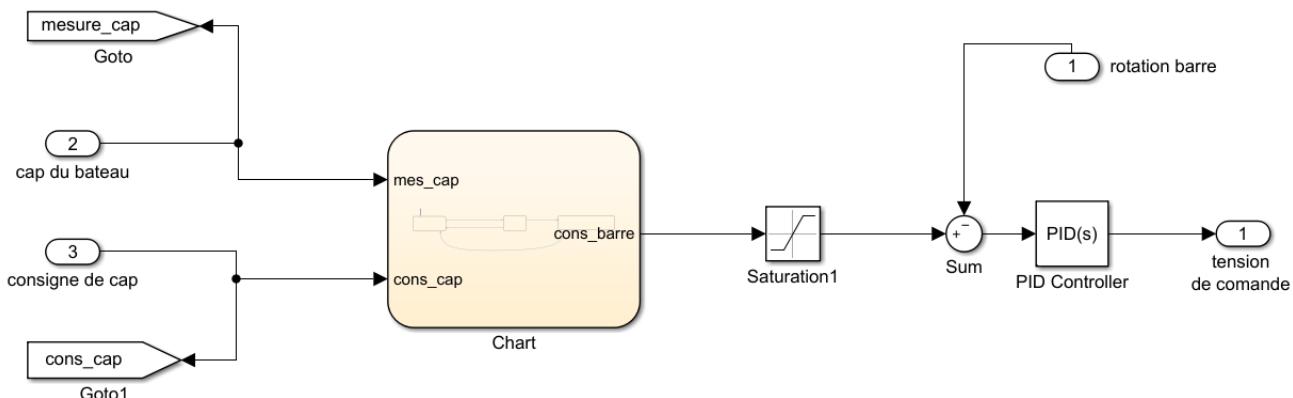


Figure 462 : connexion du chart avec le schéma bloc Simulink

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.

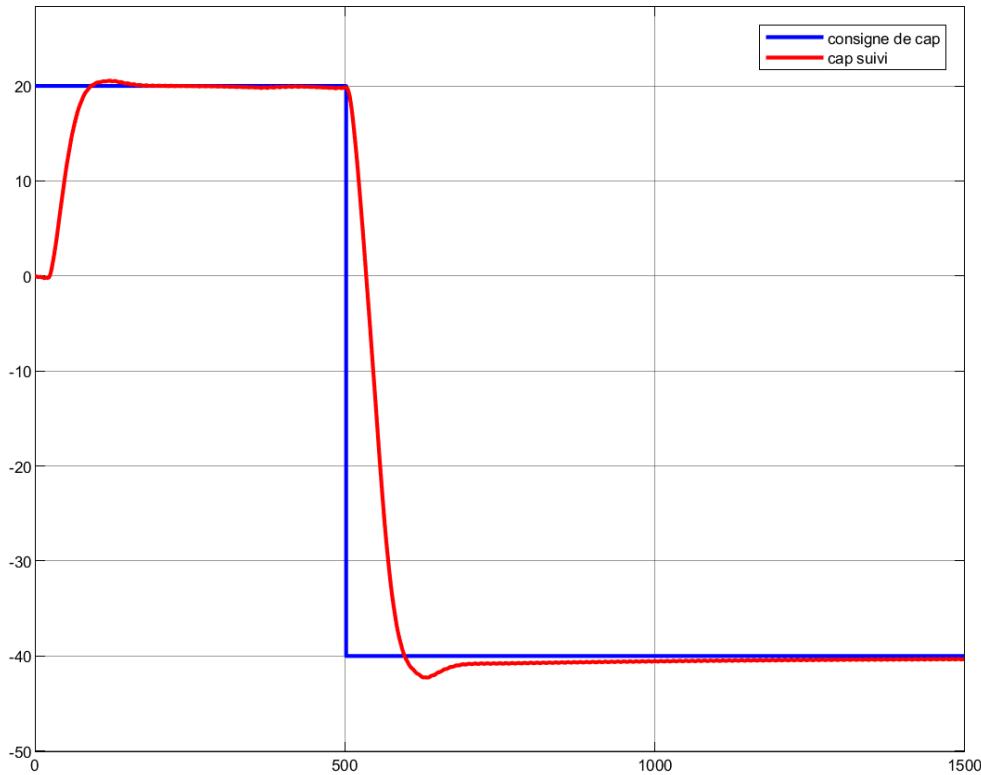


Figure 463 : résultat de la simulation

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 complexes **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.

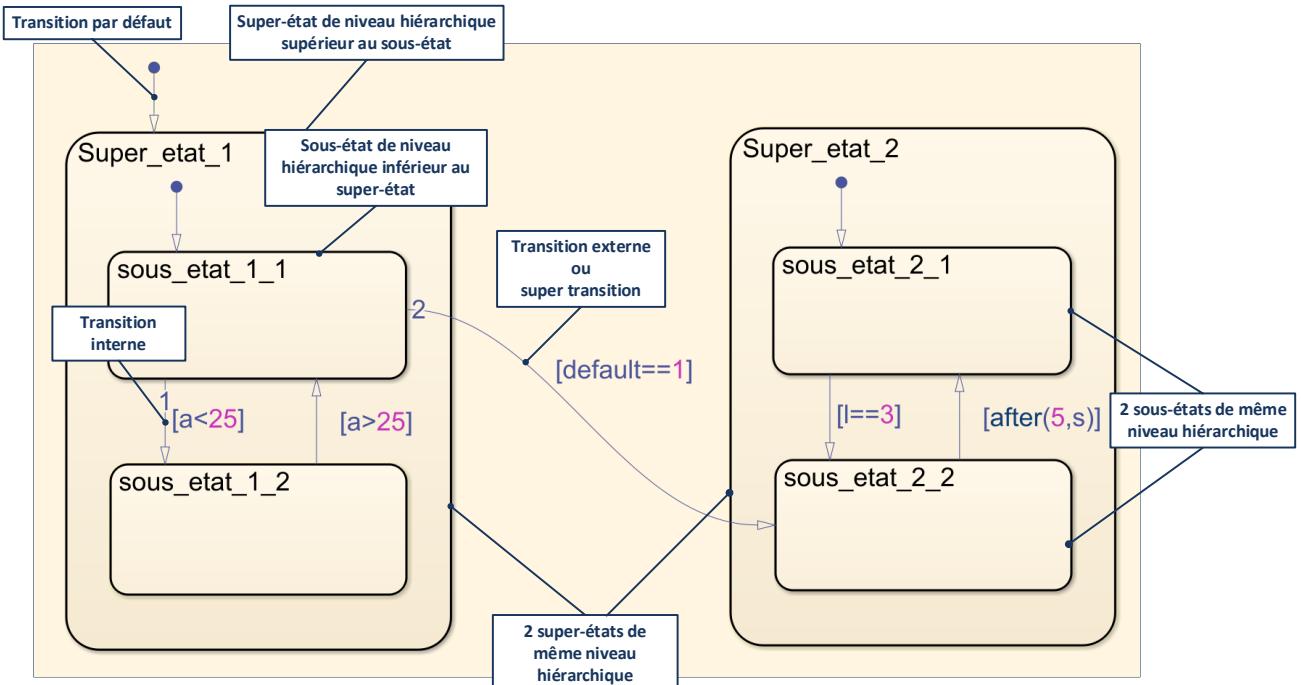


Figure 464 : représentation de super-état et de sous-état

2. Les priorités de test des transitions

Ces architectures plus complexes nécessitent 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 ou super transition** : une transition qui traverse les frontières de l'état

Règles de test des transitions :

- 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.

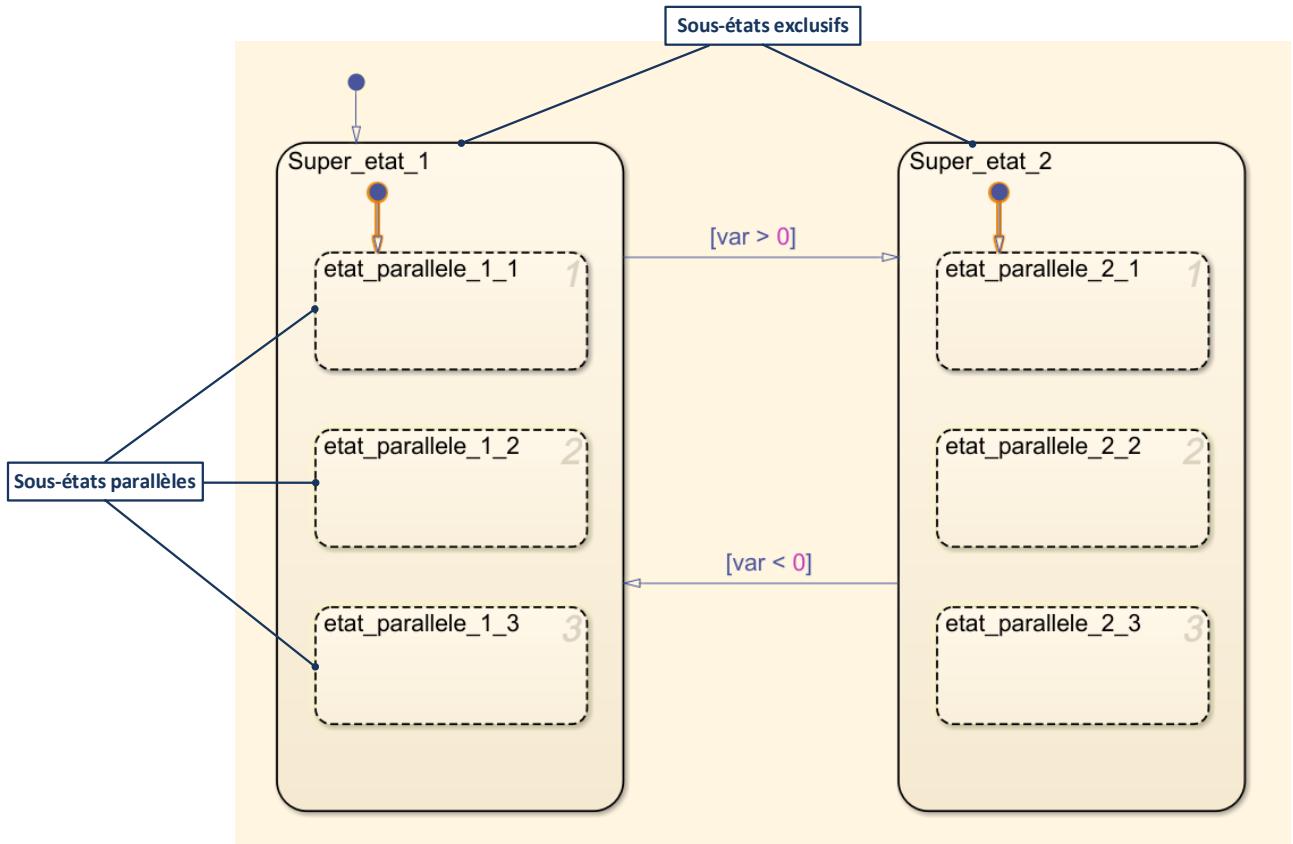


Figure 465 : représentation de sous-états parallèles

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 466. 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.

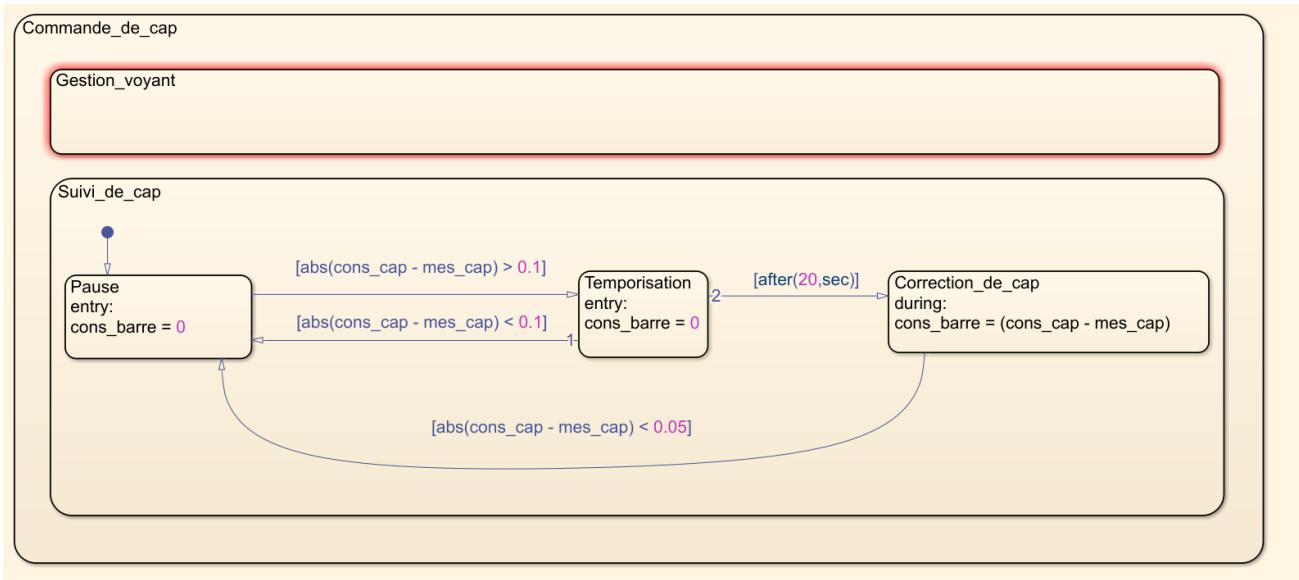


Figure 466 : création de super-états

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 lorsque 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 467). Dans le menu contextuel choisir **Decomposition/parallel**.

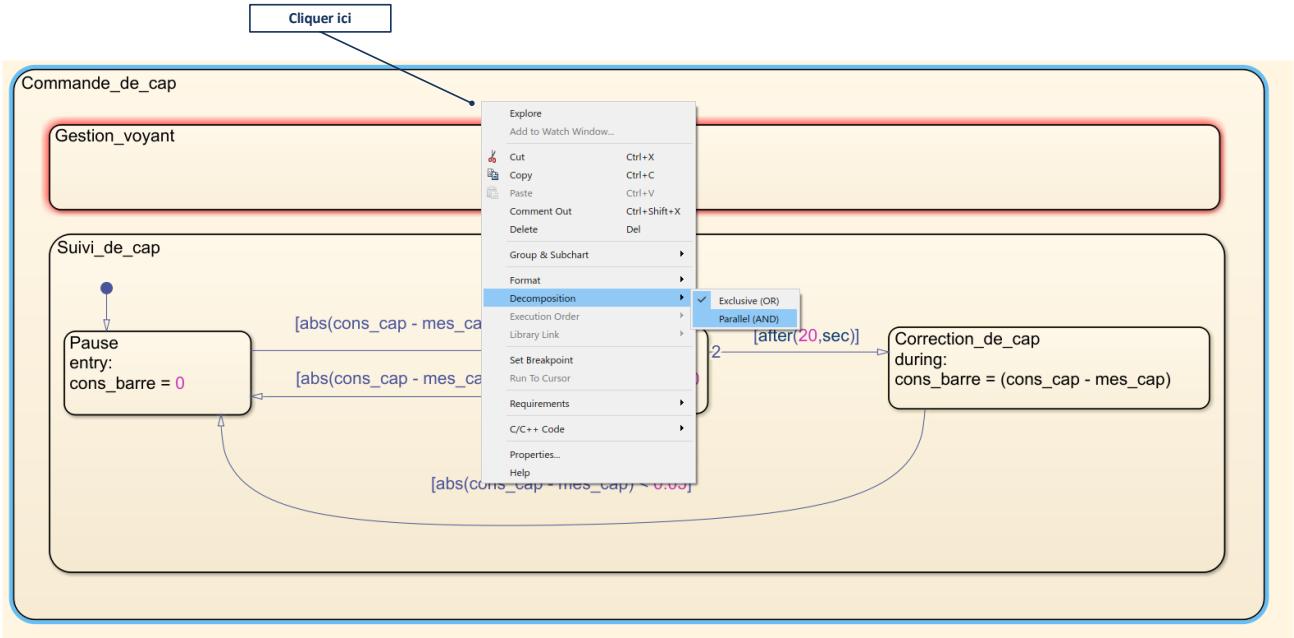


Figure 467 : créer des sous-états parallèles

Les états parallèles apparaissent maintenant en pointillés.

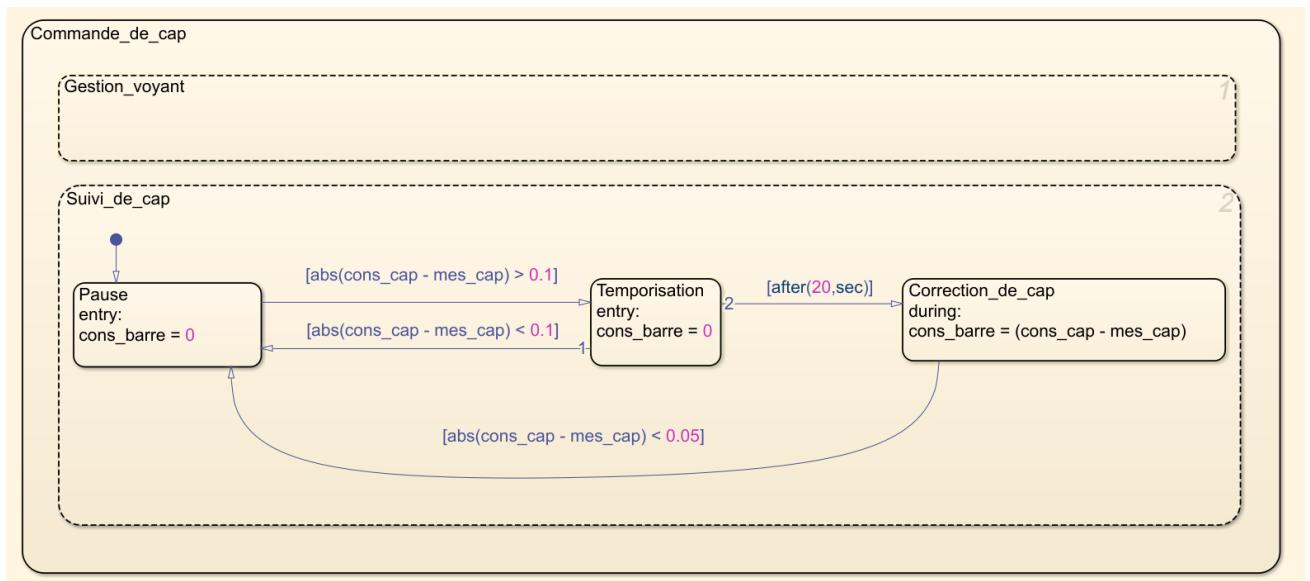


Figure 468 : représentation des états parallèles dans Stateflow

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 469.

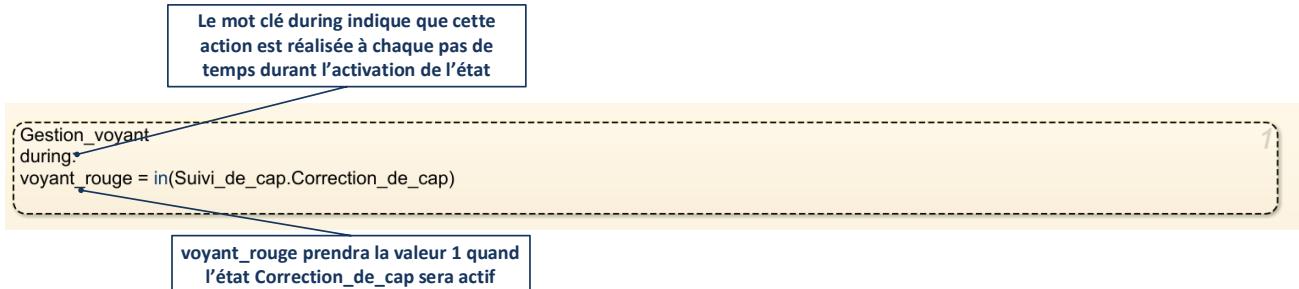


Figure 469 : utilisation de variables internes pour évaluer l'activation d'un état

En utilisant le Symbols Pan indiquer que la variable voyant_rouge est de type Output Data

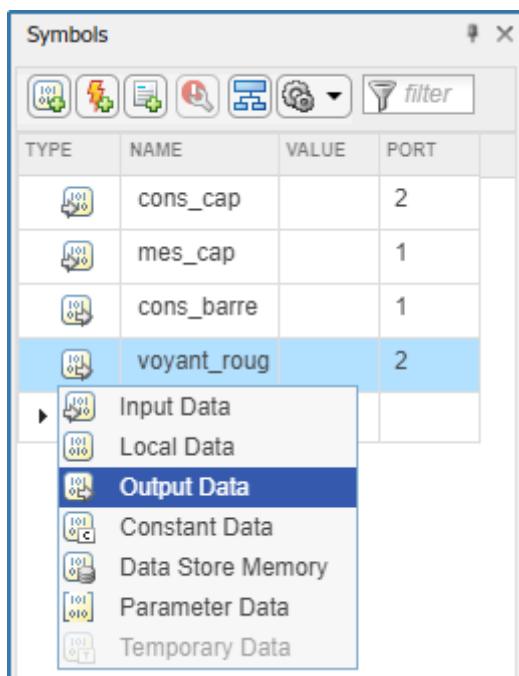


Figure 470 : paramétrage de la variable voyant_rouge

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.

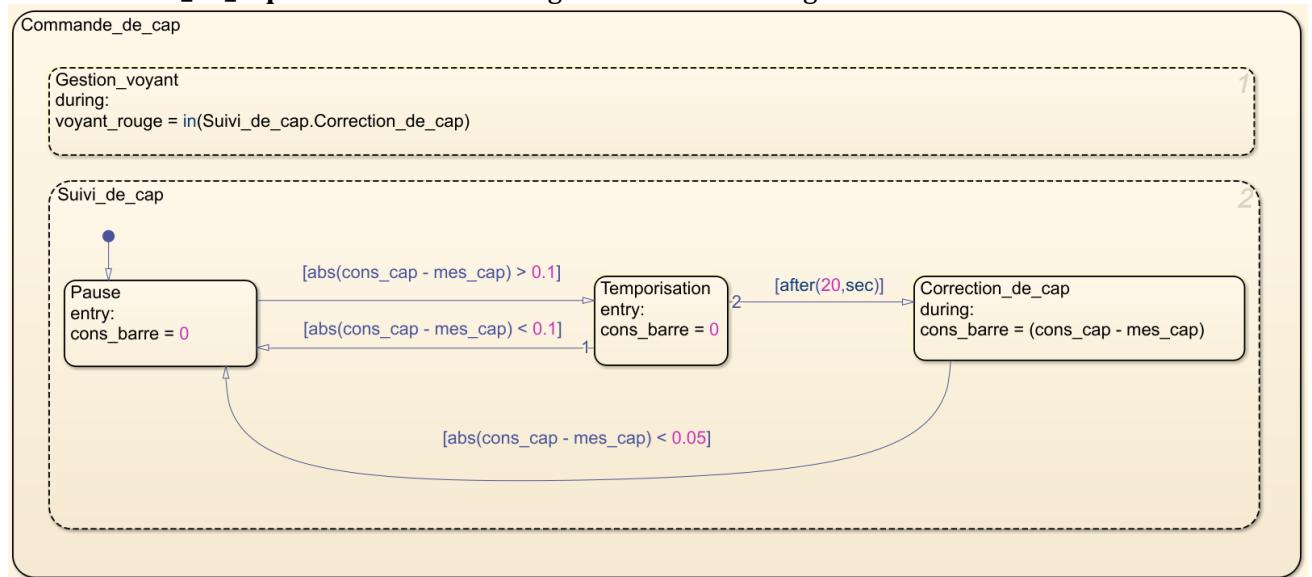


Figure 471 : chart complet de commande de cap avec super-états et états parallèles

Connecter la variable de sortie `voyant_rouge` à un scope pour la visualiser.

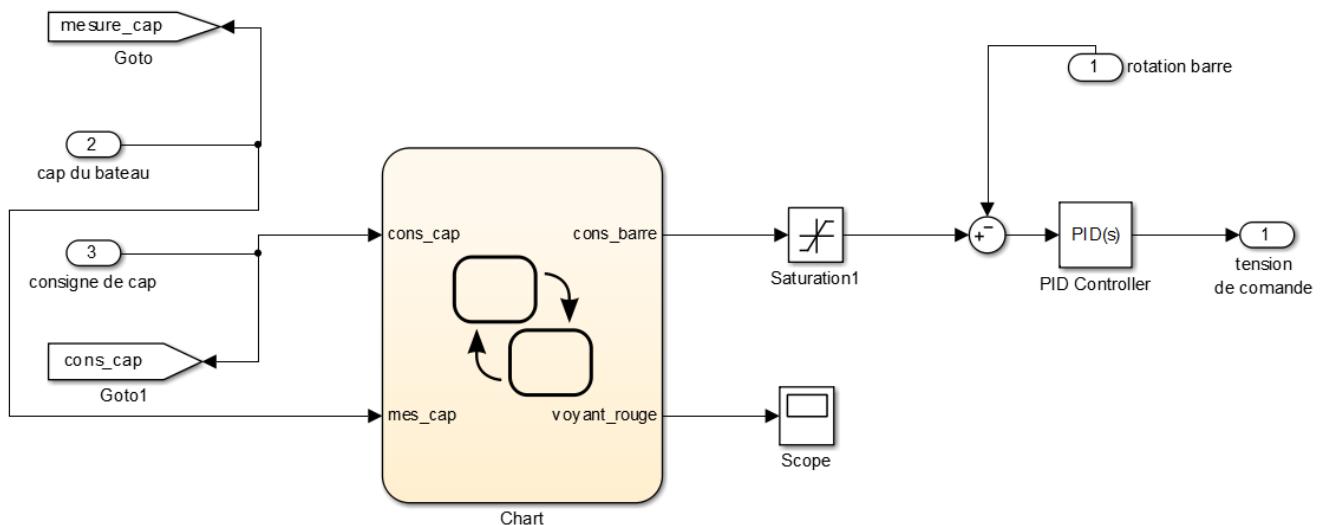


Figure 472 : connexion de la variable de sortie du chart à un scope

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.

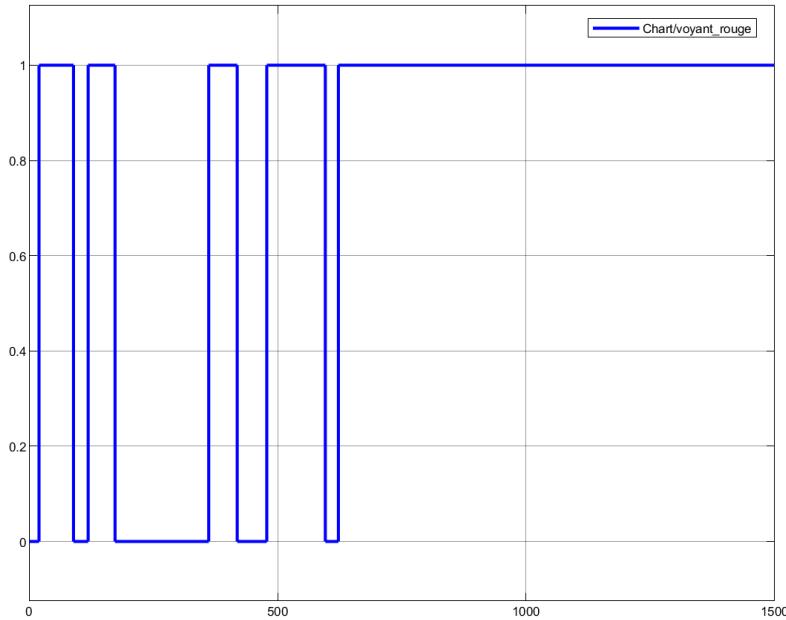


Figure 473 : visualisation de l'état d'activation du voyant

Il est possible d'ajouter un voyant de la bibliothèque Dashboard afin de visualiser l'activation de la correction de cap.

Le fichier **pilote_hydraulique_Stateflow_1_dashboard.slx** intègre cette fonctionnalité.

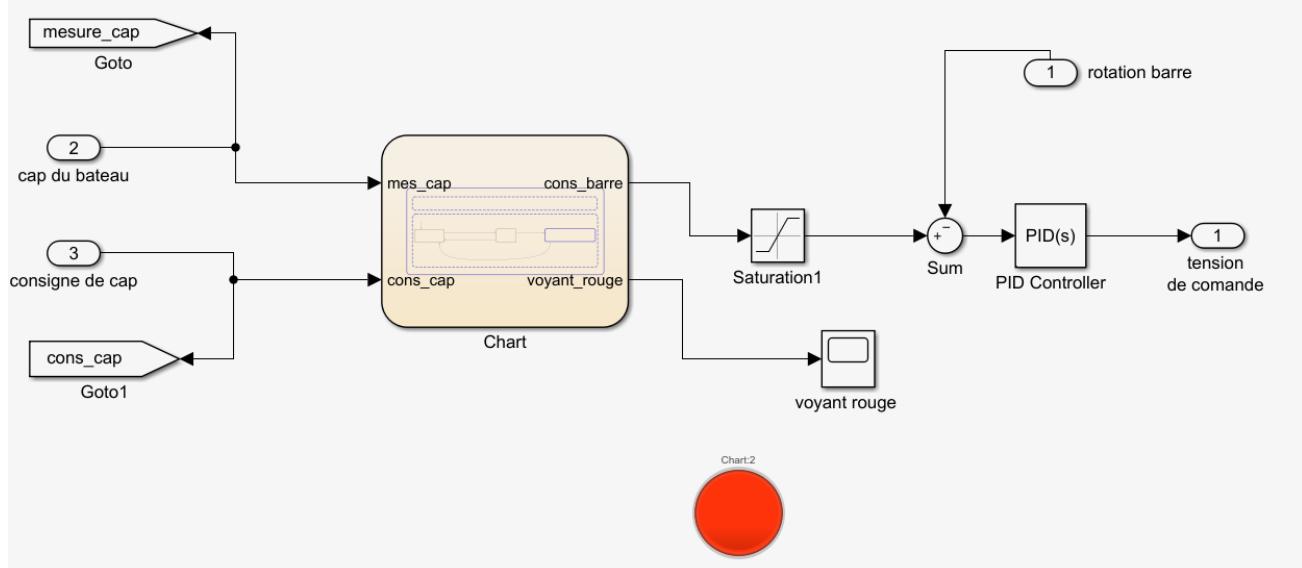
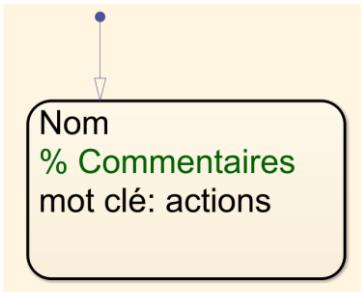


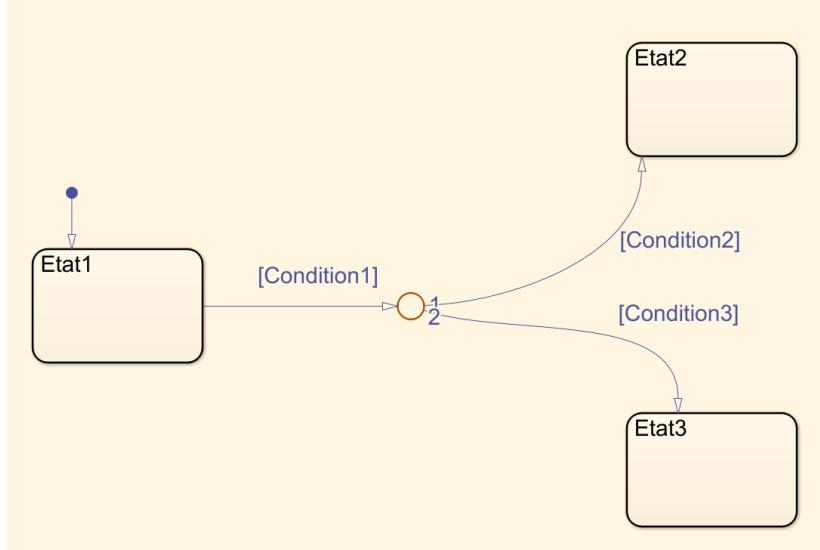
Figure 474 : ajout d'un voyant de la bibliothèque Dashboard dans le modèle

Lors de l'exécution du modèle, le voyant est vert si le pilote n'est pas en phase de correction et rouge lorsqu'il est en phase de correction.

F. Récapitulatif et complément des commandes utiles de Stateflow

Commandes utiles		
Fonctions	Commandes/syntaxe	
Création d'un état :		
	<p><i>Les mots-clés :</i></p> <p>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</p>	

Création d'une transition avec jonction:



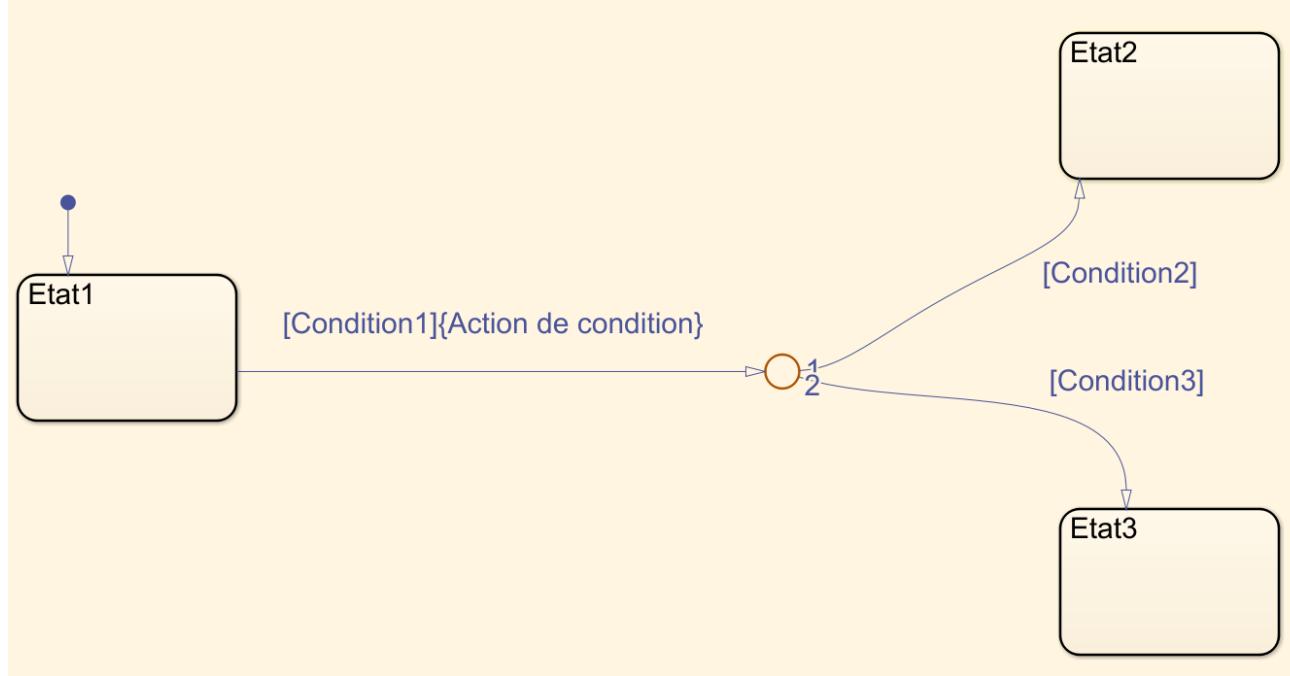
Pour créer une transition avec chemin multiples, il est possible d'utiliser une jonction disponible dans le menu « Chart ».

Pour passer à l'Etat2, il faut que **[Condition1]** et **[Condition2]** soient vraies.

Pour passer à l'Etat3, il faut que **[Condition1]** et **[Condition3]** soient vraies

Le chemin numéroté **1** est évalué avant le chemin numéroté **2**.

Création d'une transition incluant des actions :



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 ou de l'état 3.

Si **[condition]** est vrai alors **{action de condition}** s'exécute même si **[Condition2]** et **[Condition3]** sont faux.

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)]

Figure 475 : rappel des principales règles de syntaxe de Stateflow

Chapitre 8 : Prise en main de Multibody

I. Introduction à Multibody

Multibody est un outil de MATLAB permettant d'inclure dans un modèle multi-phérique, la maquette CAO 3D du système en l'important depuis Solidworks (ou tout autre logiciel de CAO). Multibody possède également des fonctionnalités d'édition de pièces qui permettent de créer des pièces volumiques directement dans l'environnement MATLAB.

Multibody 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-phérique 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.

A. Analyse d'un modèle Multibody

Dans **Multibody** les systèmes sont représentés sous la forme d'un graphe de liaisons. Les solides sont donc représentés par des blocs reliés entre eux par des liaisons.

Ouvrir le fichier « **pilote_Multibody_0.slx** » (Figure 476)

Ce fichier a été importé directement depuis le logiciel Solidworks. Les procédures d'importation seront abordées ultérieurement.

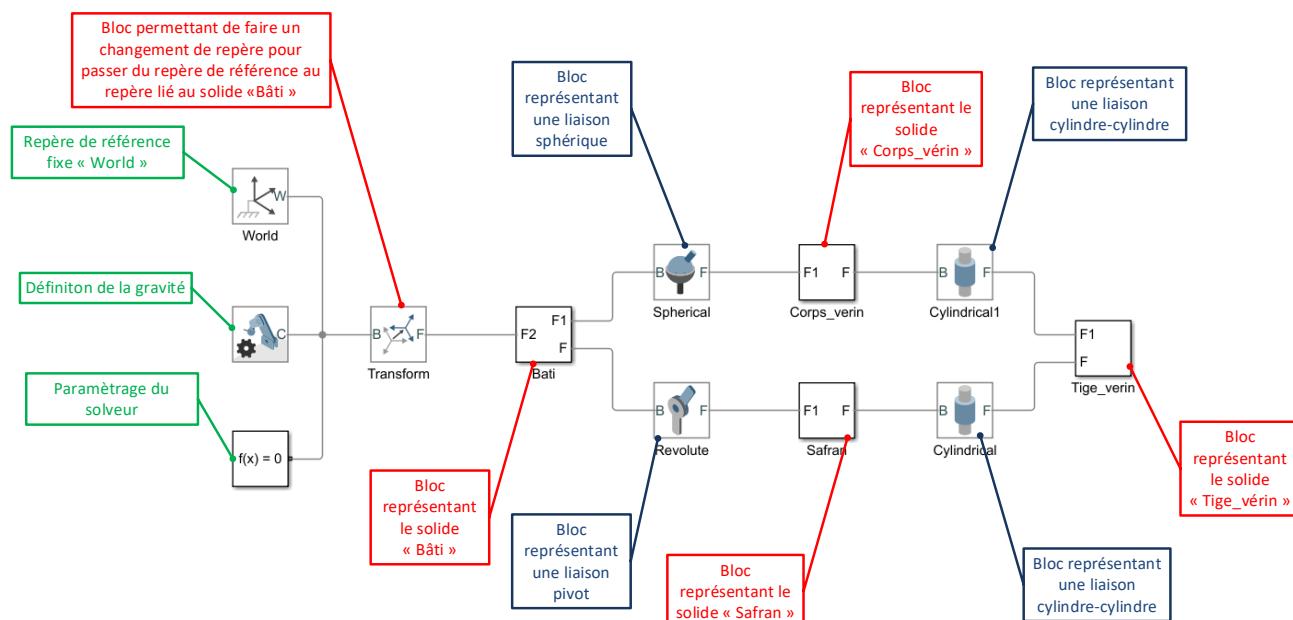


Figure 476 : modèle Multibody de la partie mécanique du pilote hydraulique

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_Multibody_0_mask.slx* »

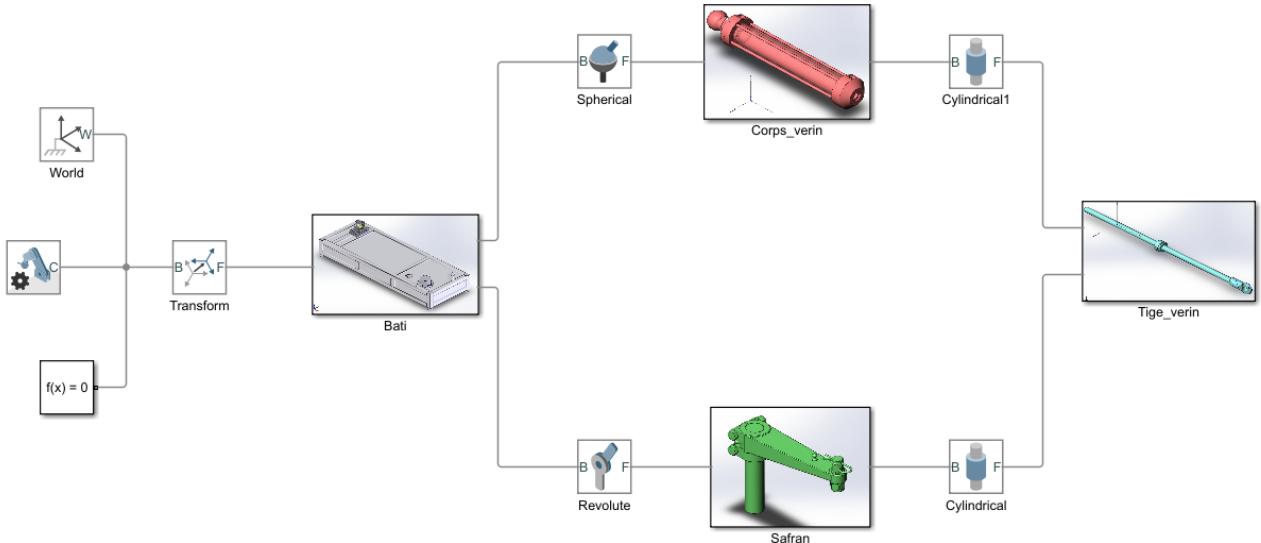


Figure 477 : modèle Multibody du pilote avec masques sur les solides

Lancer la simulation du modèle.

La fenêtre **Mechanics Explorers** apparaît et la maquette 3D de la partie mécanique du pilote hydraulique est visualisable ().

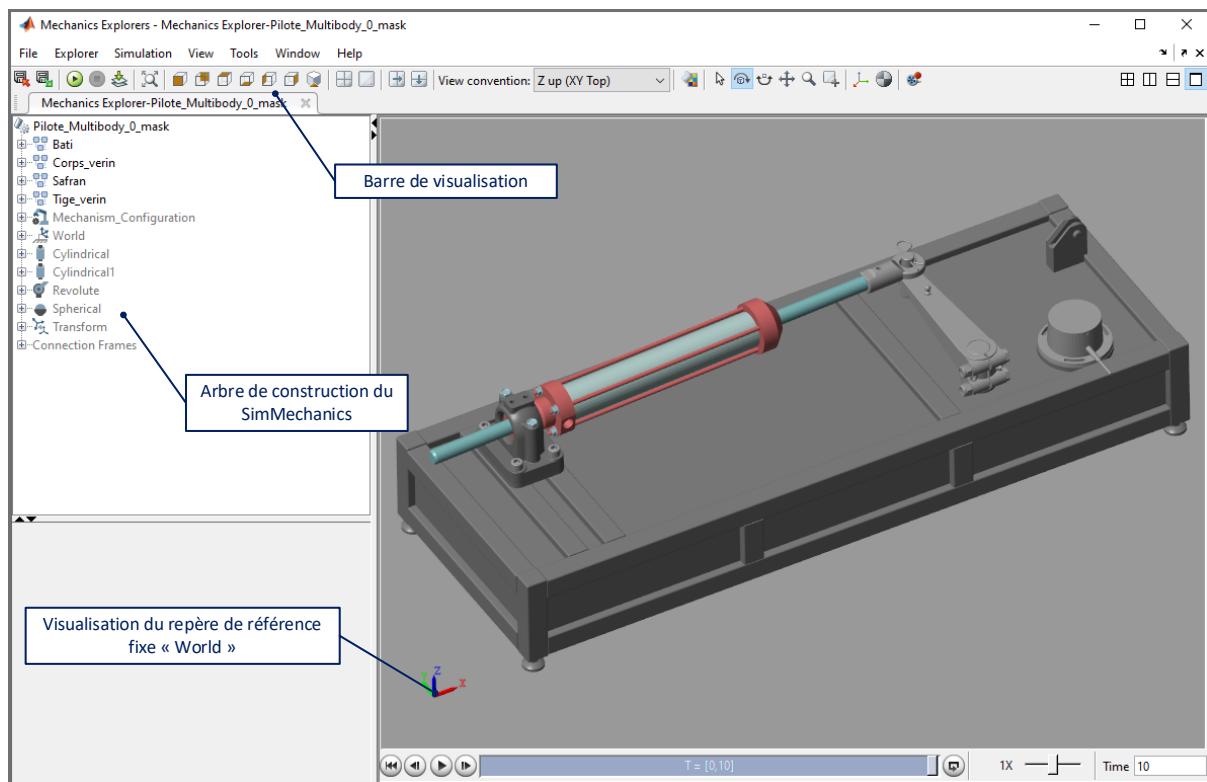


Figure 478 : fenêtre Mechanics Explorers de la partie mécanique du pilote hydraulique

La barre d'outils de Multibody permet différentes options d'affichage (Figure 479).

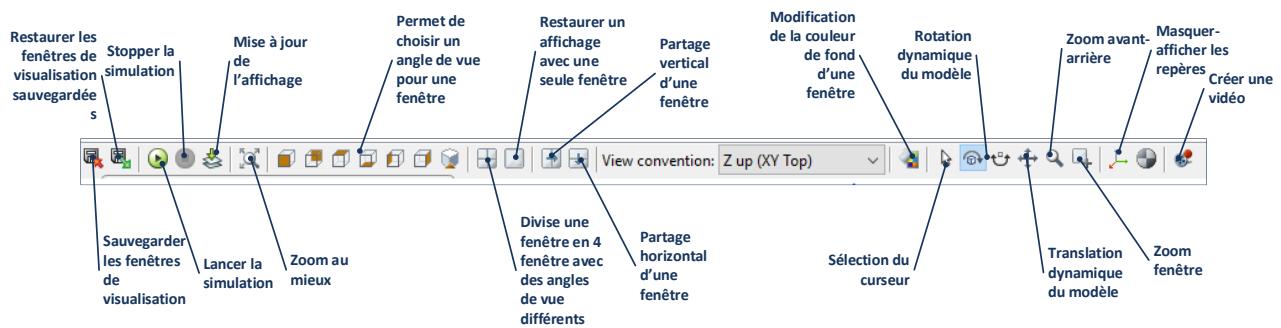


Figure 479 : options de la barre de visualisation de Multibody

A noter qu'aucun mouvement n'est observable pour le moement 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 480).

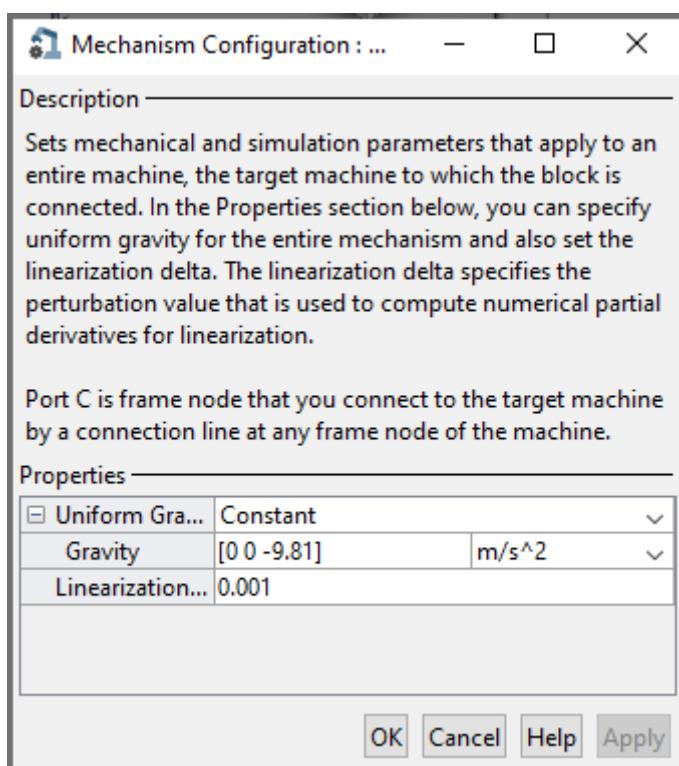


Figure 480 : paramétrage de la pesanteur dans Multibody

Par défaut la pesanteur a été placée sur l'axe z (valeur -9.81 m/s^2), 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 481).

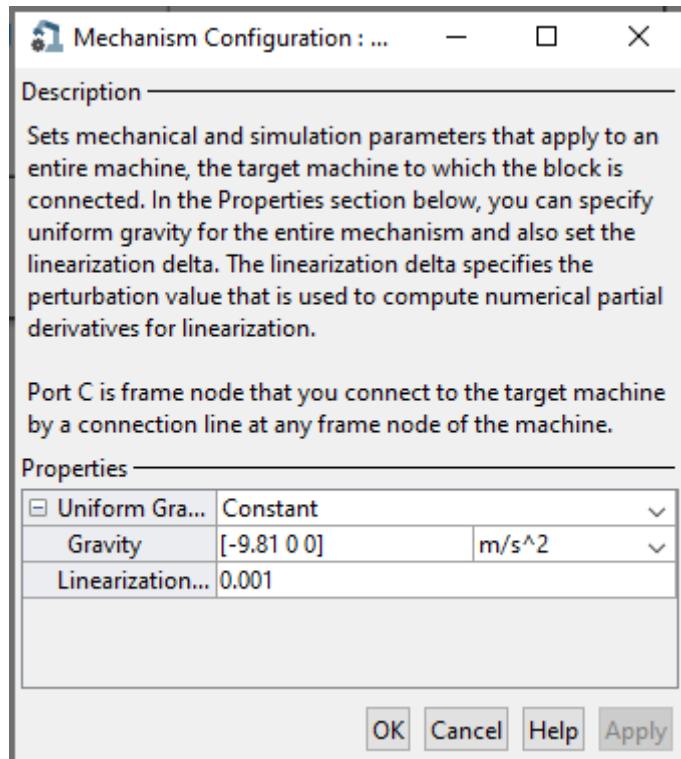


Figure 481 : modification de l'action de la pesanteur

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 Multibody dans un modèle multi-physique

Nous allons apprendre dans cette partie à intégrer un modèle Multibody dans un modèle multi-physique. Pour cela nous utiliserons le modèle du pilote hydraulique de bateau.

Ouvrir le fichier **pilote_hydraulique_Multibody_start.slx**

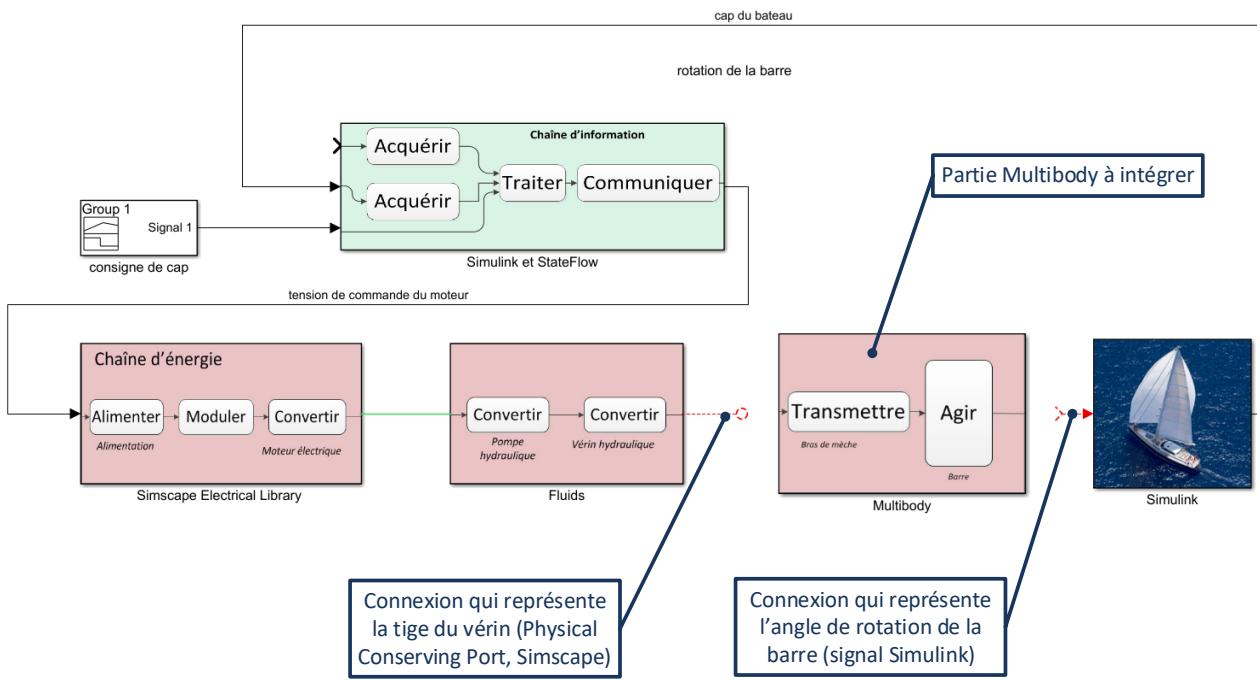


Figure 482 : modèle du pilote hydraulique avec modèle Multibody à intégrer

Le sous-système **Multibody** contient le modèle de la partie mécanique du pilote, mais qui n'est pas connecté avec le reste du modèle.

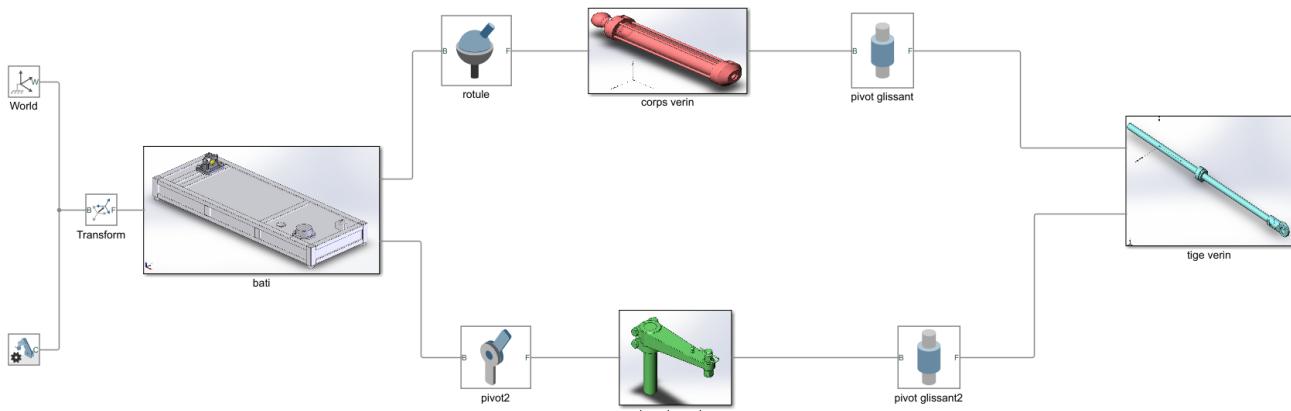


Figure 483 : visualisation du système Multibody sans connexion avec le reste du modèle

Sur la Figure 482, il est possible de visualiser les connexions que le sous-système **Multibody** 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 **Multibody** 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. Cette information prend la forme d'un signal Simulink.

A. Connexions du modèle

Afin de connecter le modèle, il faut créer à l'intérieur du sous-système **Multibody** deux ports :

- Un port Simscape pour se connecter à la tige du vérin
- Un port Simulink pour renvoyer l'angle 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 les renommer conformément à la Figure 484.

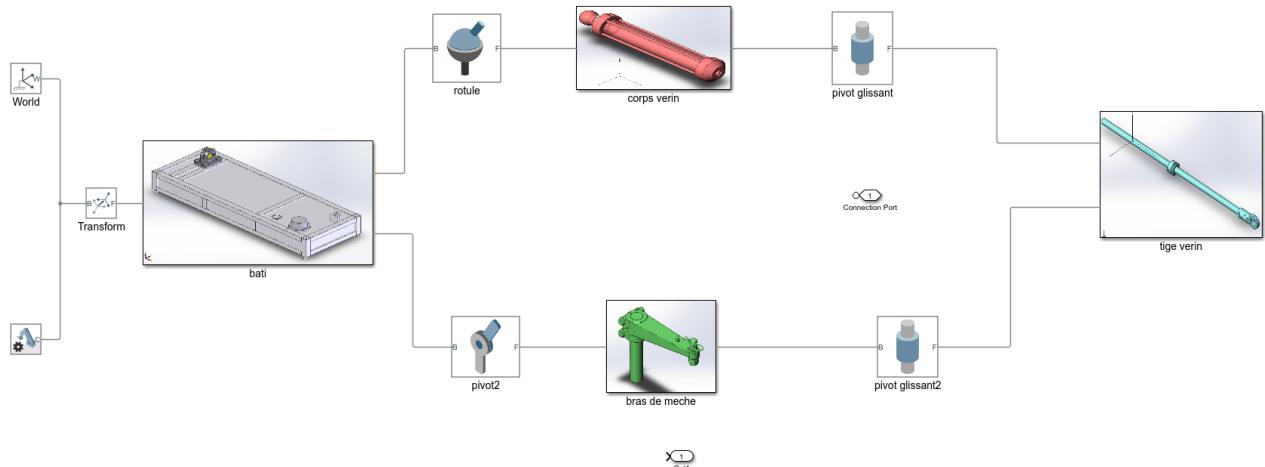


Figure 484 : placement des ports de connexion dans le modèle

Des ports apparaissent sur le sous-système Multibody. **Connecter** ces ports avec le modèle.

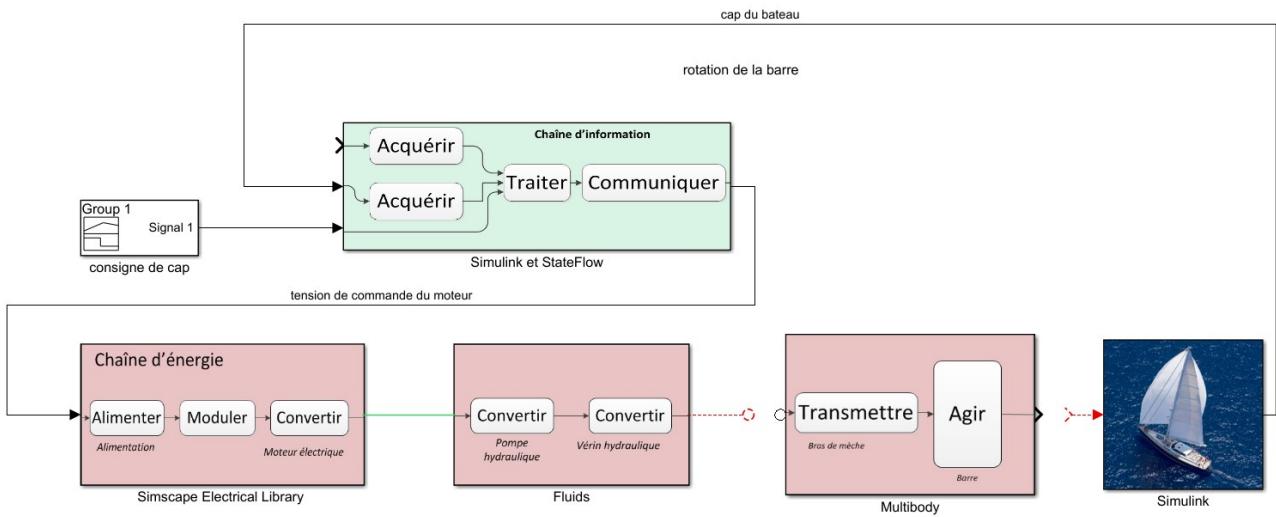


Figure 485 : connexions des ports du sous-système avec le modèle

B. Interfaçage entre Simscape et Multibody

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 **Multibody**, 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 **Multibody**, il est impératif de réinjecter cette vitesse dans le modèle **Simscape**. Cela garantira durant tout le fonctionnement une équivalence de comportement entre **Simscape** et **Multibody**.

1. Interfaçage entre Simscape et Multibody pour la translation

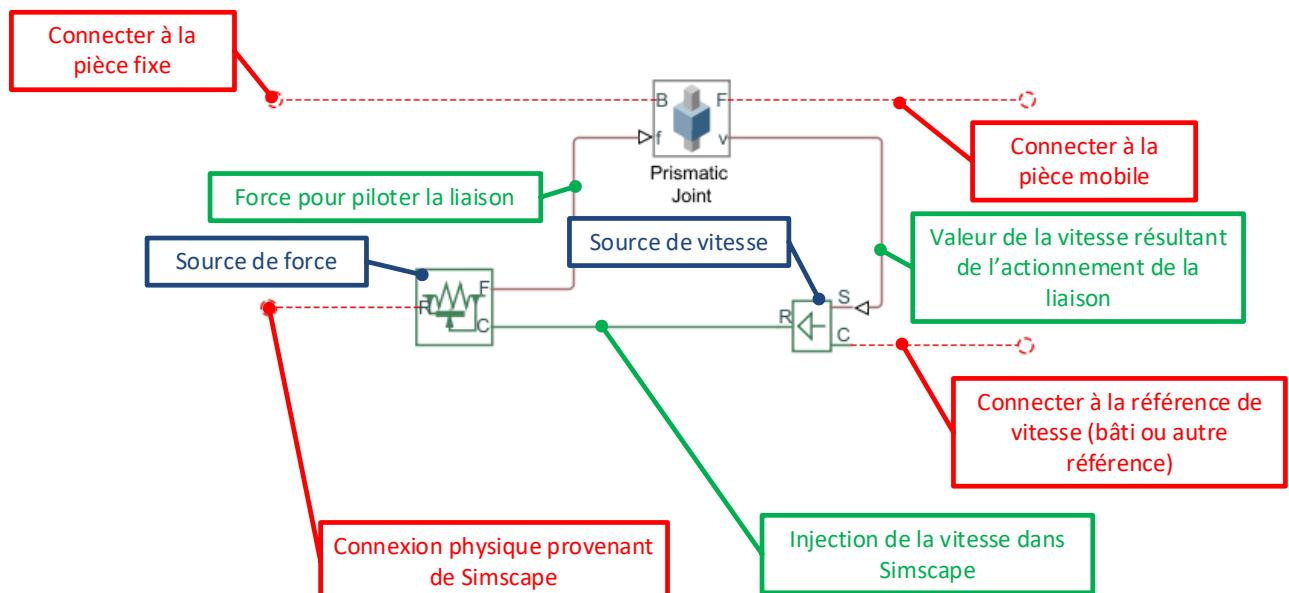


Figure 486 : interface de translation entre Simscape et Multibody

2. Interfaçage entre Simscape et Multibody pour la rotation

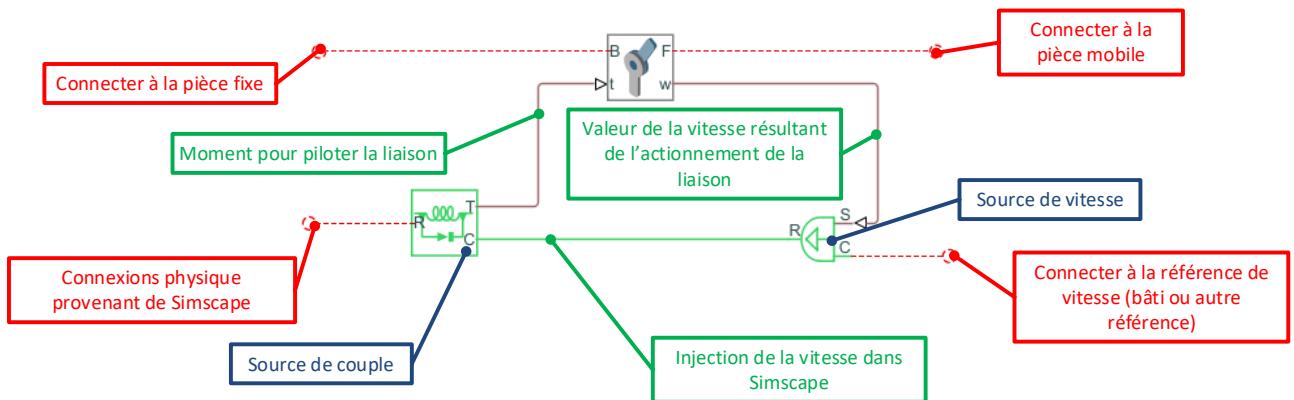


Figure 487 : interfaçage de rotation entre Simscape et Multibody

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

Ce fichier contient les interfaces entre **Simscape** et **Multibody**. Il est également possible de mettre ces interfaces sous la forme de sous-systèmes afin de les réutiliser facilement.

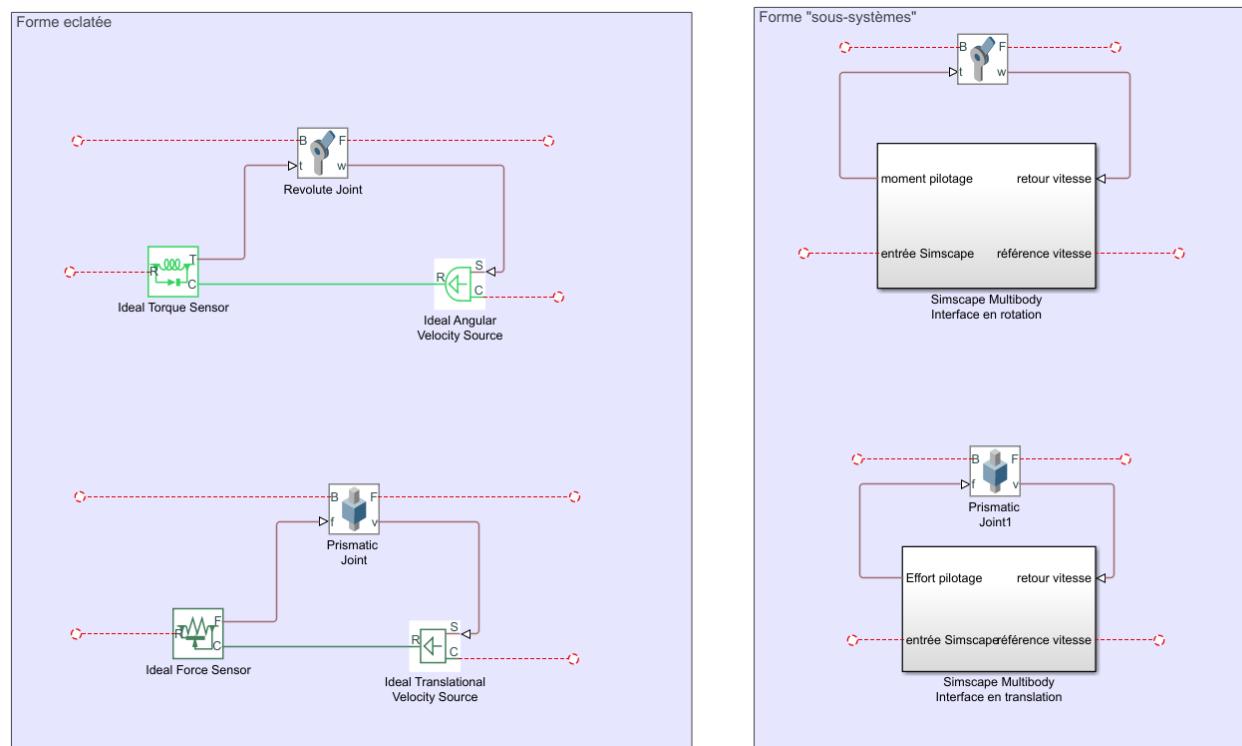


Figure 488 : les interfaces entre Simscape et Multibody

Copier le bloc **Simscape Multibody Interface** en translation dans le modèle.

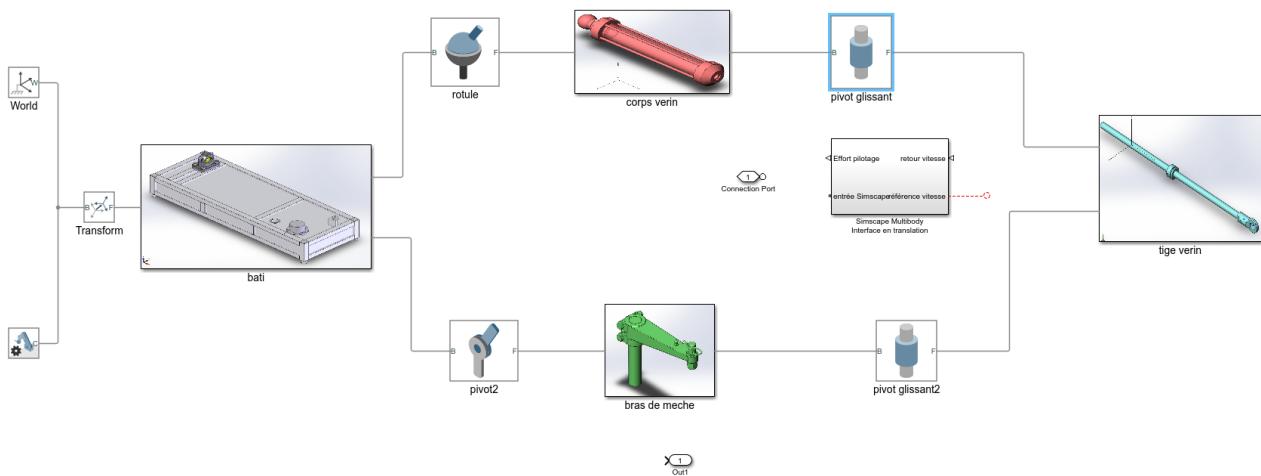


Figure 489 : ajout du bloc Simscape Multibody Interface

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 boîte de dialogue du paramétrage d'une liaison et paramétrer la liaison conformément à la Figure 490 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.

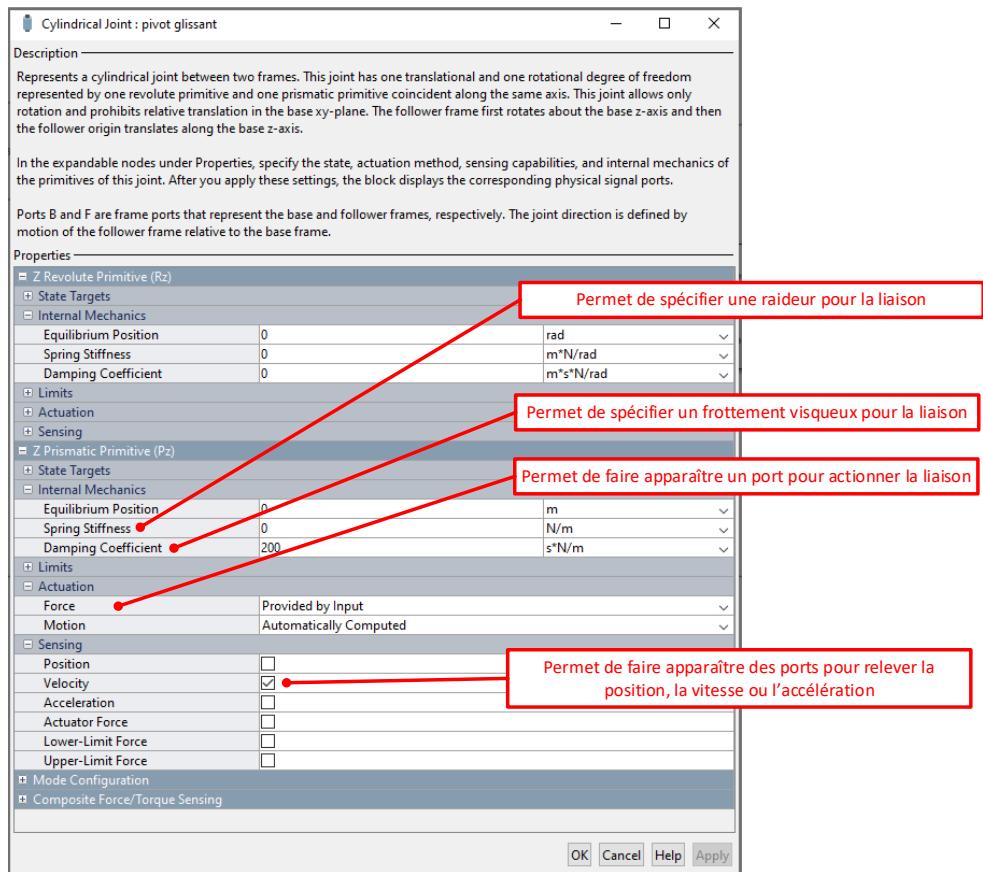


Figure 490 : paramétrage des ports d'une liaison

Des ports apparaissent sur le bloc de la liaison. **Connecter** l'interface conformément à la Figure 491.

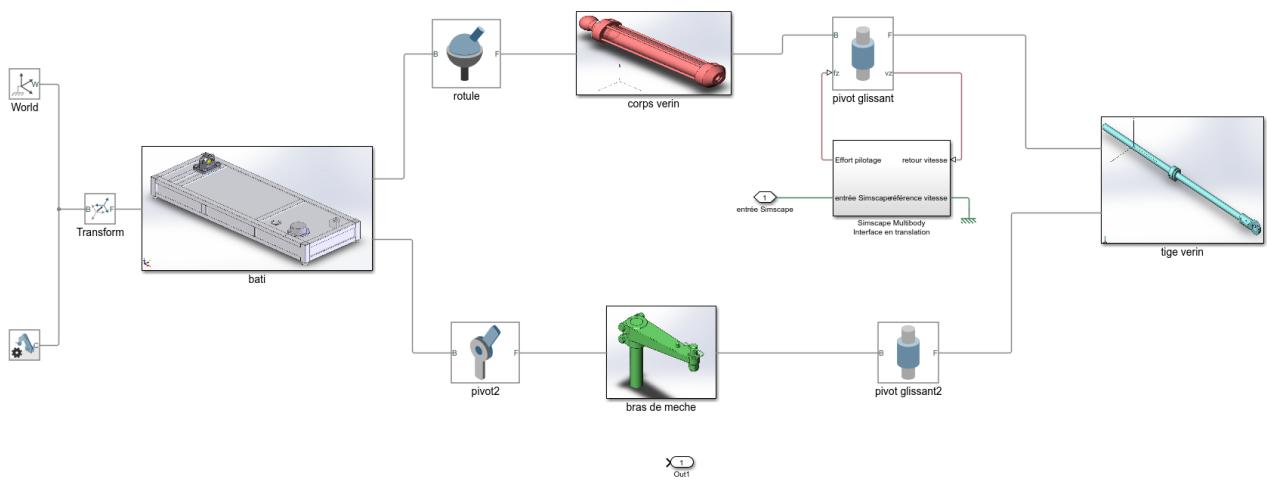


Figure 491 : connexion entre Simscape et Multibody

L'interface entre Simscape et Multibody 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 492.

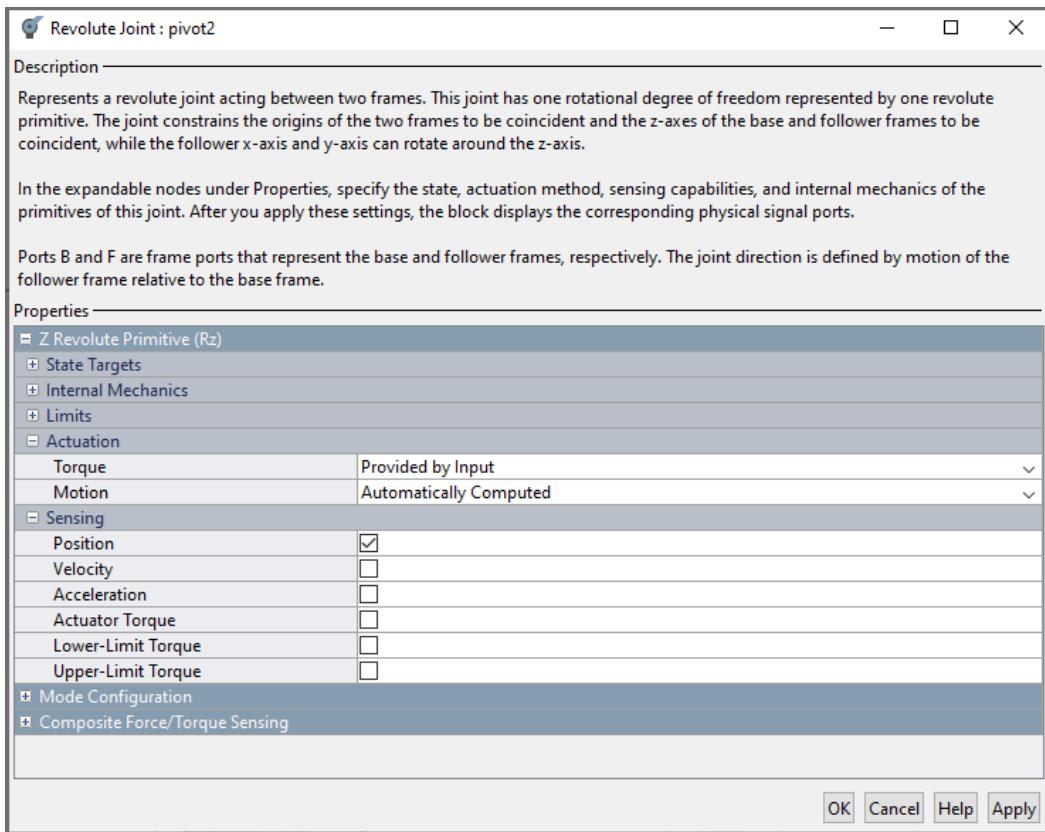


Figure 492 : ajout d'un port pour imposer un couple à la liaison

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.

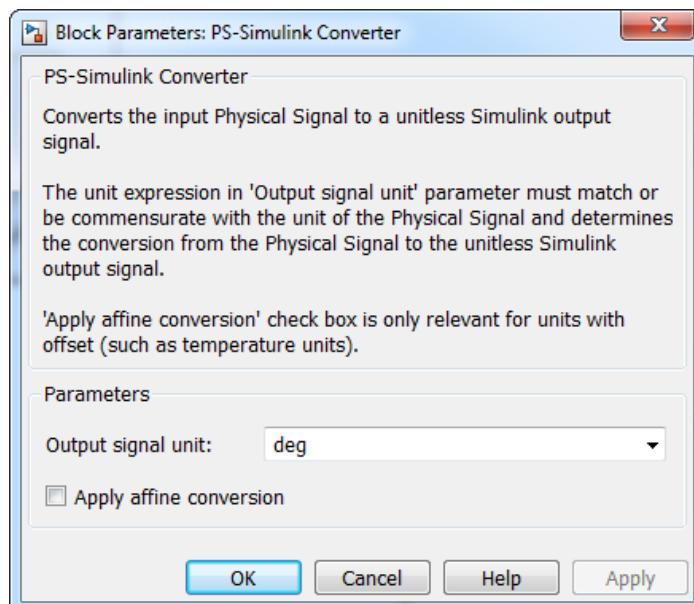


Figure 493 : conversion du signal physique en signal Simulink

Vous devez obtenir la configuration de la Figure 494.

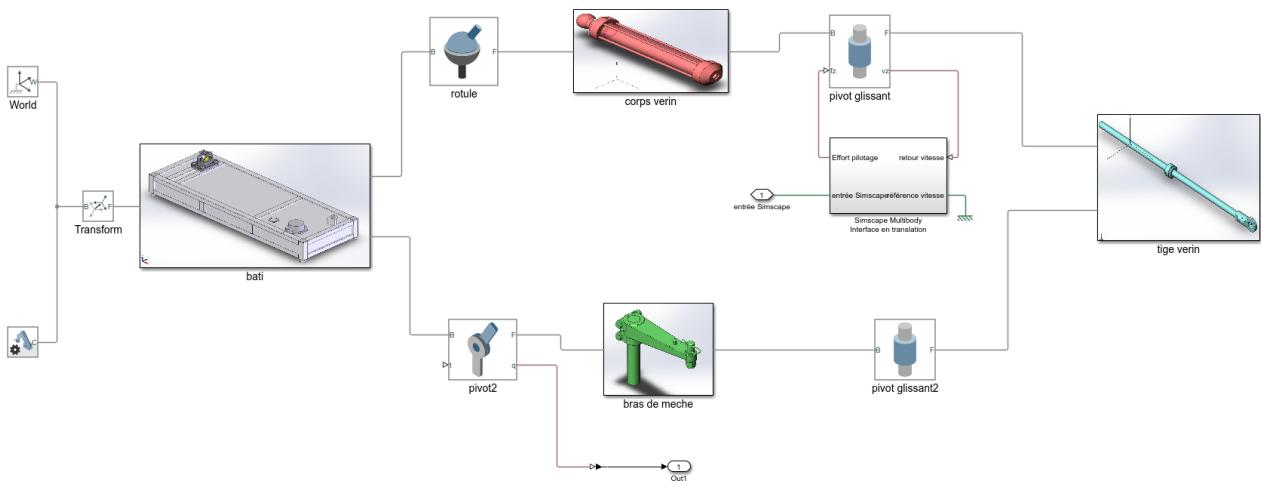


Figure 494 : ajout de port sur une liaison dans Multibody

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**.

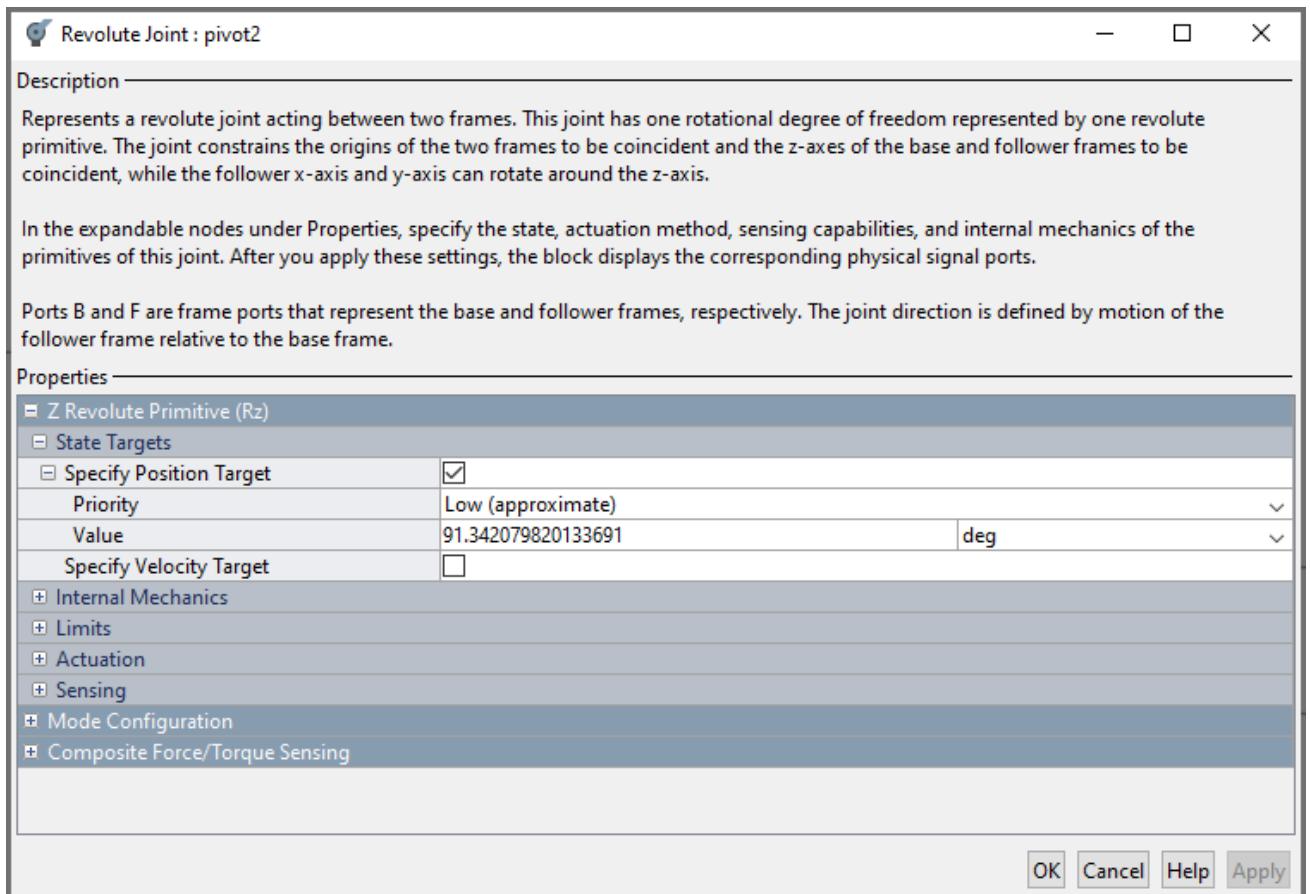


Figure 495 : visualisation de l'offset de l'angle de barre

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 496 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.

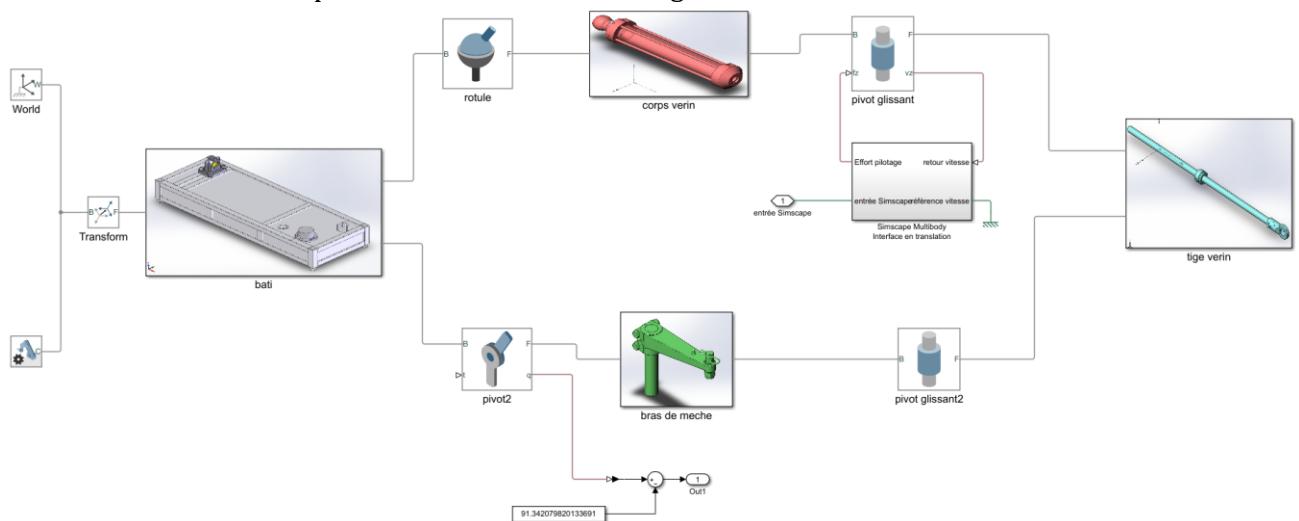


Figure 496 : compensation de l'offset de l'angle de la barre

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_Multibody_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.

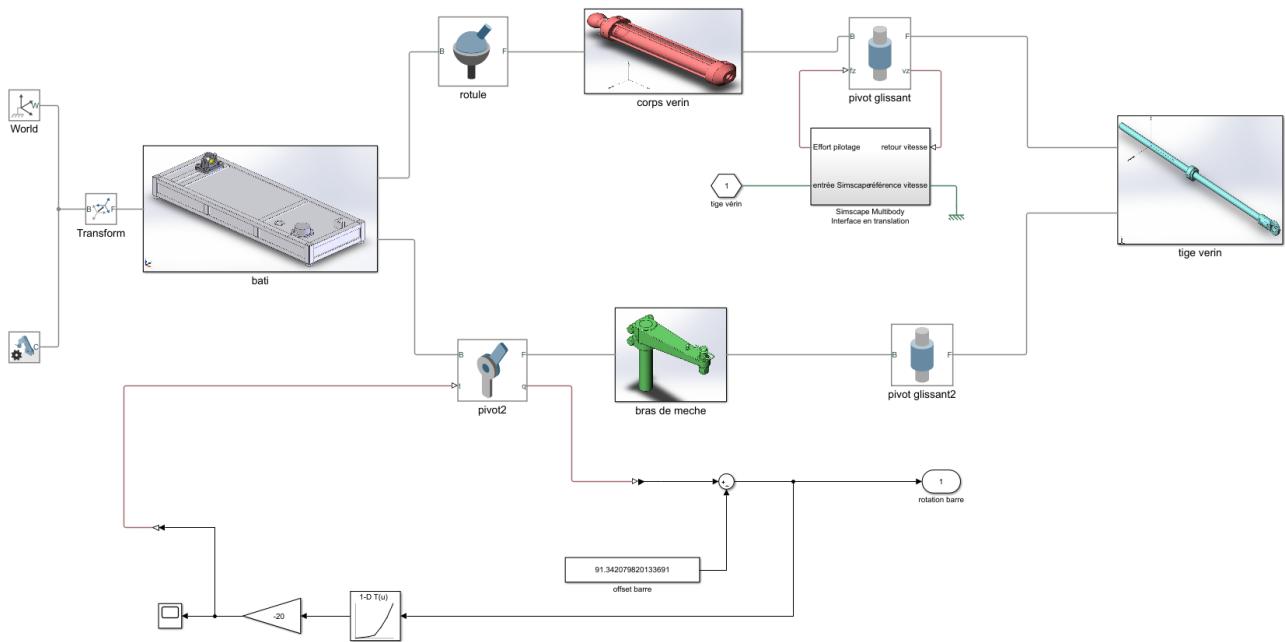


Figure 497 : ajout d'un effort sur la liaison, utilisation d'une Lookup Table

Double cliquer sur le bloc **1D Look-up Table** pour explorer le paramétrage de ce type de bloc.

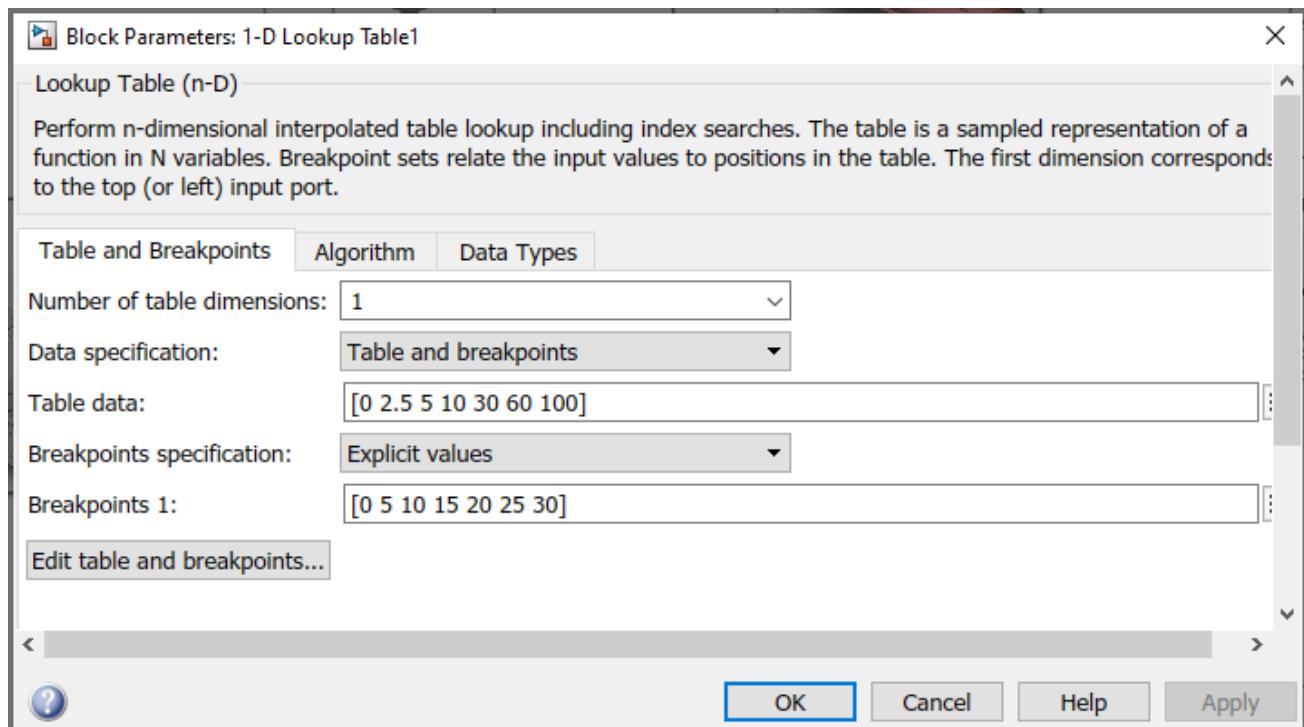


Figure 498 : paramétrage d'une Lookup Table

Pour paramétriser 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**.

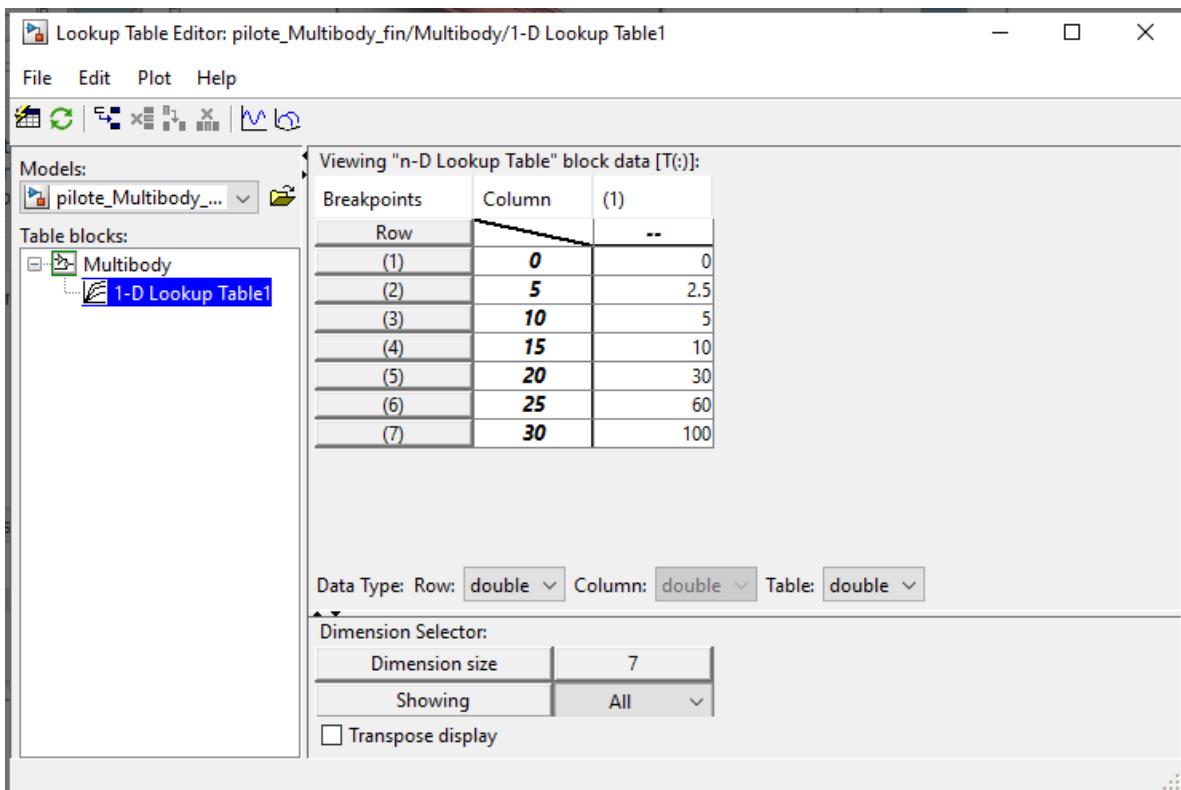


Figure 499 : visualisation d'une Lookup Table

Il est possible de visualiser la courbe représentant la loi entrée sortie en cliquant sur **Linear Plot** .

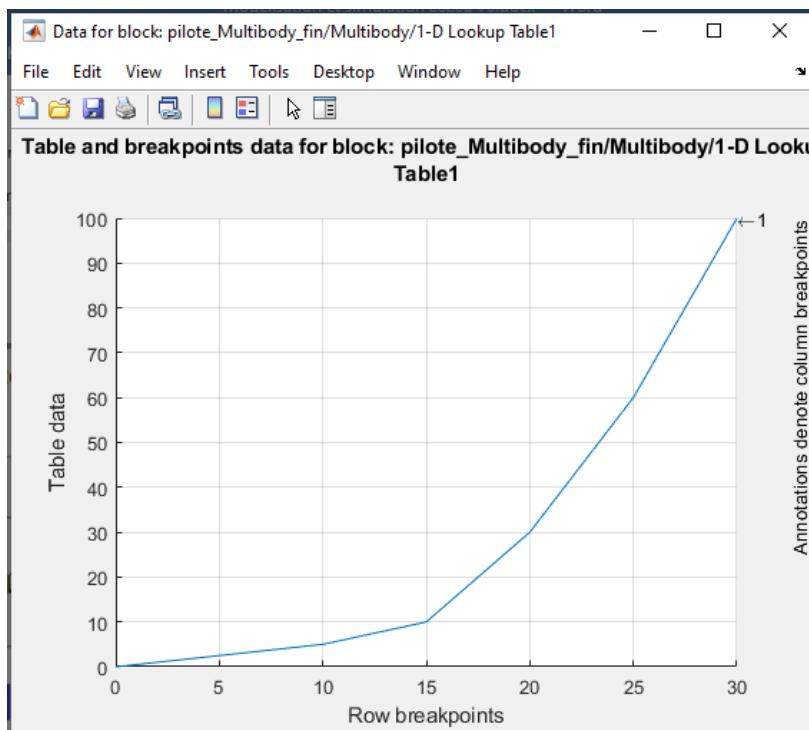


Figure 500 : visualisation de la courbe représentative de la loi entrée sortie d'une Lookup Table

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 **Multibody** ajouté.

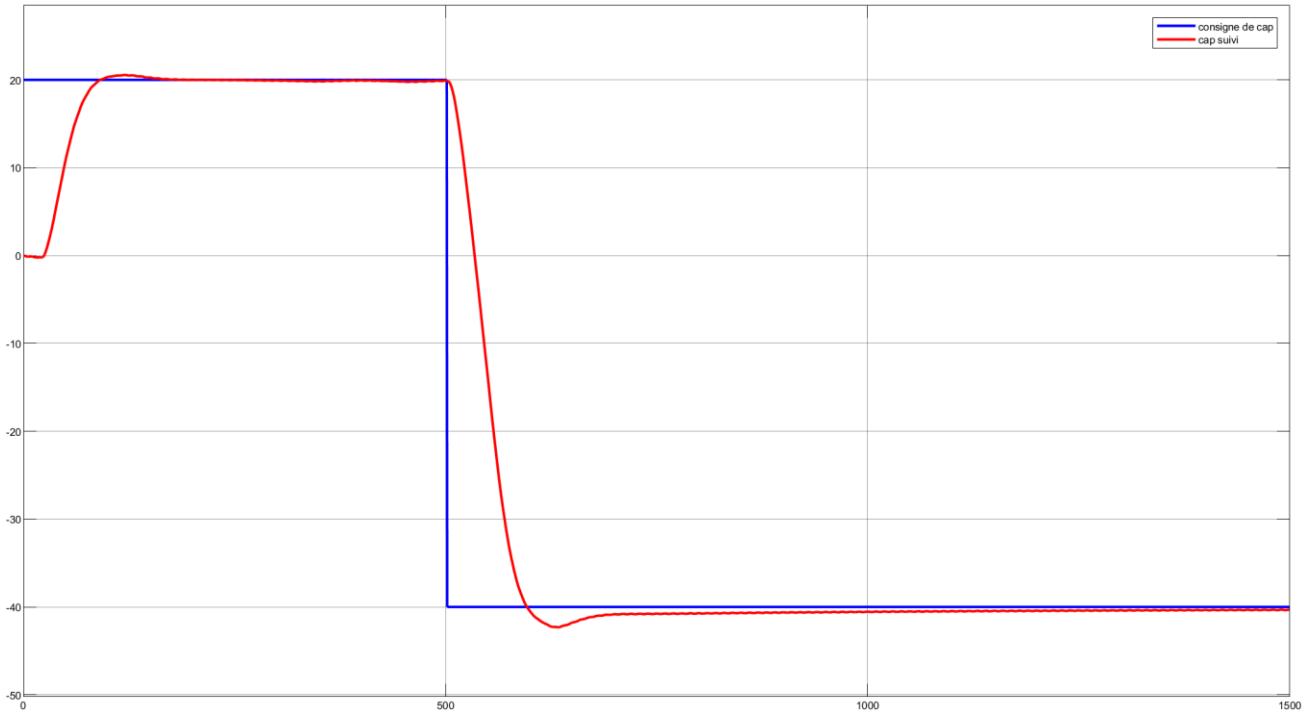


Figure 501 : résultat de la simulation du cap suivi par le bateau

III. Importation d'un modèle SolidWorks dans Multibody

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 « **Multibody 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 Multibody.

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 « Multibody Link »

Pour effectuer le téléchargement de **Multibody 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 :

https://www.mathworks.com/campaigns/offers/download_smlink.html

Vous êtes alors sur la page permettant de télécharger les fichiers d'installation nécessaires (Figure 502)

 Download and installation instructions

[Expand all](#)

▼ Simscape Multibody Link 7.2 – Release 2020b (Simscape Multibody 7.2)

Simscape Multibody 7.2	
Win64 (PC) Platform	smlink.r2020b.win64.zip install_addon.m
UNIX (64-bit Linux)	smlink.r2020b.glnxa64.zip install_addon.m
Mac OS X (64-bit Intel)	smlink.r2020b.maci64.zip install_addon.m

➤ Simscape Multibody Link 7.1 – Release 2020a (Simscape Multibody 7.1)

➤ Simscape Multibody Link 7.0 – Release 2019b (Simscape Multibody 7.0)

➤ Simscape Multibody Link 6.1 – Release 2019a (Simscape Multibody 6.1)

➤ Simscape Multibody Link 6.0 – Release 2018b (Simscape Multibody 6.0)

Figure 502 : choix de la version de Multibody Link

Il faut choisir la version de **Multibody 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.r2020b.win64.zip**

Dans la fenêtre de commande de MATLAB taper la commande qui lance l'installation de Multibody Link :

```
>> installAddon('smlink.r2020b.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 **installAddon** doit être exactement celui que vous avez téléchargé et dépend donc de votre version de MATLAB et de votre système d'exploitation.

Il faut ensuite enregistrer MATLAB comme serveur Automation en tapant la commande suivante :

```
>> regmatlabserver
```

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

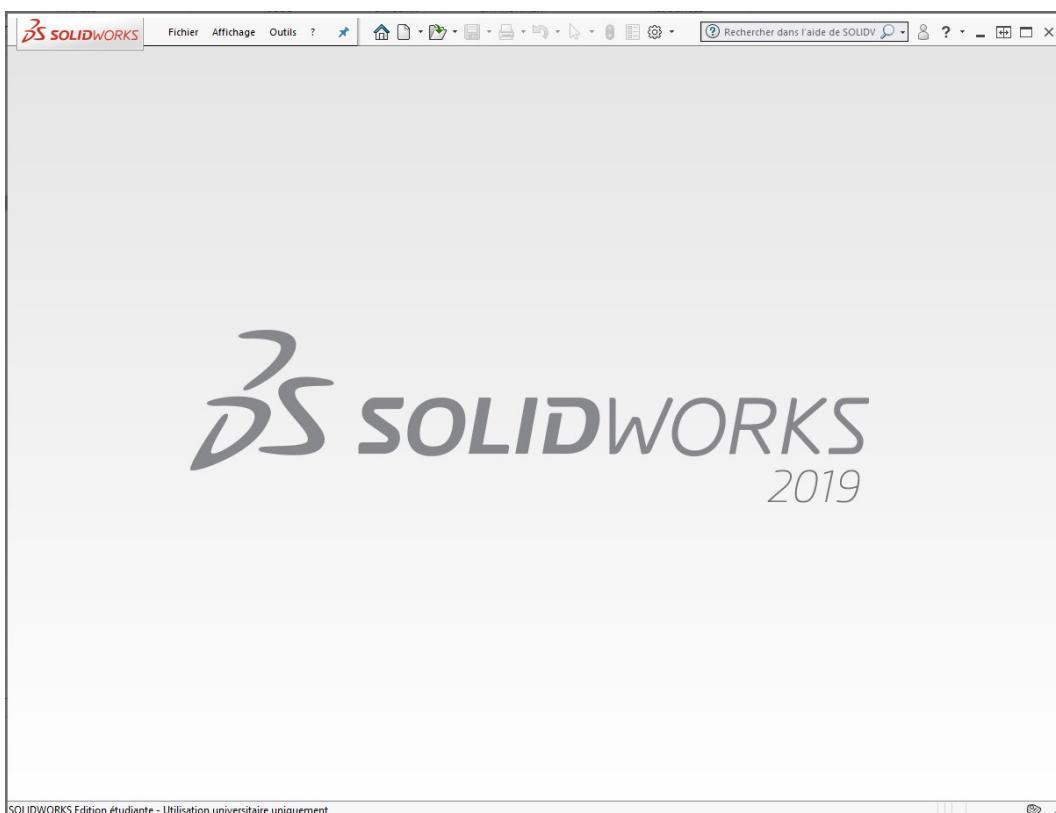


Figure 503 : fenêtre d'accueil de Solidworks

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.

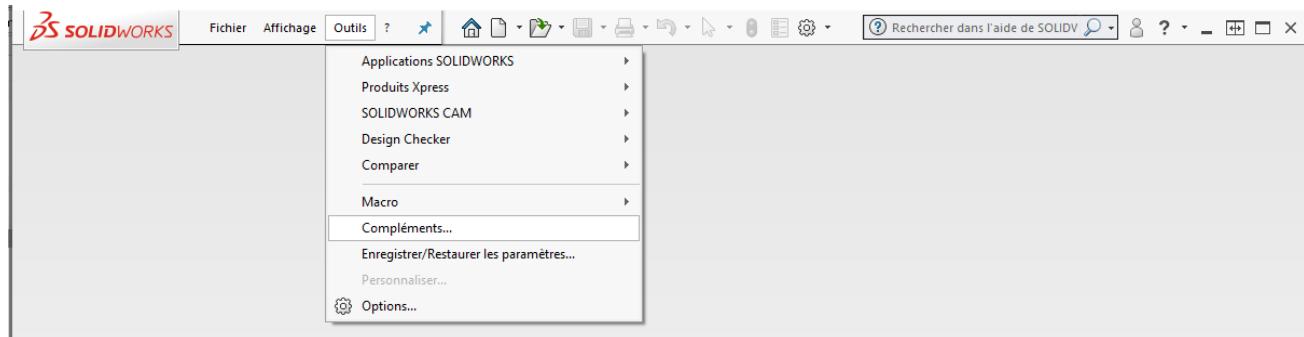


Figure 504 : ouverture de la fenêtre des compléments de Solidworks

Cocher les deux cases à gauche et à droite de **Simscape Multibody Link** afin d'activer **Simscape Multibody Link** à chaque démarrage de Solidworks (Figure 505).

Cliquez sur **OK**.

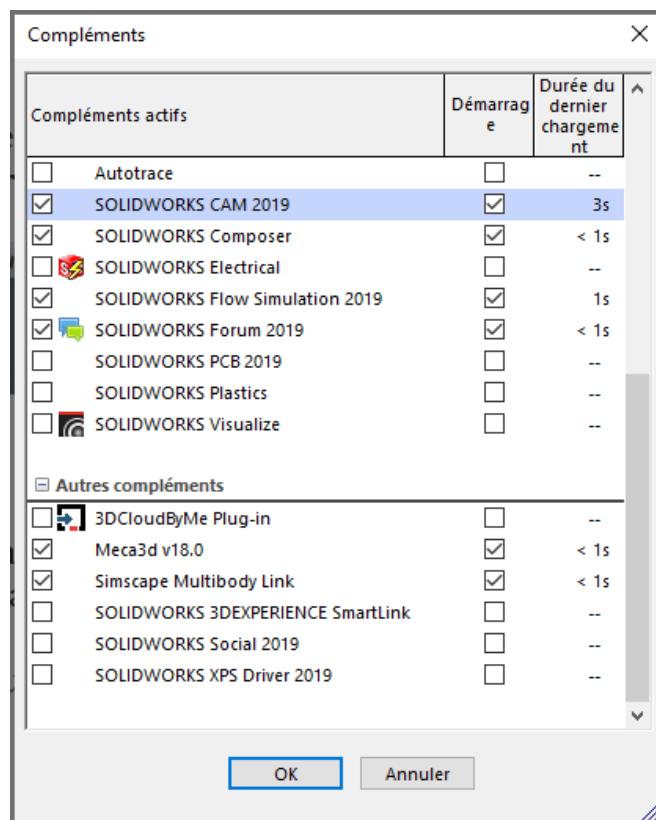


Figure 505 : sélection des compléments de Solidworks

Quitter, puis **relancer** Solidworks.

Avec Solidworks, **Ouvrir** le fichier **Assemblage pilote hydraulique.sldasm** dans le dossier « Modèle CAO 3D du pilote/Partie Mécanique »

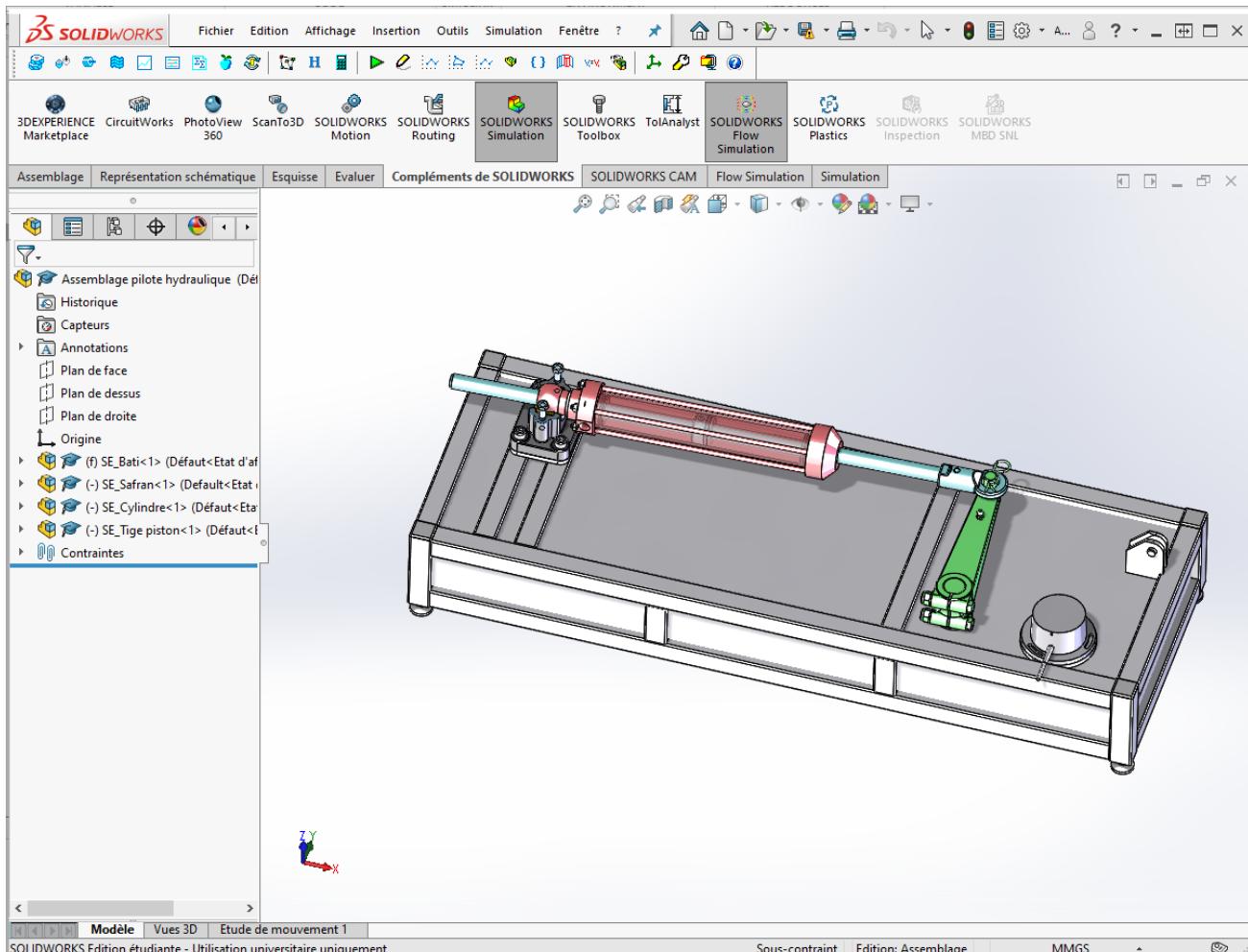


Figure 506 : ouverture du modèle Solidworks de la partie mécanique du pilote

Pour convertir l'assemblage au format **xml**, cliquer dans la barre de menu sur **Outils/Simscape Multibody Link/Export/Simscape Multibody**

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**.

Après quelques secondes de travail, Solidworks va ouvrir et fermer tous les fichiers pièces qui constituent l'assemblage pour les convertir au format STL. Une fenêtre qui s'ouvre automatiquement indique la fin de la conversion.

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 **Multibody** correspondant à l'assemblage et ouvre automatiquement un nouveau fichier contenant le modèle.

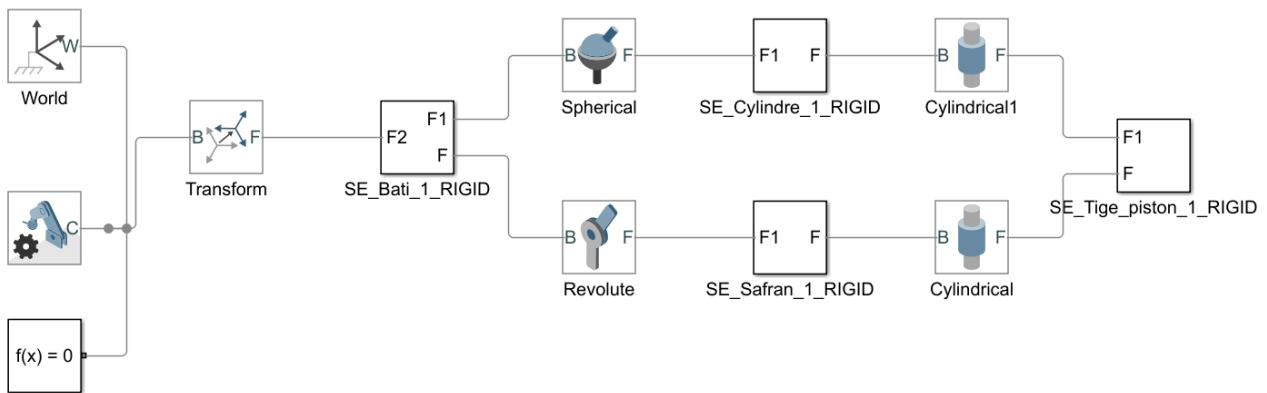


Figure 507 : modèle Multibody obtenu

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.

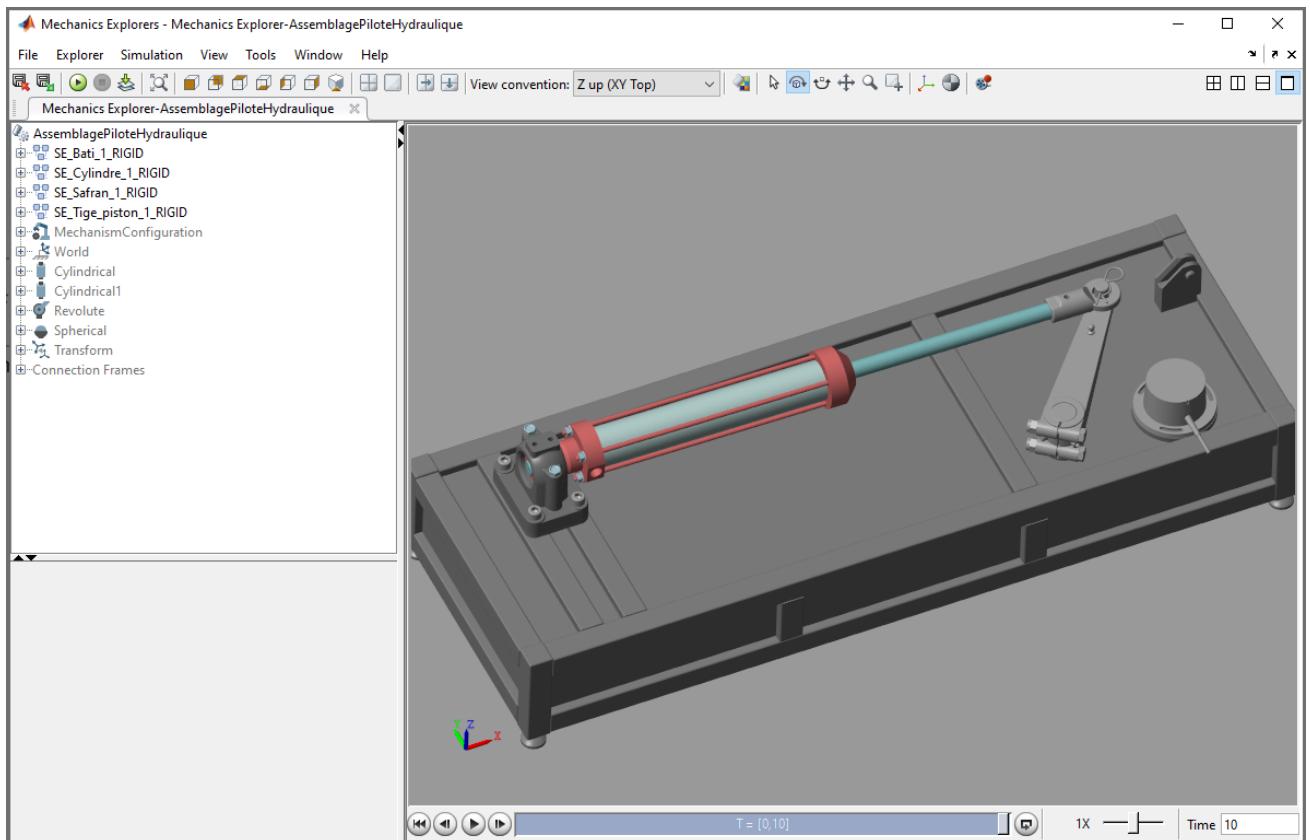


Figure 508 : fenêtre Mechanics Explorer

Chapitre 9 : 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 boîte 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.

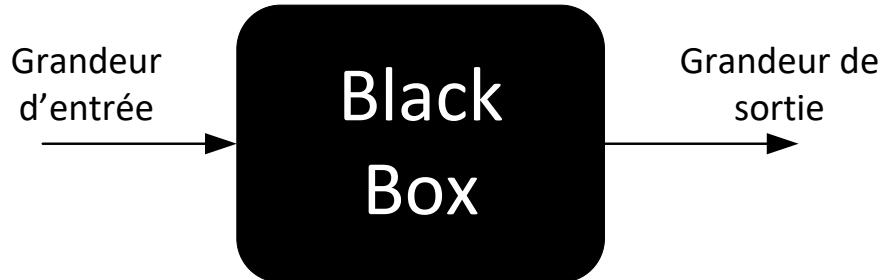


Figure 509 : la modélisation black box

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 où des phénomènes non-linéaires entrent 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 application appelée « System Identification » qui permet de mener à bien ce processus. Comme toutes les applications, elle est accessible à partie de l'onglet **APPS**.

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 510.

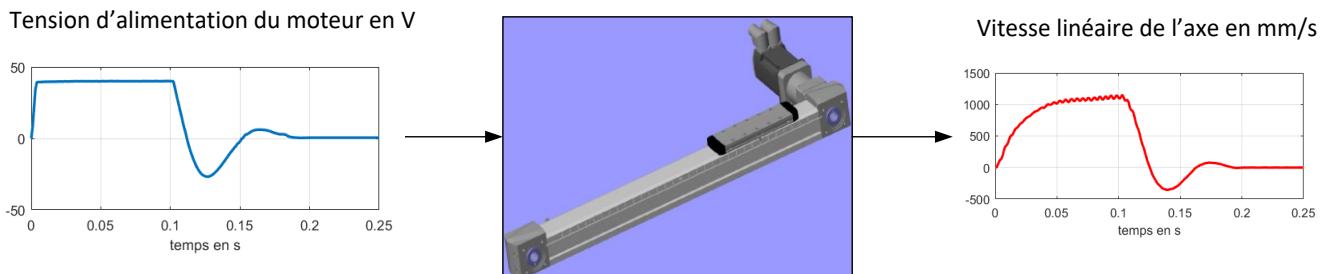


Figure 510 : les conditions de l'essai

La Figure 511 permet de visualiser les signaux d'entrée et de sortie.

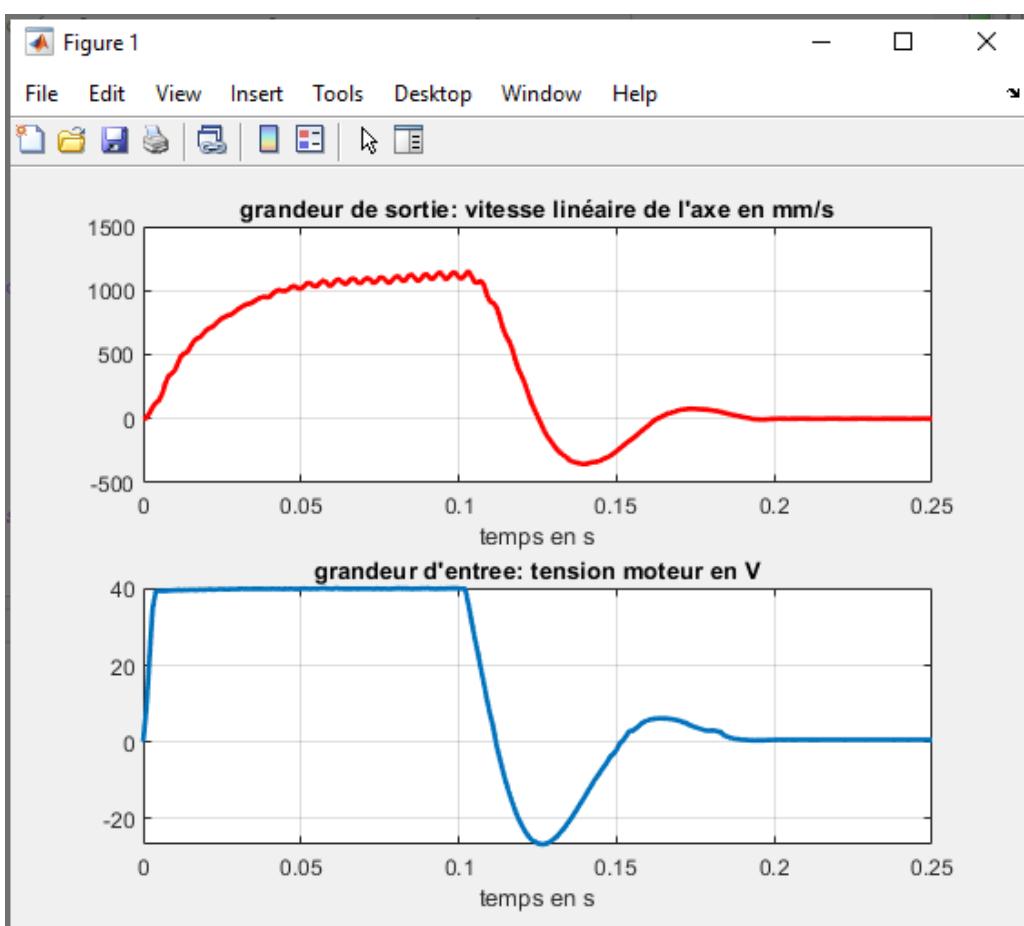


Figure 511 : visualisation des données utilisées pour l'identification

L'exécution du script a également créé dans le **Workspace** de MATLAB les variables suivantes :

Name	Value	Class
Reponse_indicielle	251x17 double	double
t	251x1 double	double
u	251x1 double	double
v	251x1 double	double
x	251x1 double	double

Figure 512 : 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 « System Identification »

Pour lancer l'application « **System Identification** », sélectionner l'onglet APPS dans la fenêtre de commande de MATLAB et activer le menu déroulant des applications proposées par le logiciel. Vous pouvez alors voir apparaître toutes les applications disponibles. Sélectionner l'application **System Identification** (Figure 513).

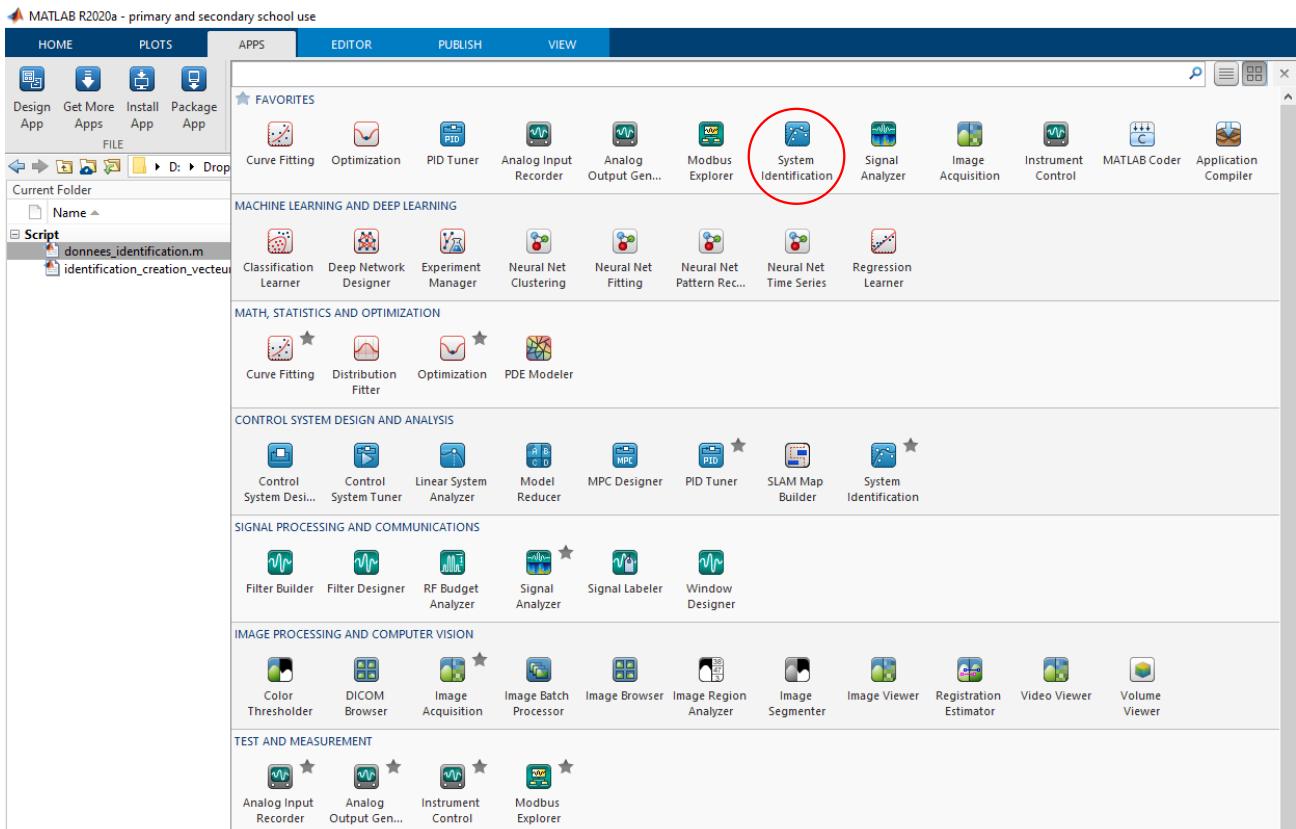


Figure 513 : menu déroulant des applications de MATLAB

Une fenêtre s'ouvre et présente les fonctionnalités de l'application (Figure 514).

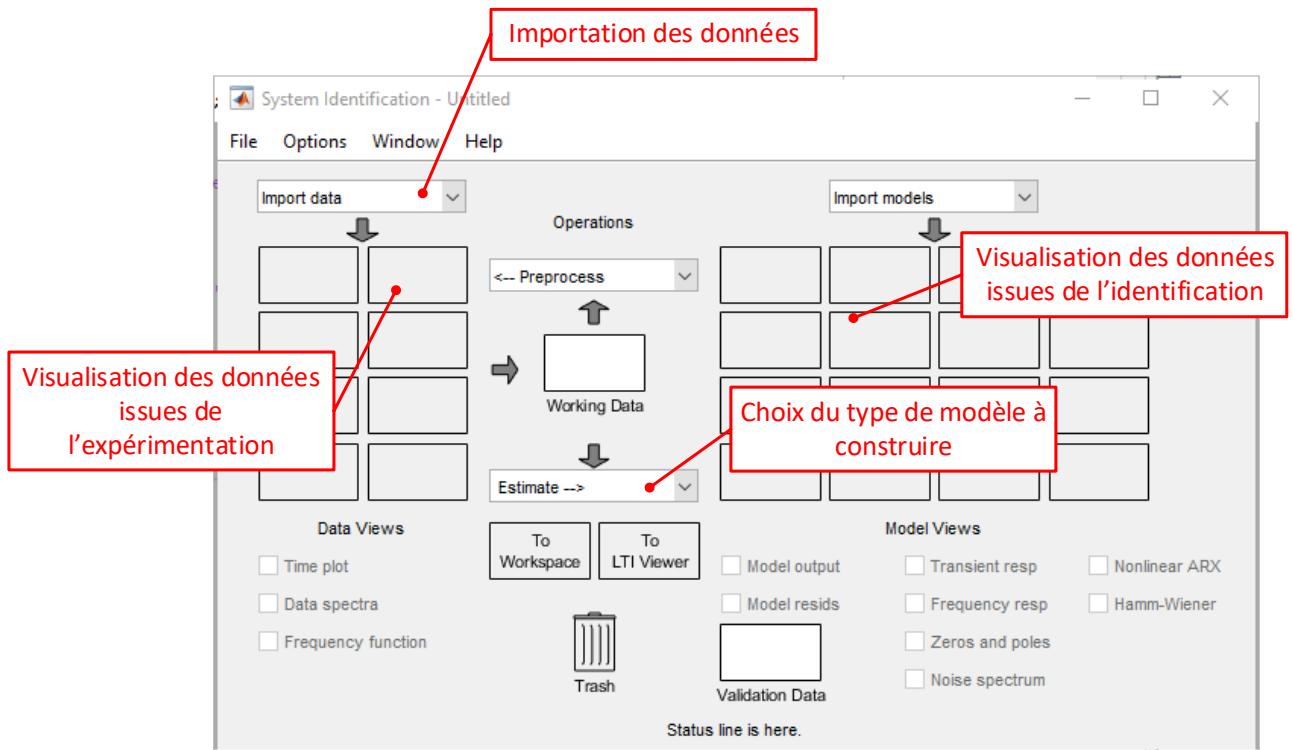


Figure 514 : description de la fenêtre de la toolbox identification

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**.

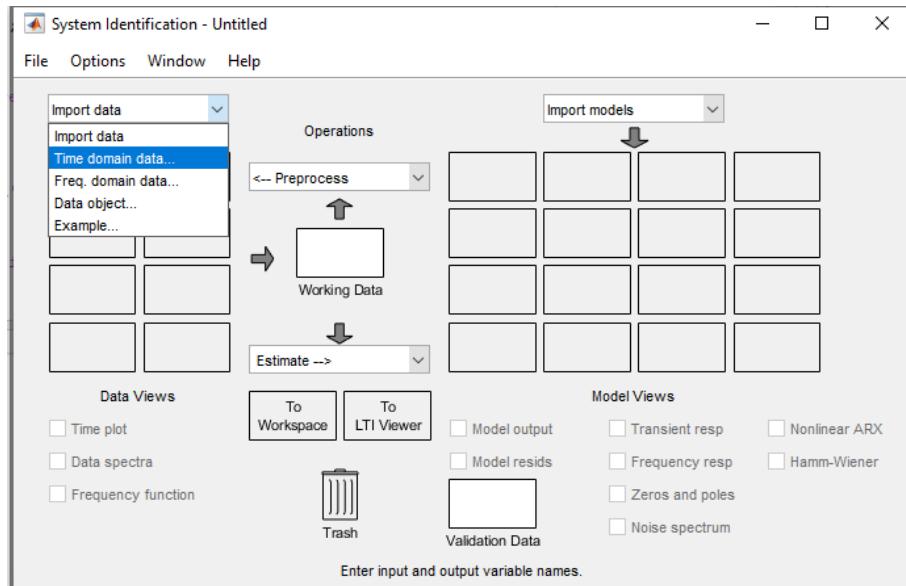


Figure 515 : importation de données temporelles

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 516.

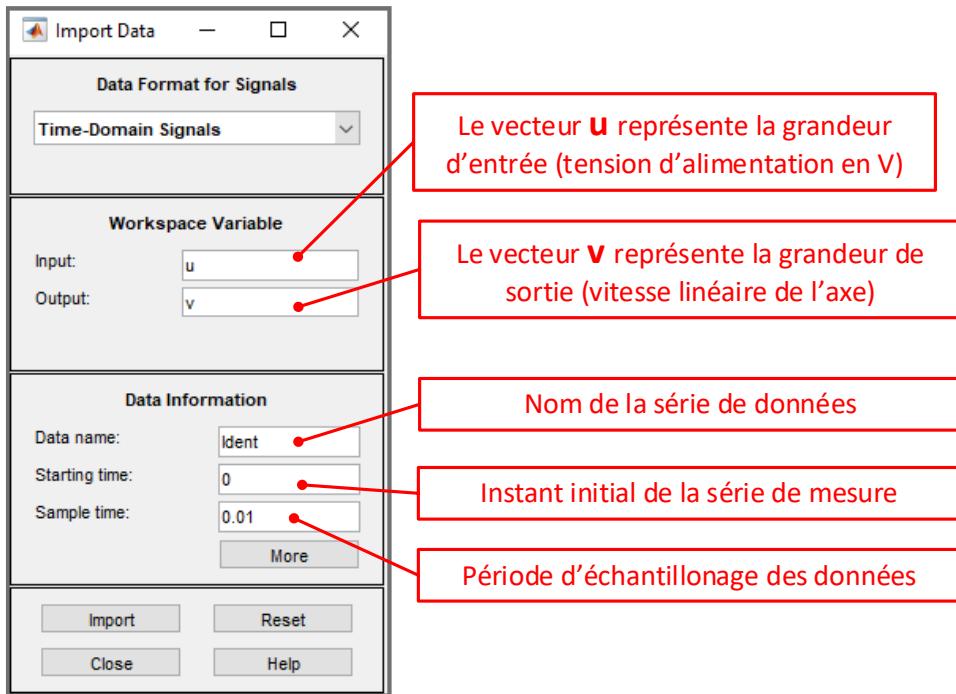


Figure 516 : sélection des données temporelles pour l'identification

Cliquer ensuite sur **Import** pour importer les données dans la fenêtre d'identification. Les données apparaissent alors dans la fenêtre de la « System Identification » toolbox comme le montre Figure 517. Cocher la case **Time plot** pour vérifier et visualiser la bonne importation des données (Figure 518).

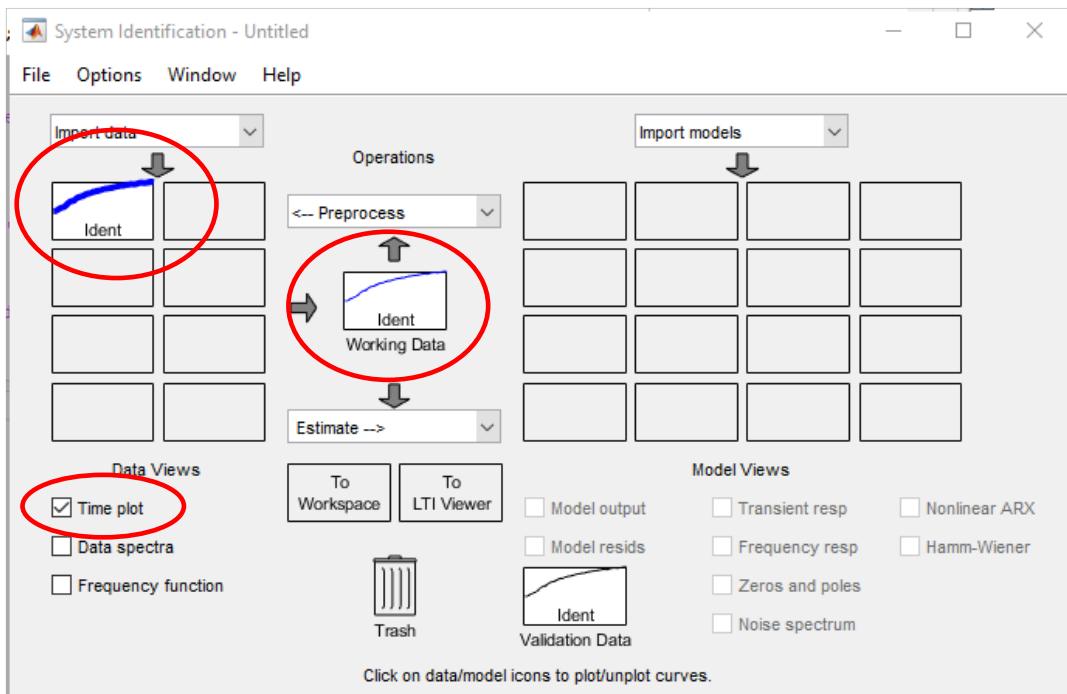


Figure 517 : visualisation des données importées dans la fenêtre de la « System Identification » toolbox

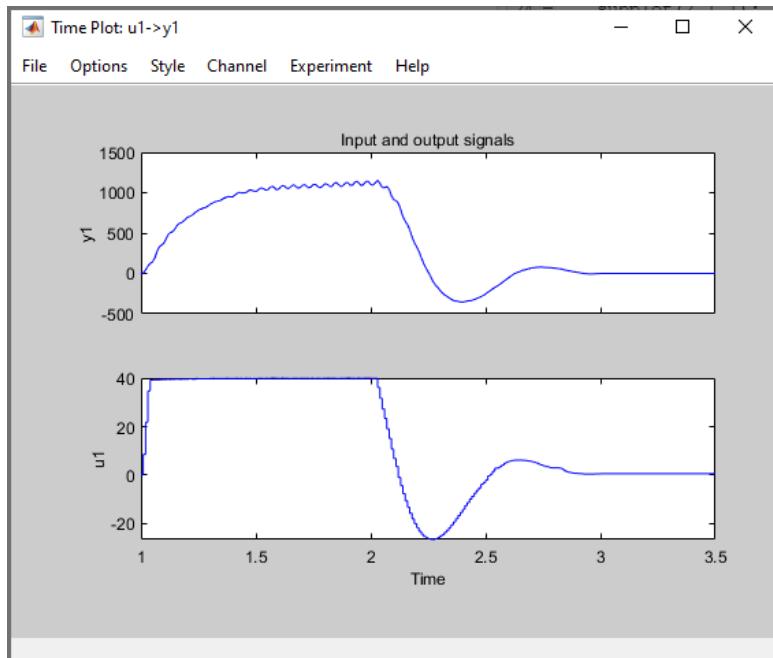


Figure 518 : visualisation des données importées dans la « System Identification »toolbox

Il faut maintenant choisir la forme du modèle que l'on souhaite obtenir (fonction de transfert, représentation d'état, modèle non-linéaires...). Nous choisirons ici un modèle de type **fonction de transfert**.

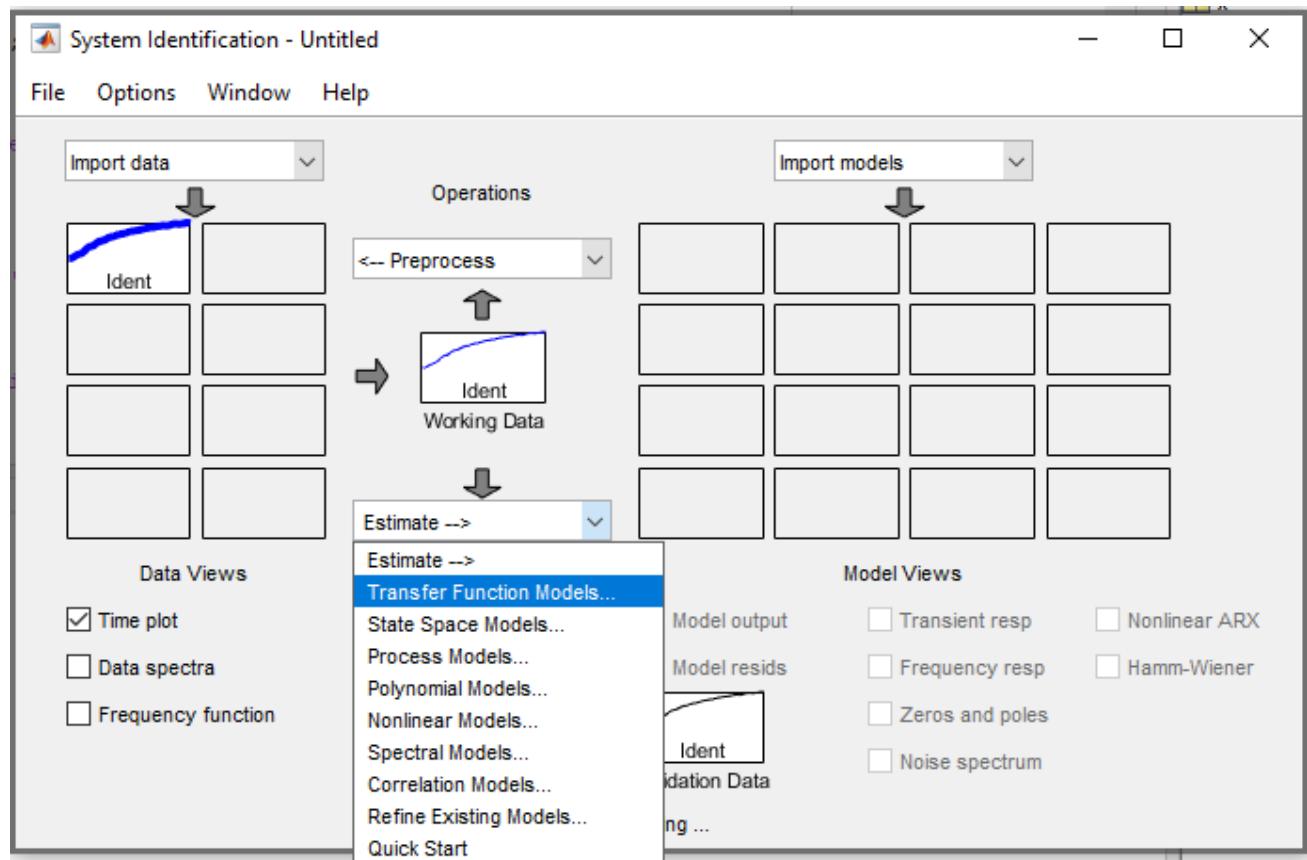


Figure 519 : choix de la forme du modèle dans la « System Identification » toolbox

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**.

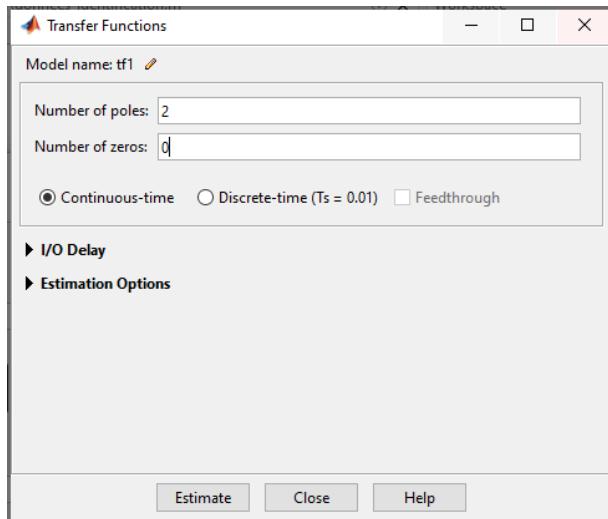


Figure 520 : choix de la forme de la fonction de transfert dans la « System Identification » toolbox

La fenêtre **Plant Identification Progress** donne les détails sur le processus numérique d'identification

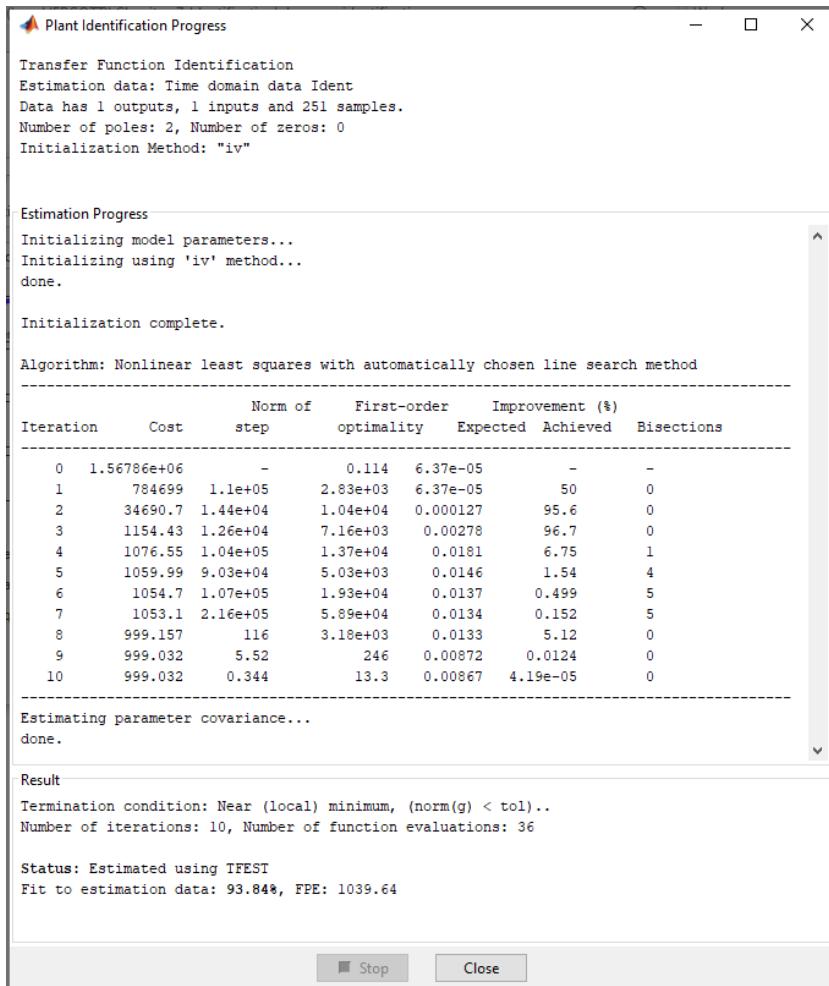


Figure 521 : la fenêtre Plant Identification Progress de la « System Identification » toolbox

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 522 et Figure 523). Cocher également la case **Model output** pour visualiser la superposition des données expérimentales et du modèle issu de l'identification (Figure 524).



Figure 522 : visualisation de résultats de l'identification

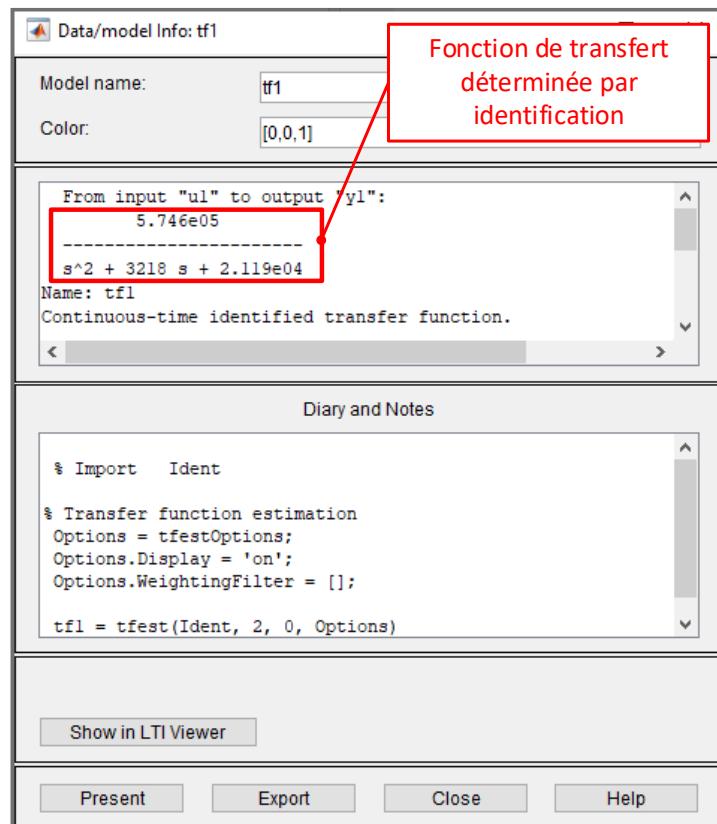


Figure 523 : fonction de transfert obtenue par identification

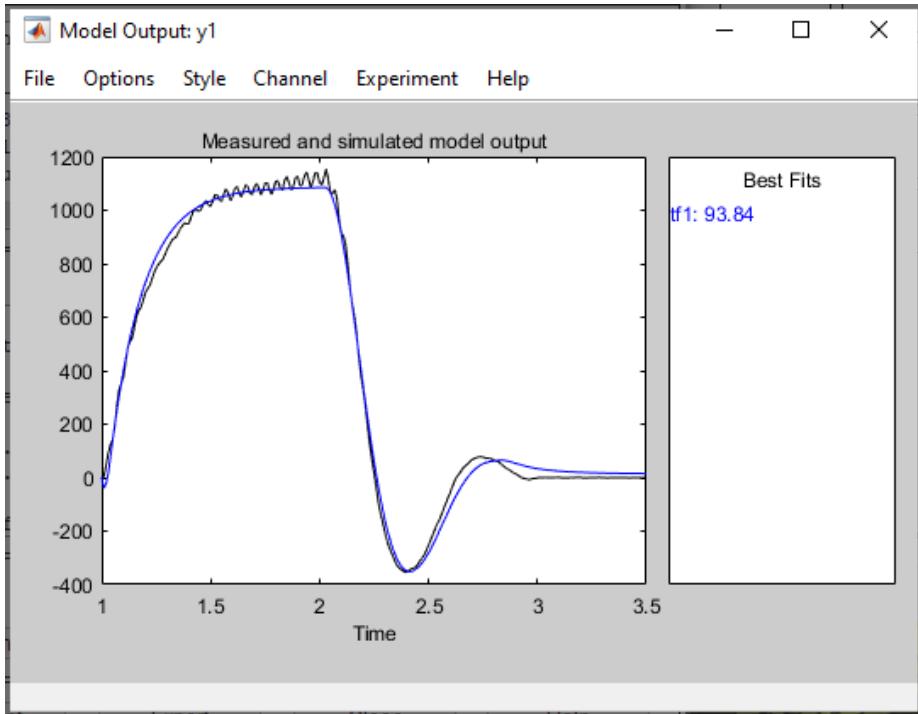


Figure 524 : superposition des données issues de l'expérimentation et du modèle issu de l'identification

Il est également possible à partir de la fenêtre Data/Model info (Figure 523) d'exporter la fonction de transfert déterminée par identification vers le Workspace de MATLAB en cliquant sur **Export**. La fonction de transfert **tf1** apparaît alors dans le Workspace. Il est possible par exemple de tracer un diagramme de Bode de cette fonction de transfert en tapant la commande **bode (tf1)** dans la fenêtre de commande de MATLAB.

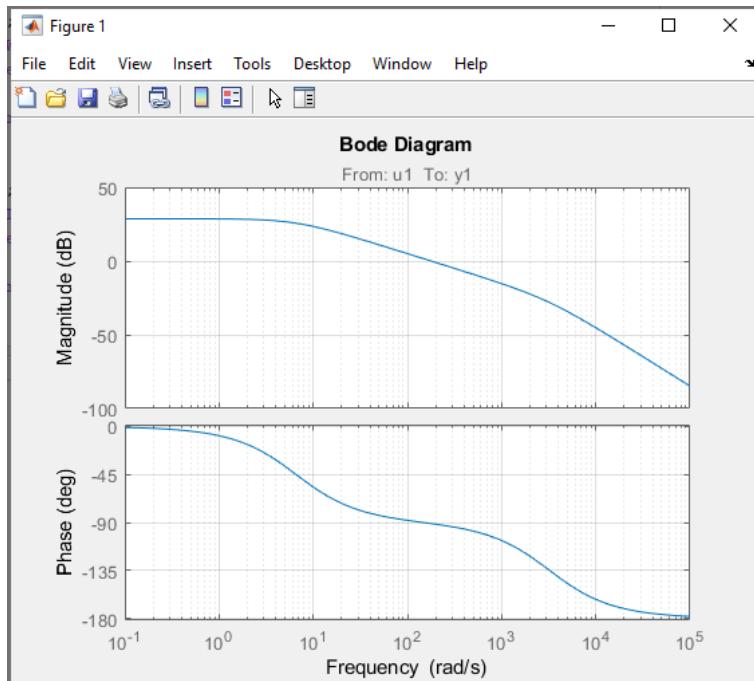


Figure 525 : diagramme de Bode de la fonction de transfert tf1

C. Utilisation de la méthode en utilisant les lignes de commandes

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 nécessaires 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 = tftest(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 l'application **System Identification**.

Commandes utiles pour l'identification	
Fonctions	Commandes
Création d'un objet <i>iddata</i> pour l'identifiacion	<code>mydata=iddata(y,u,Ts)</code> 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	 <code>H=tftest(mydata,np,nz)</code> $mydata$: objet créé à l'aide de la fonction <code>iddata</code> 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 10 : 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 où 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. Réglage 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

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

$u(t)$: tension de commande du moteur (V)
 $e(t)$: force contre électromotrice du moteur (V)
 $i(t)$: intensité dans le moteur (A)
 $C_m(t)$: couple exercé par le moteur (N.m)
 $C_r(t)$: couple résistant ramené sur l'arbre moteur (N.m)
 $\omega_m(t)$: vitesse de rotation du moteur (rad/s)
 R : résistance de l'induit (Ω)
 L : inductance de l'induit (H)
 J : inertie équivalente ramenée sur l'arbre moteur ($\text{kg} \cdot \text{m}^2$)
 f : paramètre de frottement visqueux (N.m.s)
 K_t : constante de couple (N.m/A)
 K_e : coefficient de force contre électromotrice (V.s/rad)

Figure 526 : 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 527.

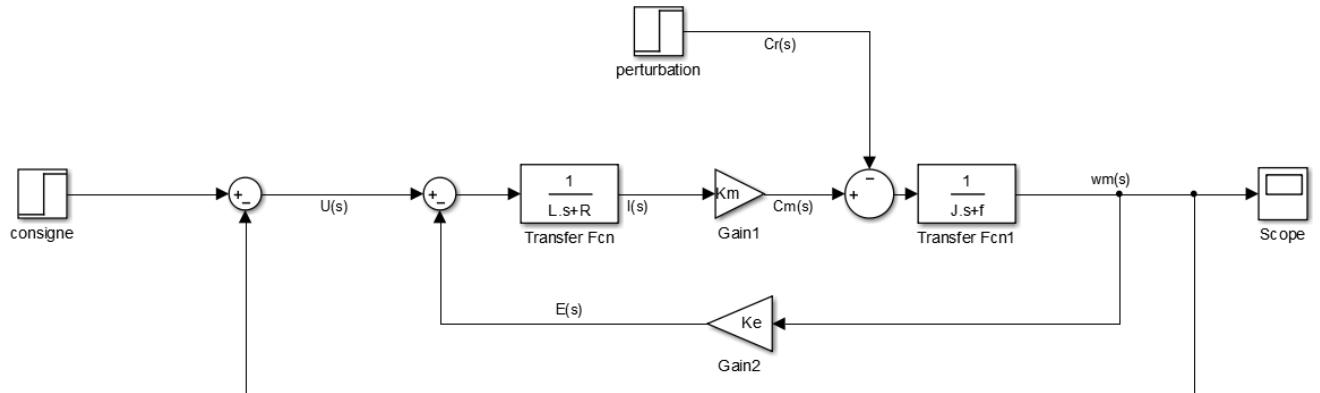


Figure 527 : asservissement en vitesse d'un moteur à courant continu

B. Ouverture du modèle

Ouvrir le fichier **moteur_cc_ass_vit.slx**. Certains blocs apparaissent entourés en rouge à l'ouverture du modèle (Figure 528). Cela signifie que certaines variables ne sont pas connues et que l'exécution du modèle entraînera un message d'erreur. Il faut donc donner des valeurs numériques à l'ensemble des variables du modèle.

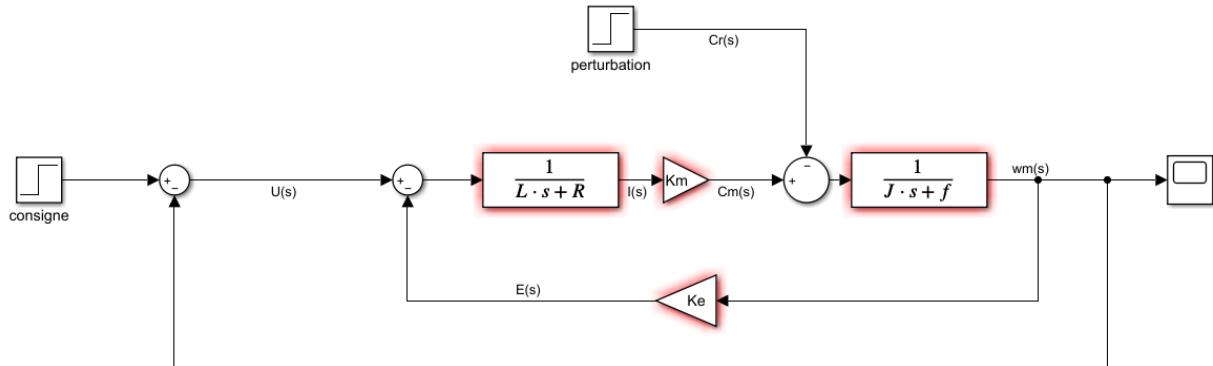


Figure 528 : modélisation de l'asservissement en vitesse du moteur

Ouvrir le script **parametres_moteur_cc.m** et exécuter-le.

Lancer la simulation et visualiser la réponse dans le scope.

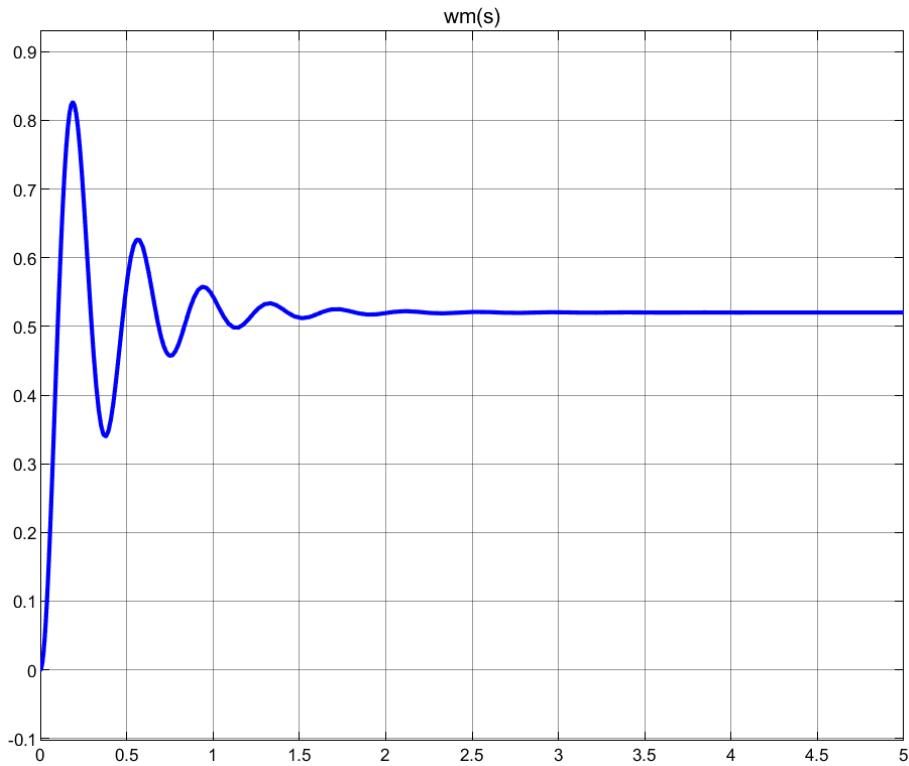


Figure 529 : réponse temporelle du système non corrigé à un échelon unitaire

Pour effectuer le réglage automatique d'un PID, il faut insérer le bloc **PID controller** conformément à la Figure 530.

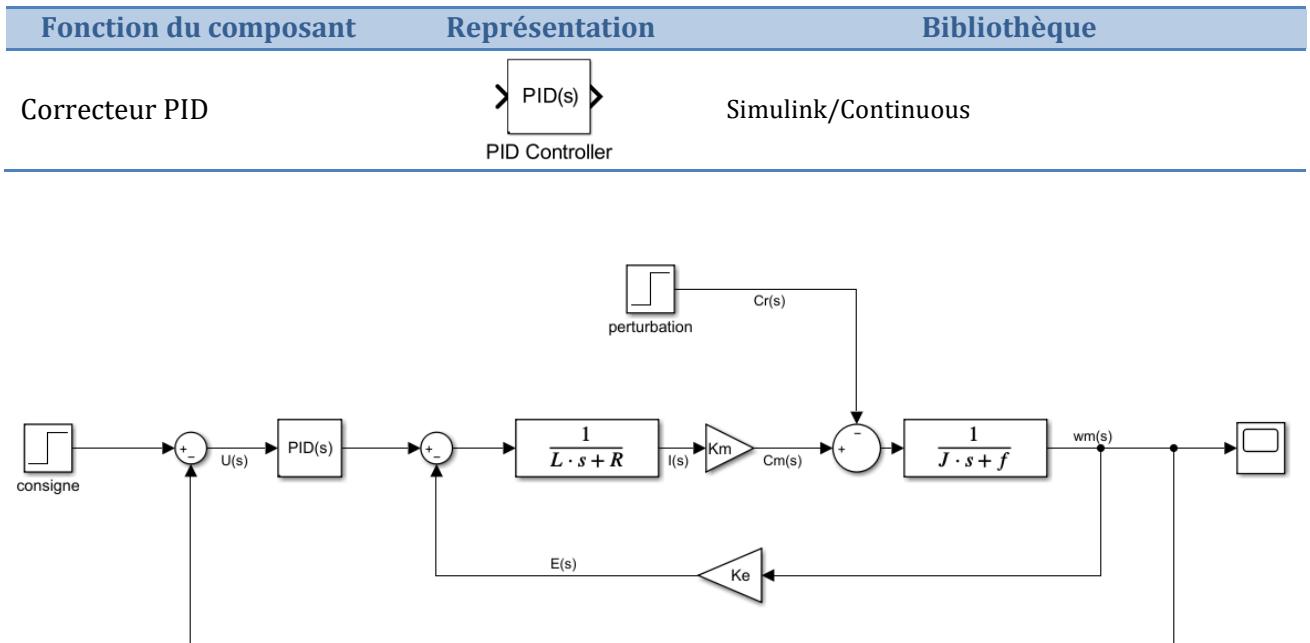


Figure 530 : ajout d'un bloc PID Controller dans l'asservissement

Double cliquer sur le bloc **PID Controller** pour ouvrir la fenêtre de paramétrage de la Figure 531.

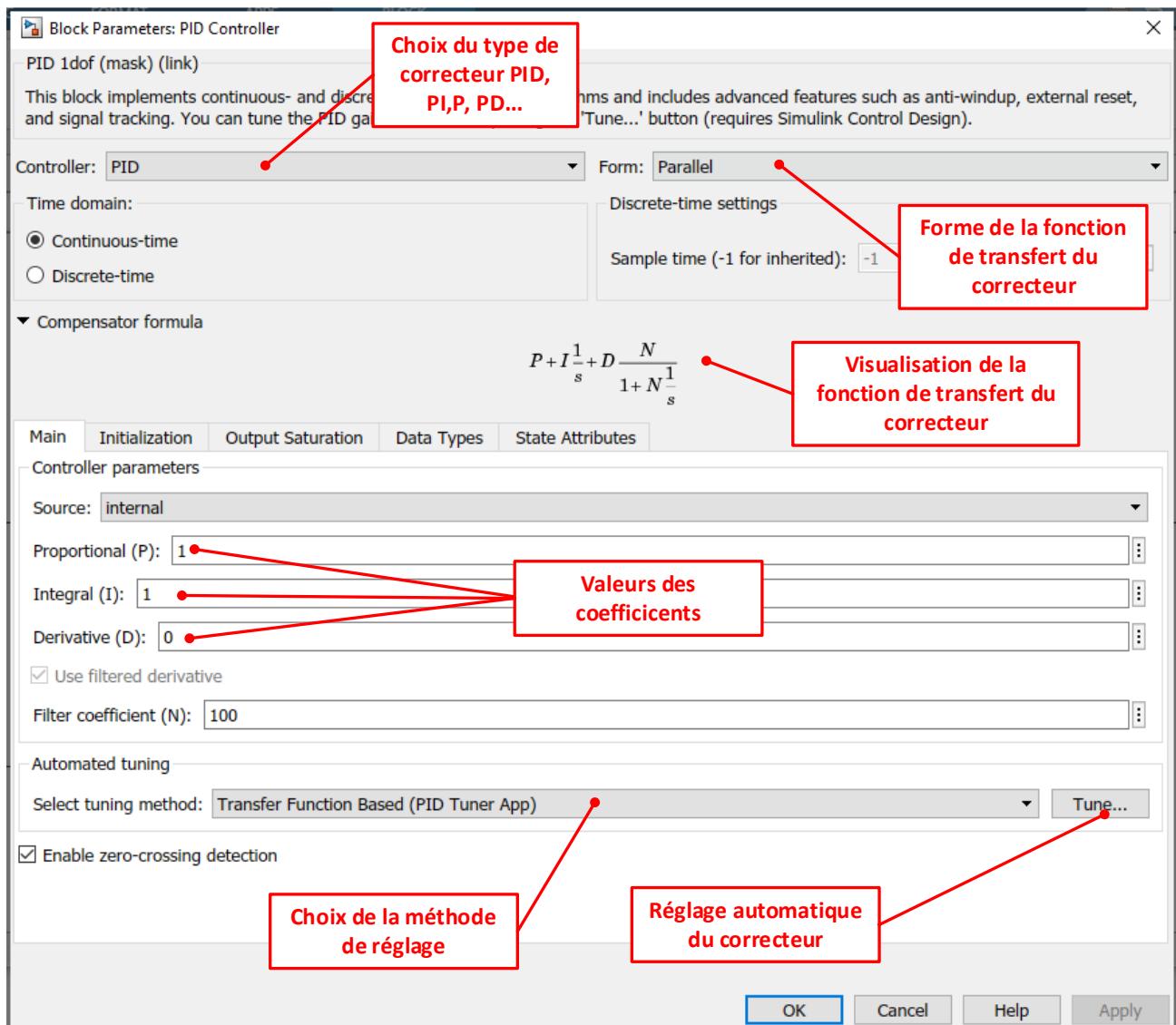


Figure 531 : fenêtre de paramétrage du PID Controller

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 532. 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.

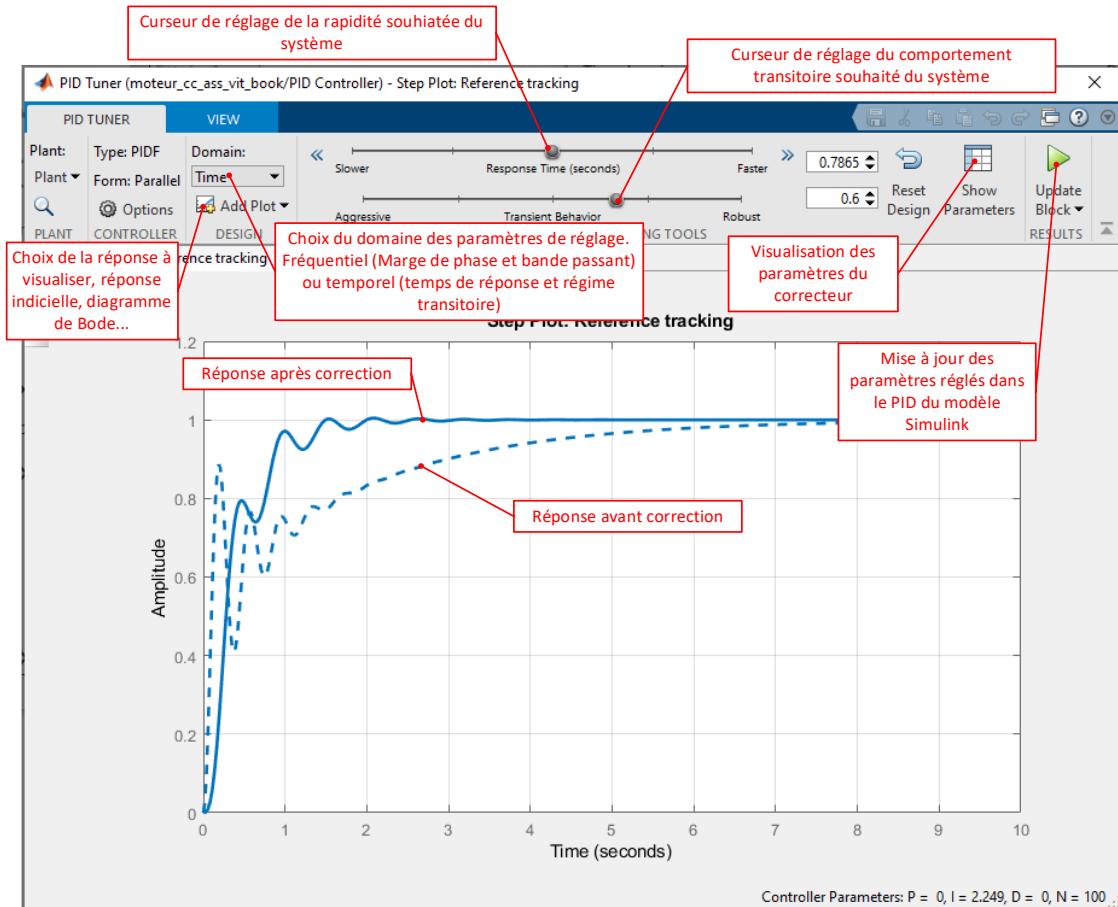


Figure 532 : fenêtre de réglage des performances du système

Il est cependant possible d'agir manuellement sur les curseurs pour modifier les performances du système et le rendre conforme à 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 533.

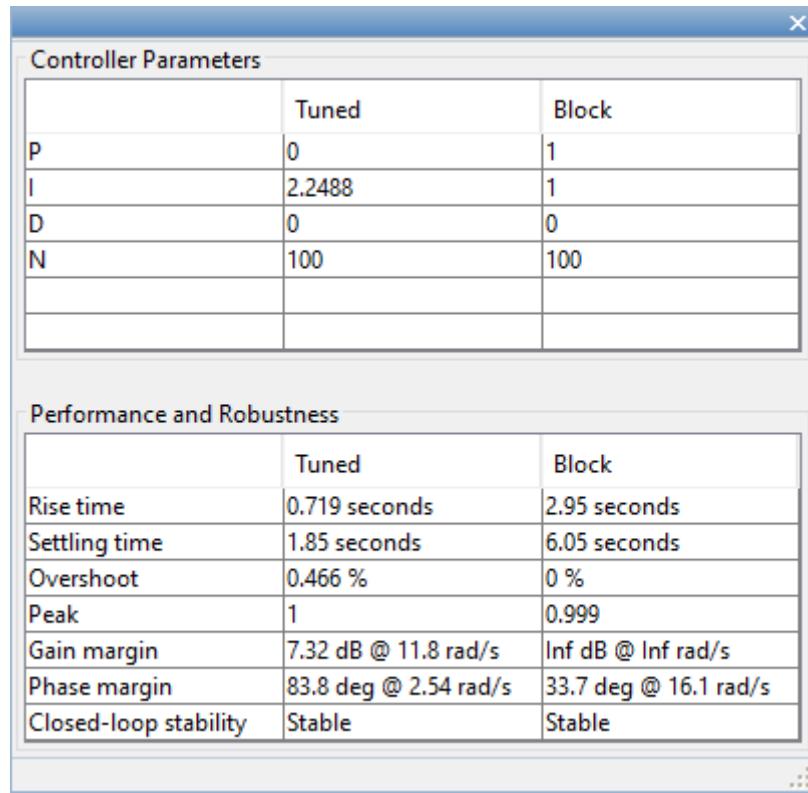


Figure 533 : les critères de performances du système et les valeurs de gain du correcteur

Pour obtenir le **diagramme de Bode la fonction de transfert en boucle ouverte**, sélectionner le menu **Add Plots/Bode/Open Loop** (Figure 534).

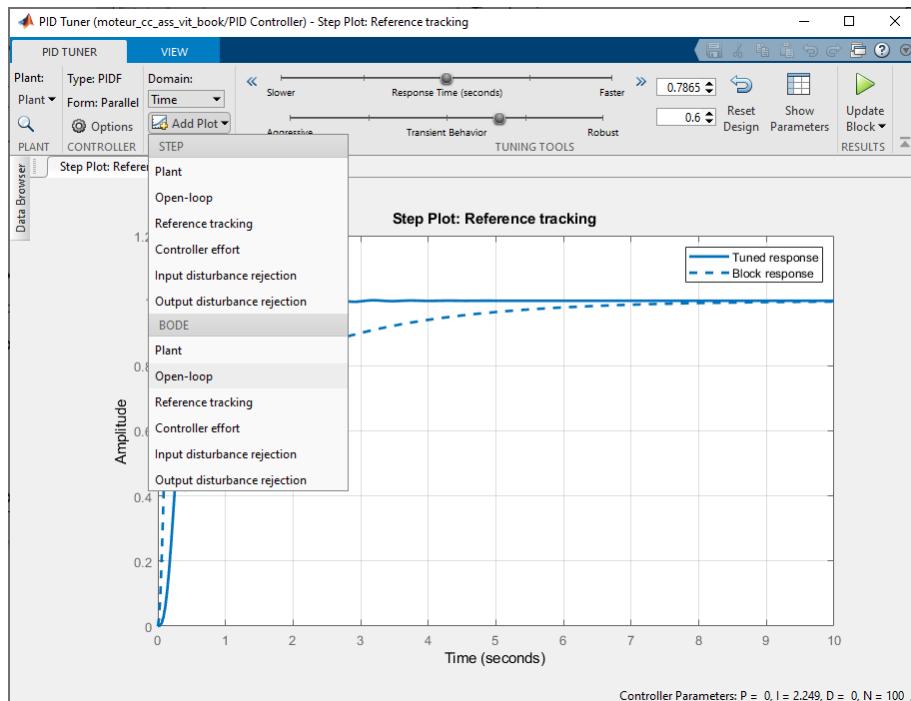


Figure 534 : ajout du diagramme de Bode de la fonction de transfert en boucle ouverte

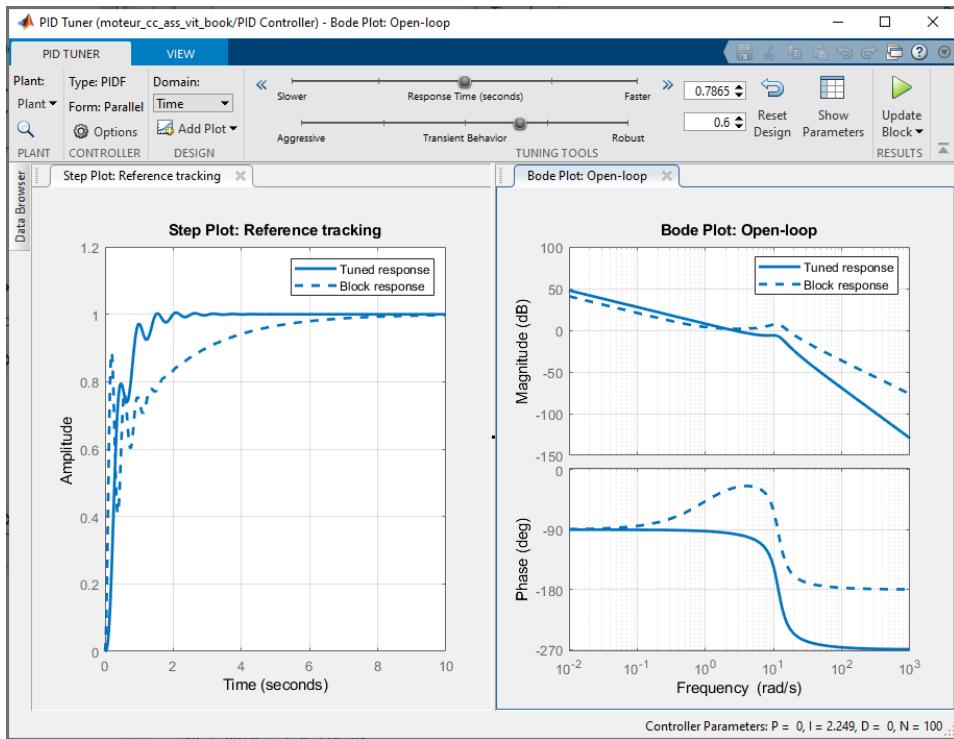


Figure 535 : visualisation du diagramme de Bode de la fonction de transfert en boucle ouverte pour le réglage du PID

Les diagrammes de Bode du système corrigé et non- corrigé apparaissent sur une seconde fenêtre (Figure 535).

Pour régler les options des critères de performances, **cliquer droit** dans la fenêtre graphique de la réponse temporelle puis sélectionner **Properties**, puis onglet **Options**

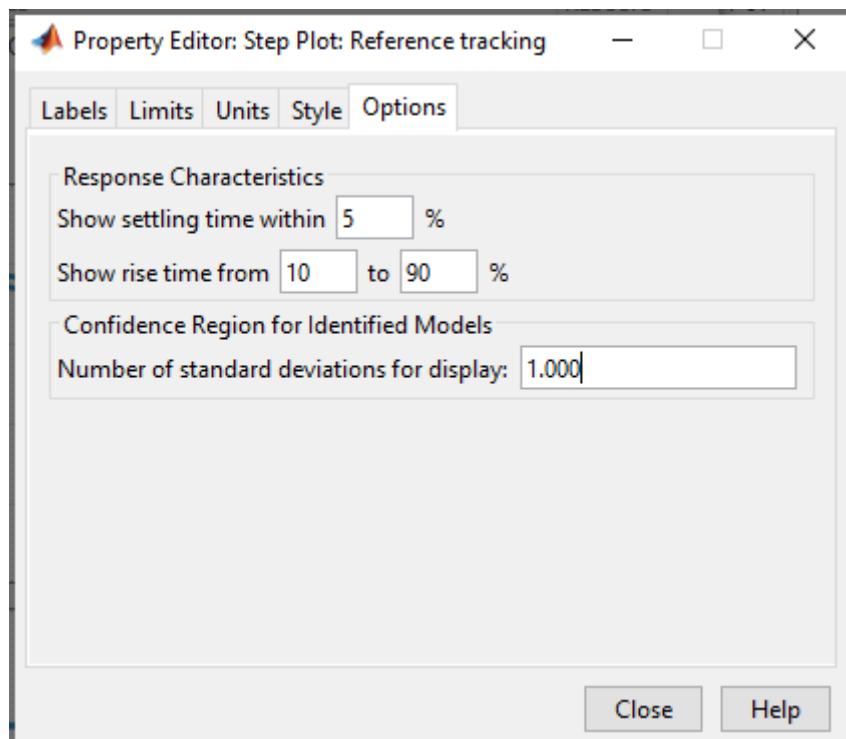


Figure 536 : réglage des options des critères de performances

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 537).

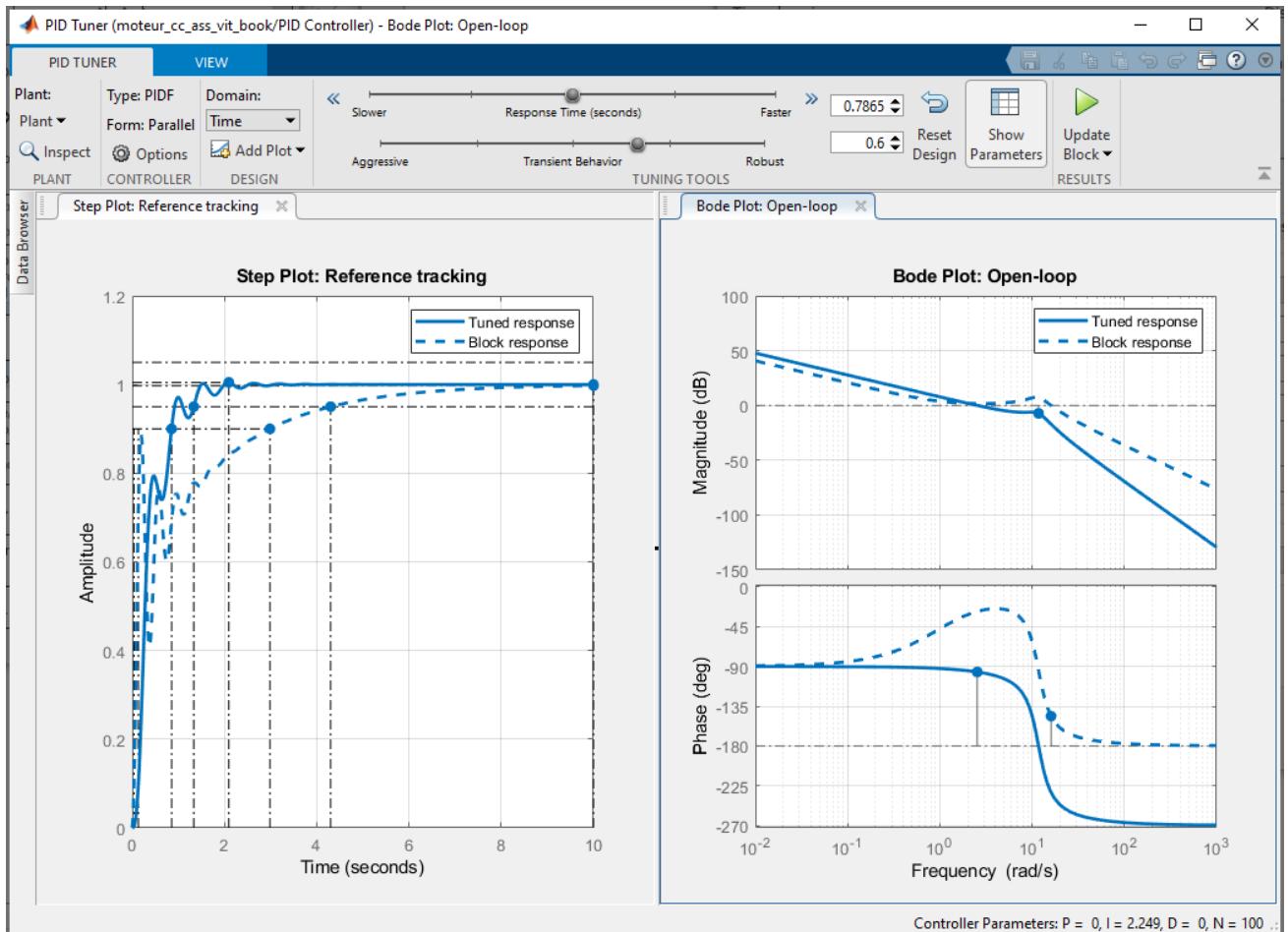


Figure 537 : visualisation des critères de performances

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**. Seule la réponse corrigée apparaît alors dans la fenêtre graphique.

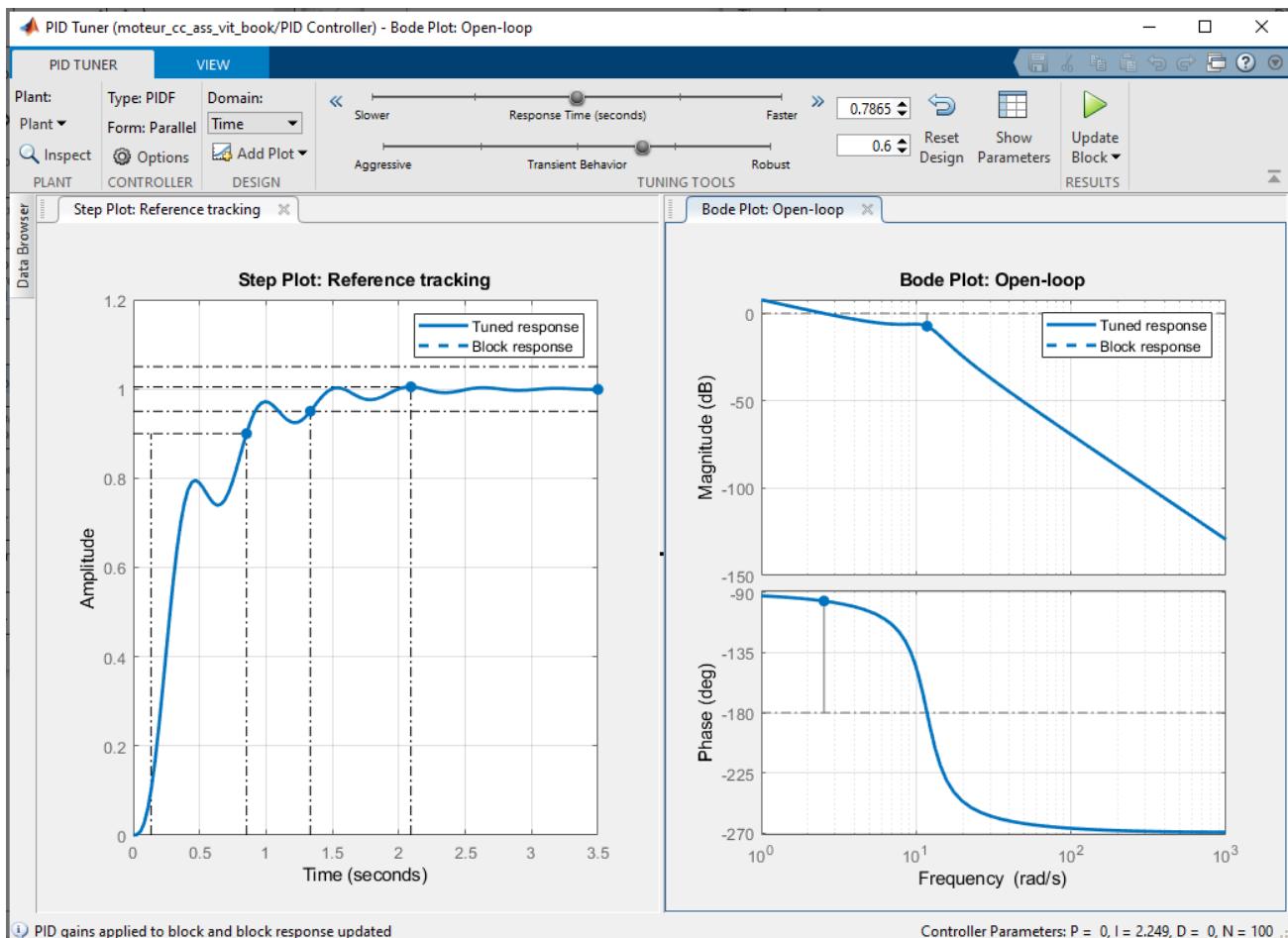


Figure 538 : réponse corrigée

2. Importation dans Simulink

Une fois que la réponse du système est 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.

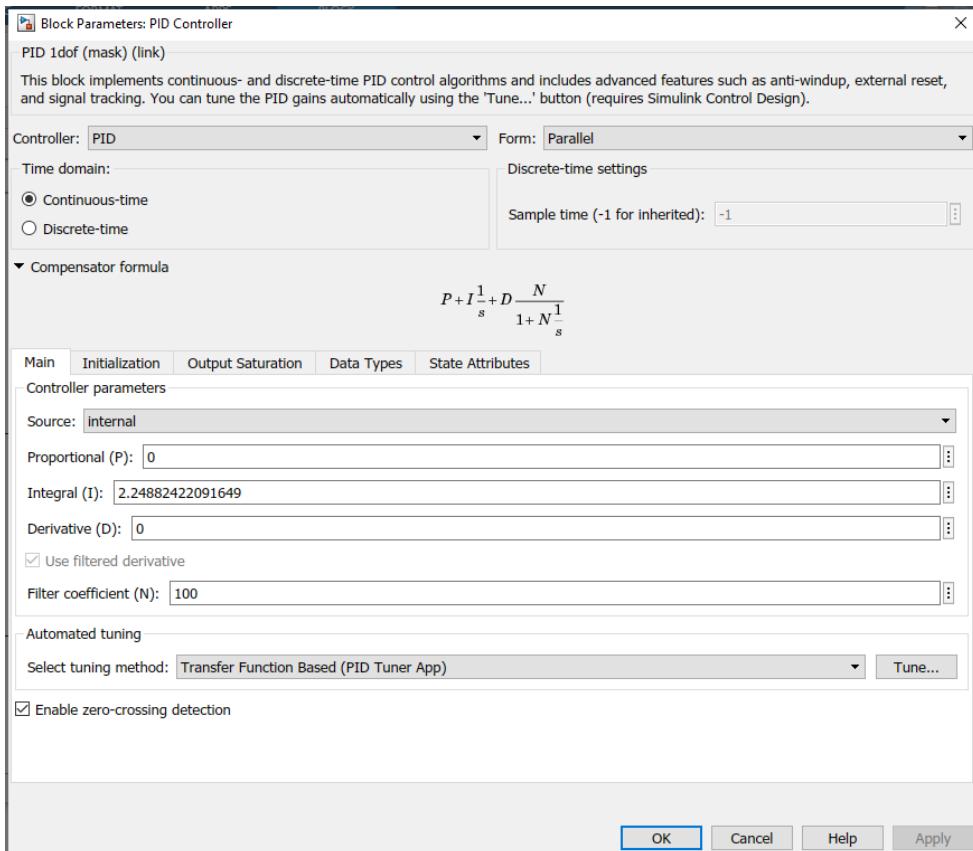


Figure 539 : actualisation automatique des paramètres du PID

Cliquez sur **OK**.

Lancer la simulation et visualiser la réponse corrigée dans le scope.

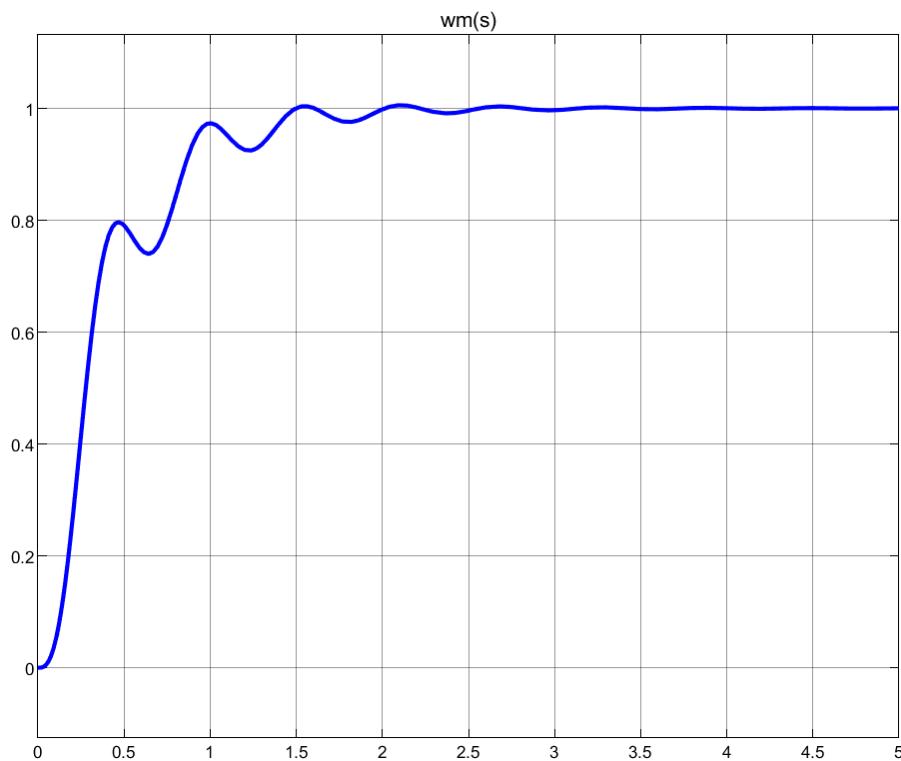


Figure 540 : réponse du système après réglage du PID

III. Réglage manuel d'un PID avec l'outil « Control System Designer »

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, nous utiliserons l'application « Control System Designer ».

A. Ouverture du modèle

Ouvrir le fichier **moteur_cc_ass_vit_control_sys_des_PID.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).

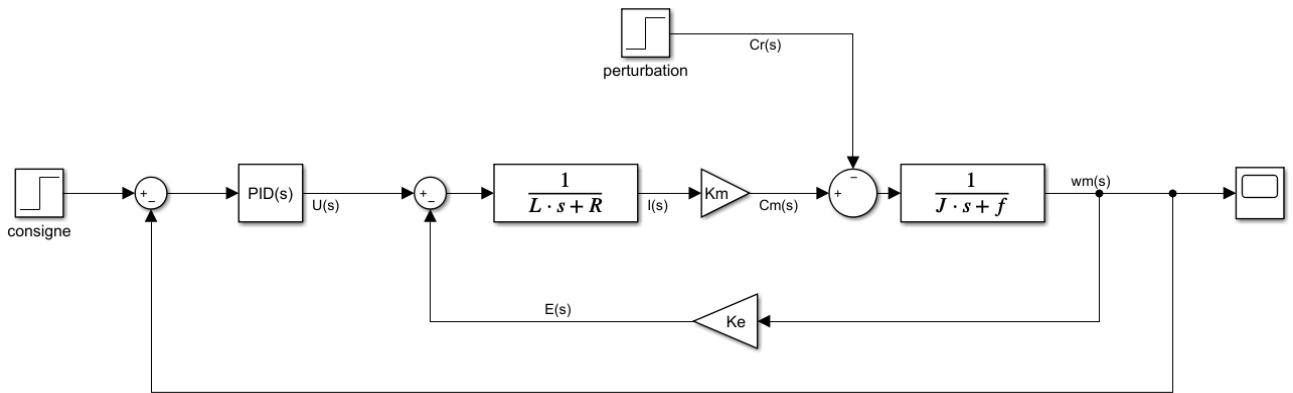


Figure 541 : modèle du moteur à courant continu asservi en vitesse avec perturbation

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.

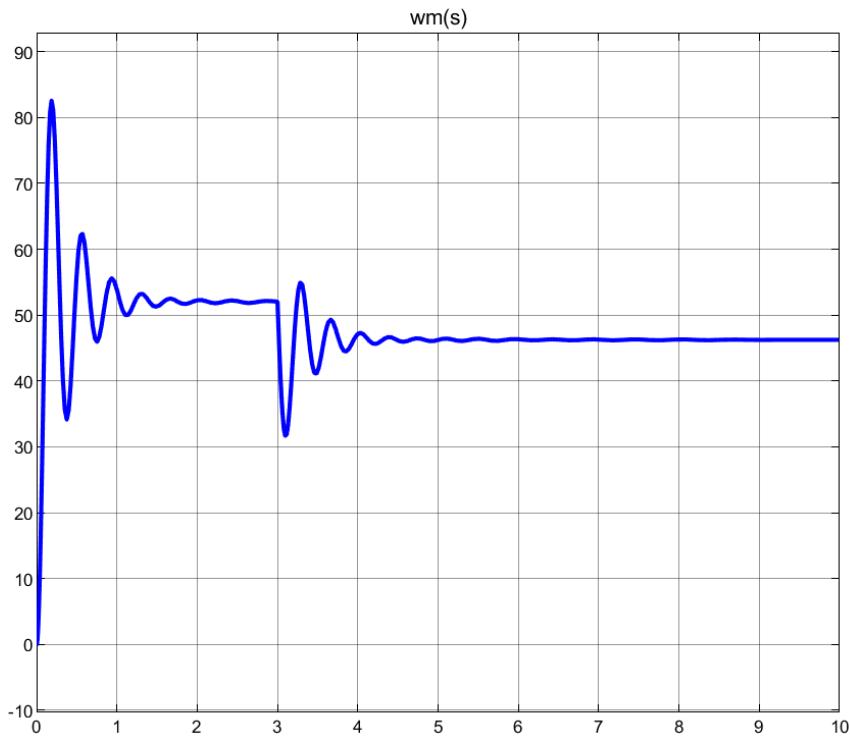


Figure 542 : réponse temporelle du système non corrigé

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

1. Lancement de Control System Designer

A partir de l'onglet APPS de la fenêtre de commande de MATLAB, sélectionner l'application « Control System Designer » (Figure 543).

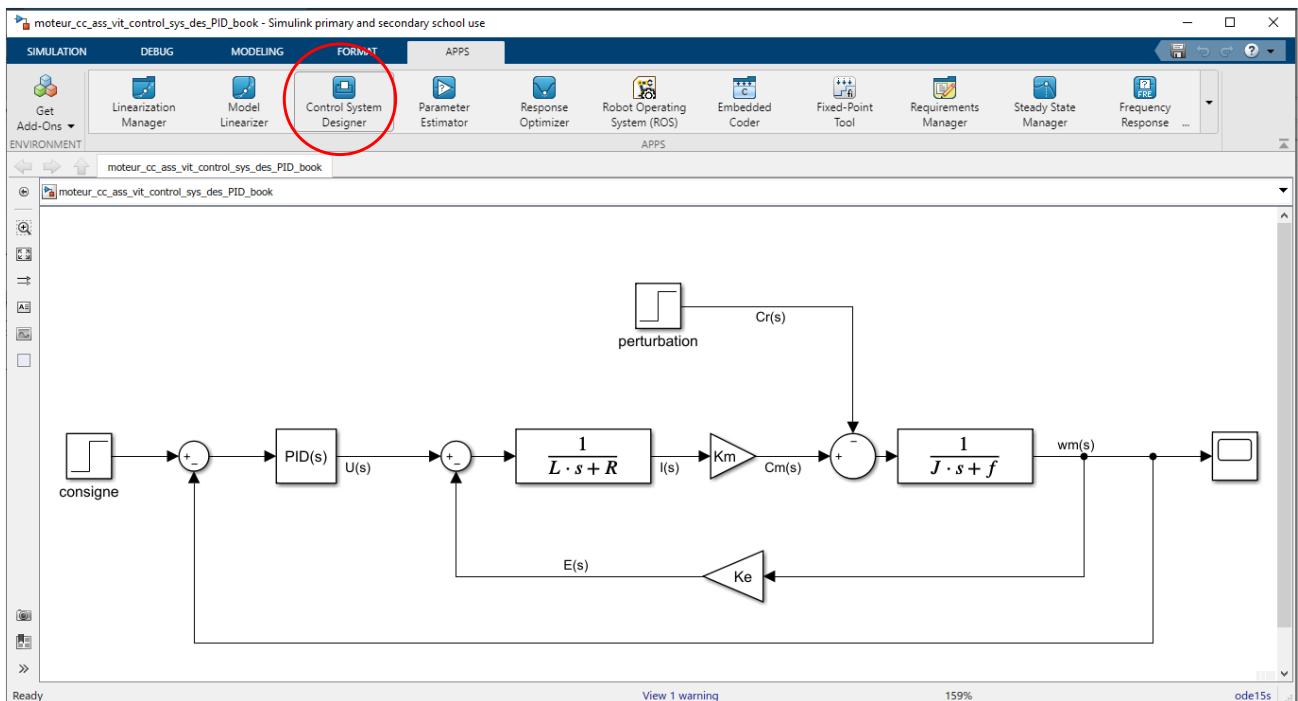


Figure 543 : lancement de l'application « Control System Designer »

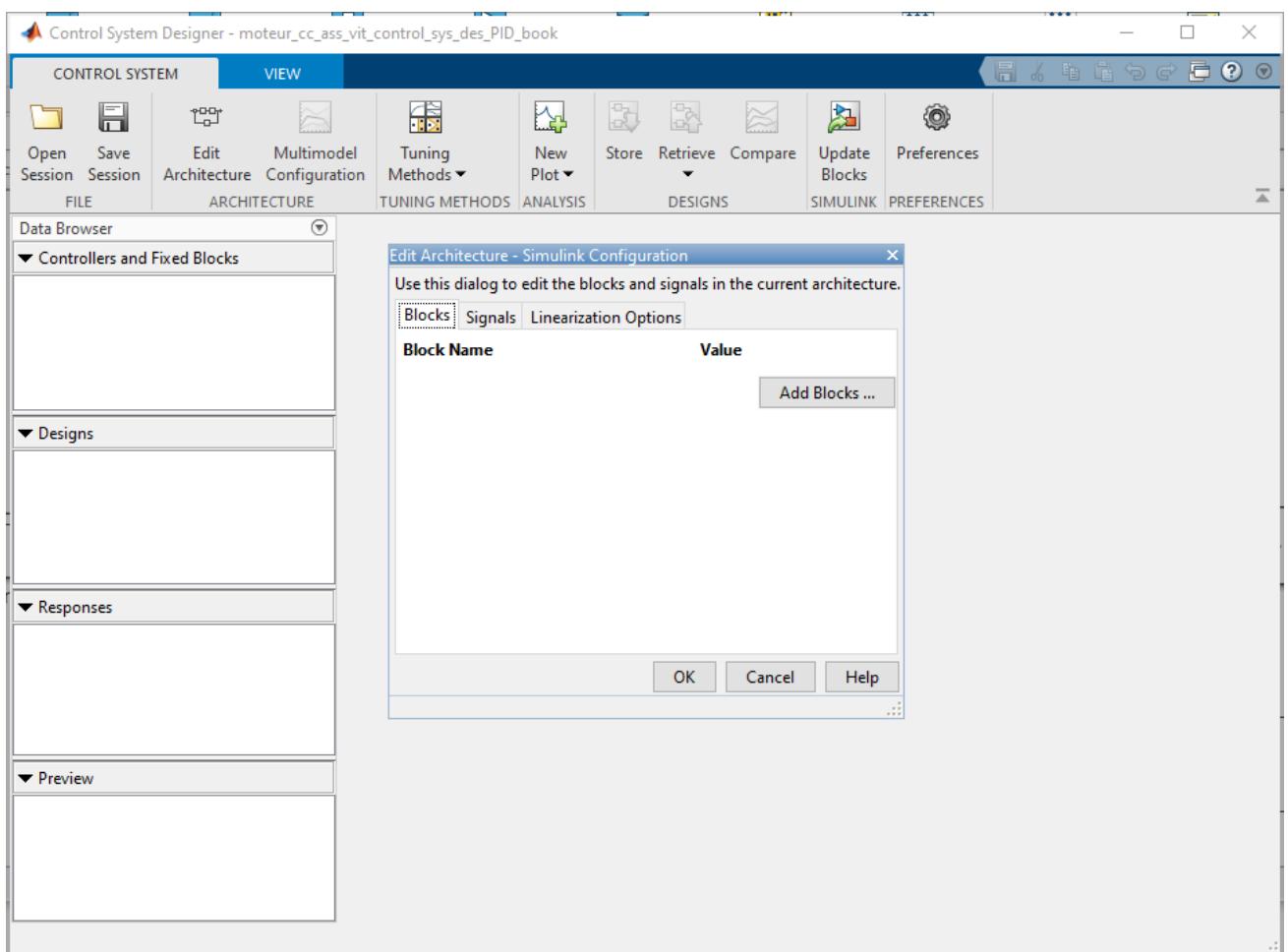


Figure 544 : Control System Designer

Afin de sélectionner le bloc à régler cliquer sur **Add Blocks** et choisir PID Controller comme bloc à régler, puis cliquer sur **OK**. (Figure 545).

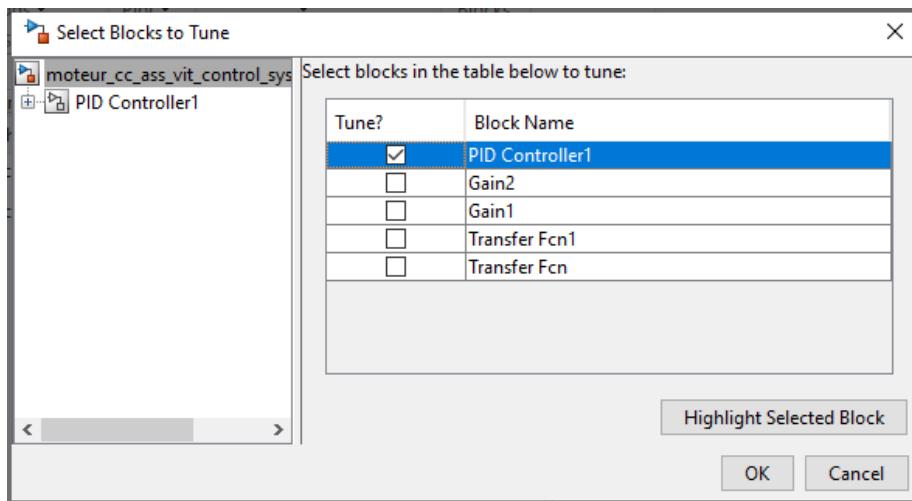


Figure 545 : choix du bloc à régler

Dans la fenêtre **Edit Architecture** (Figure 546) qui apparaît, cliquer sur OK pour valider votre choix.

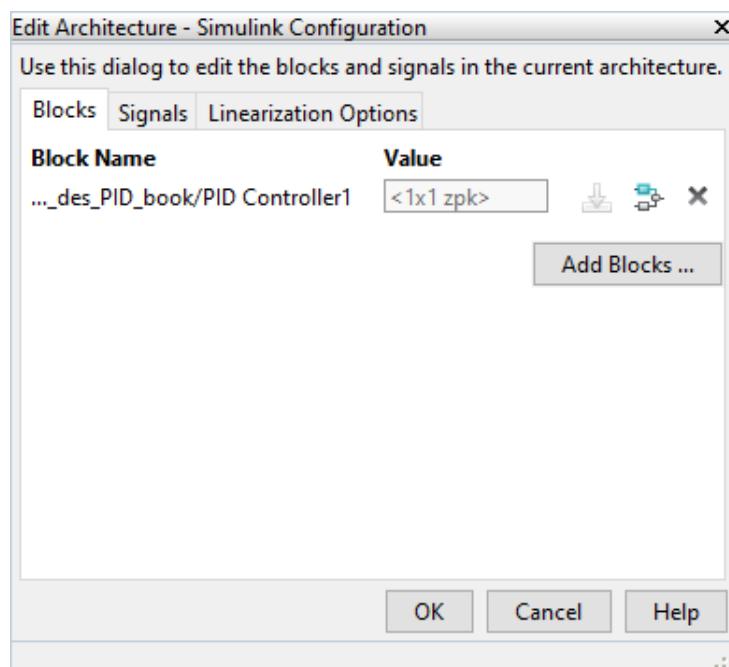


Figure 546 : fenêtre Edit Architecture

2. Diagrammes de Bode et de Black de la FTBO

Nous allons maintenant choisir la méthode de réglage. Dans la fenêtre du Control System Designer, sélectionner **Tuning Method/Bode Editor**.

La fenêtre **Select Response to Edit** apparaît. La fonction de transfert en boucle ouverte (FTBO) est automatiquement détectée. Les points de linéarisation sont indiqués (Figure 547).

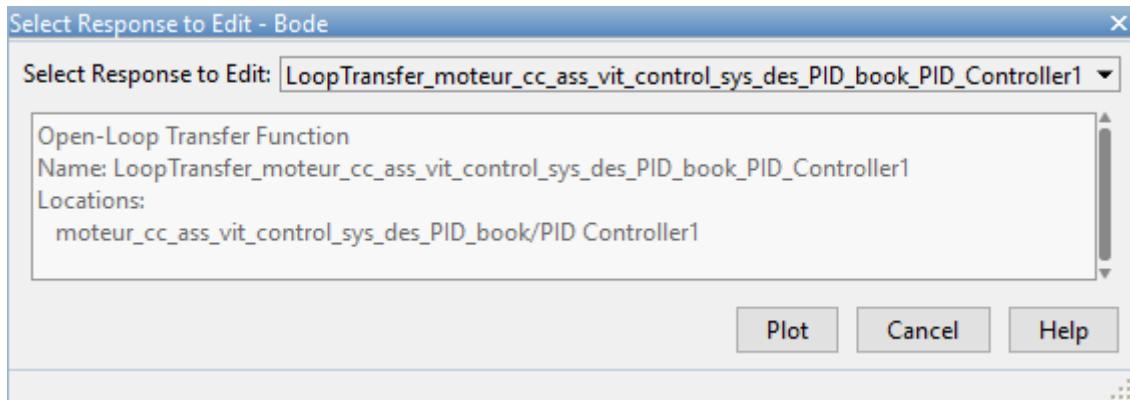


Figure 547 : choix des points de linéarisation pour la FTBO

Nous verrons plus tard comment choisir nous-même les points de linéarisation.

Cliquer sur **Plot** pour tracer le diagramme de Bode de la FTBO. Pour ajouter un quadrillage, cliquer avec le bouton droit dans la fenêtre du diagramme de Bode et choisir **Grid**.

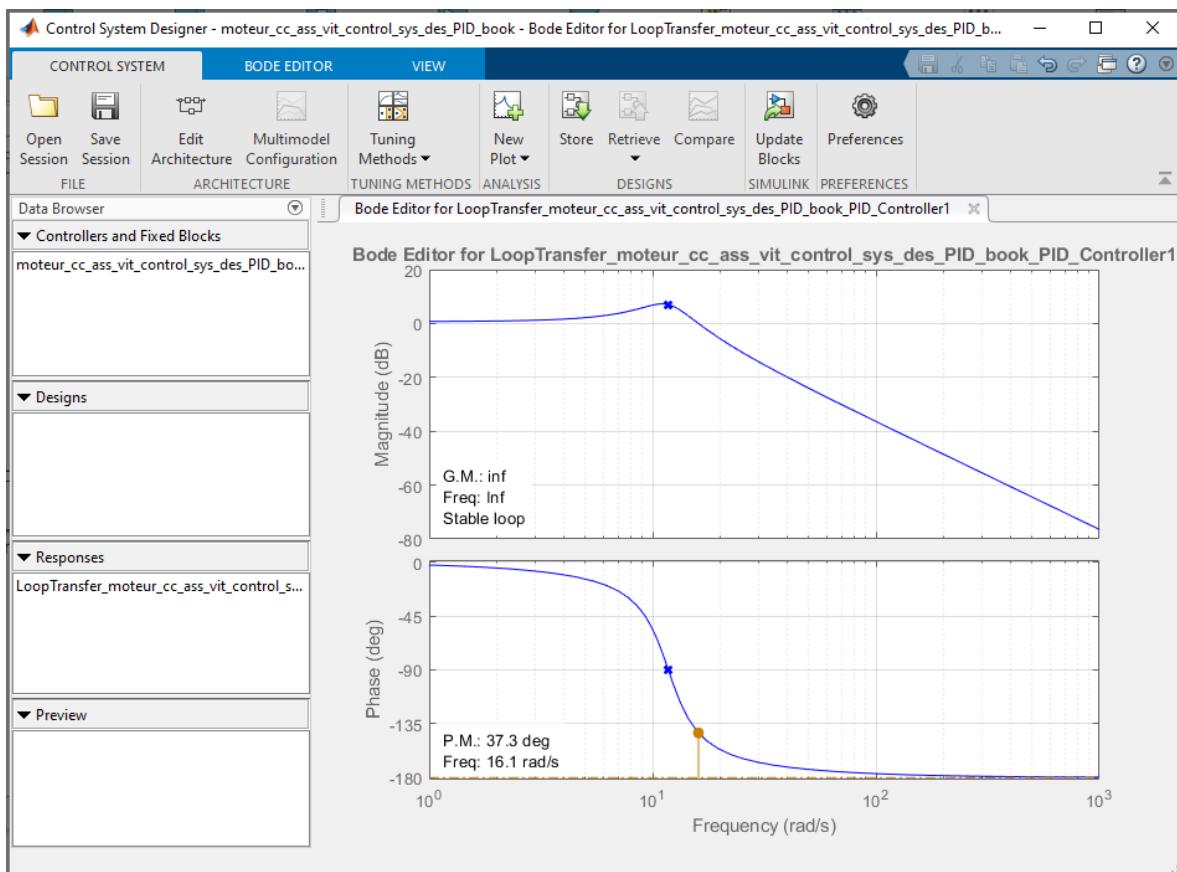


Figure 548 : diagramme de Bode de la FTBO

Il est également possible d'ajouter d'autres types de diagramme pour la FTBO.
 Cliquer sur **New Plot/New Nichols** puis cliquer sur **Plot** pour faire apparaître un diagramme de Black.

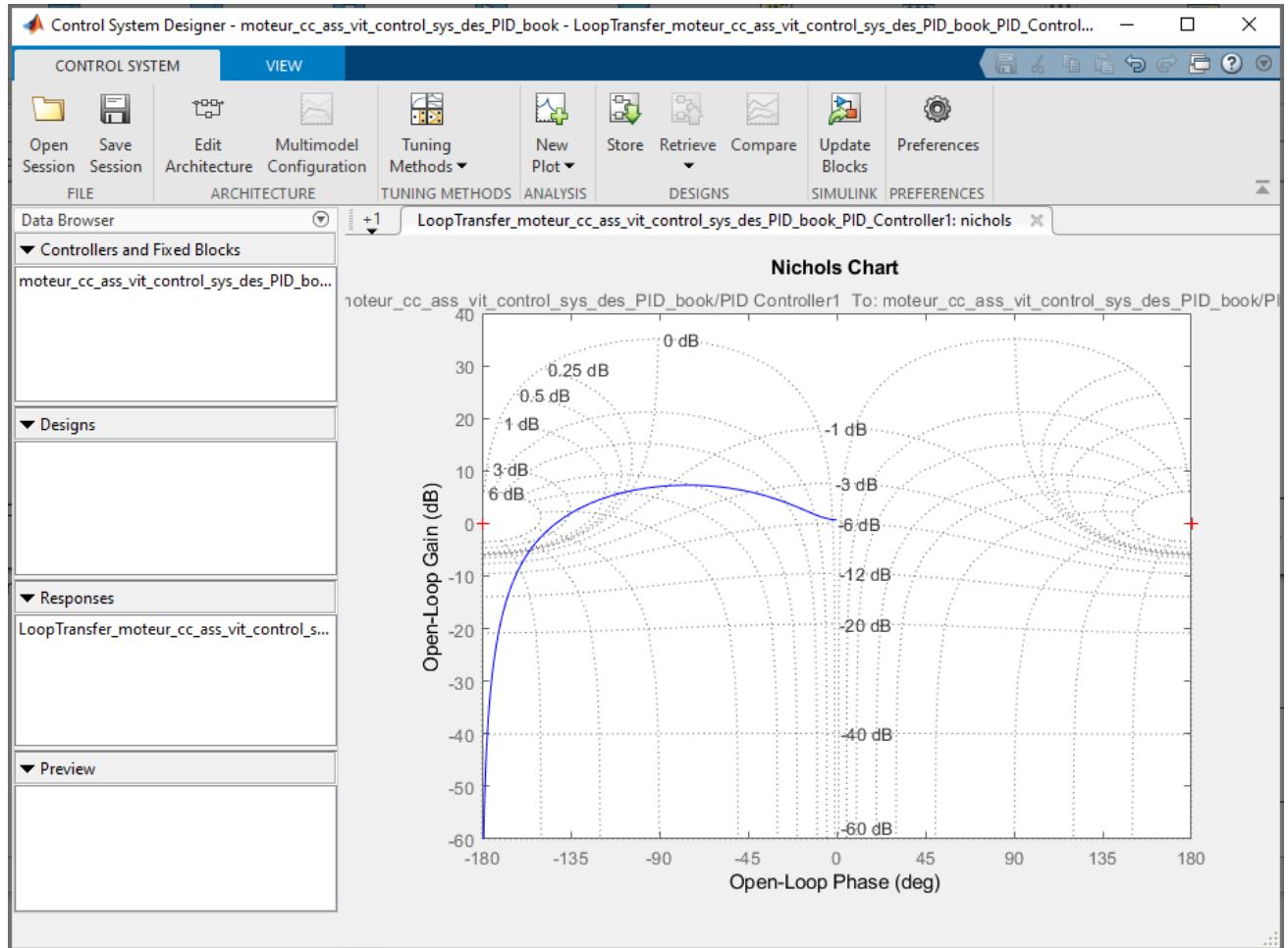


Figure 549 : diagramme de Black de la FTBO

Afin de visualiser les deux graphiques dans la même fenêtre, cliquer sur l'onglet **View** puis **Custom** pour choisir la configuration des fenêtres graphiques (Figure 550).

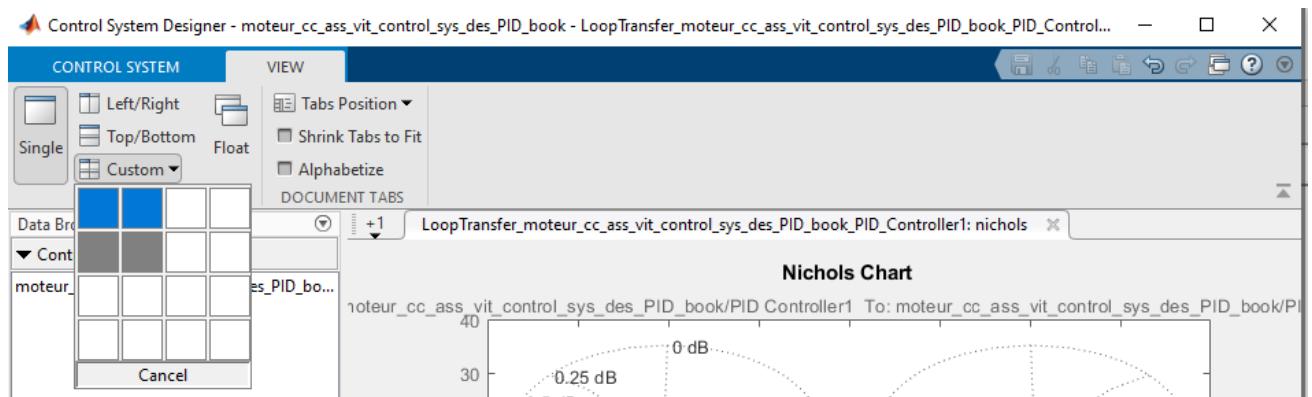


Figure 550 : choix de la configuration des fenêtres graphiques

Choisir un affichage à 4 fenêtres pour obtenir la configuration de la Figure 551.

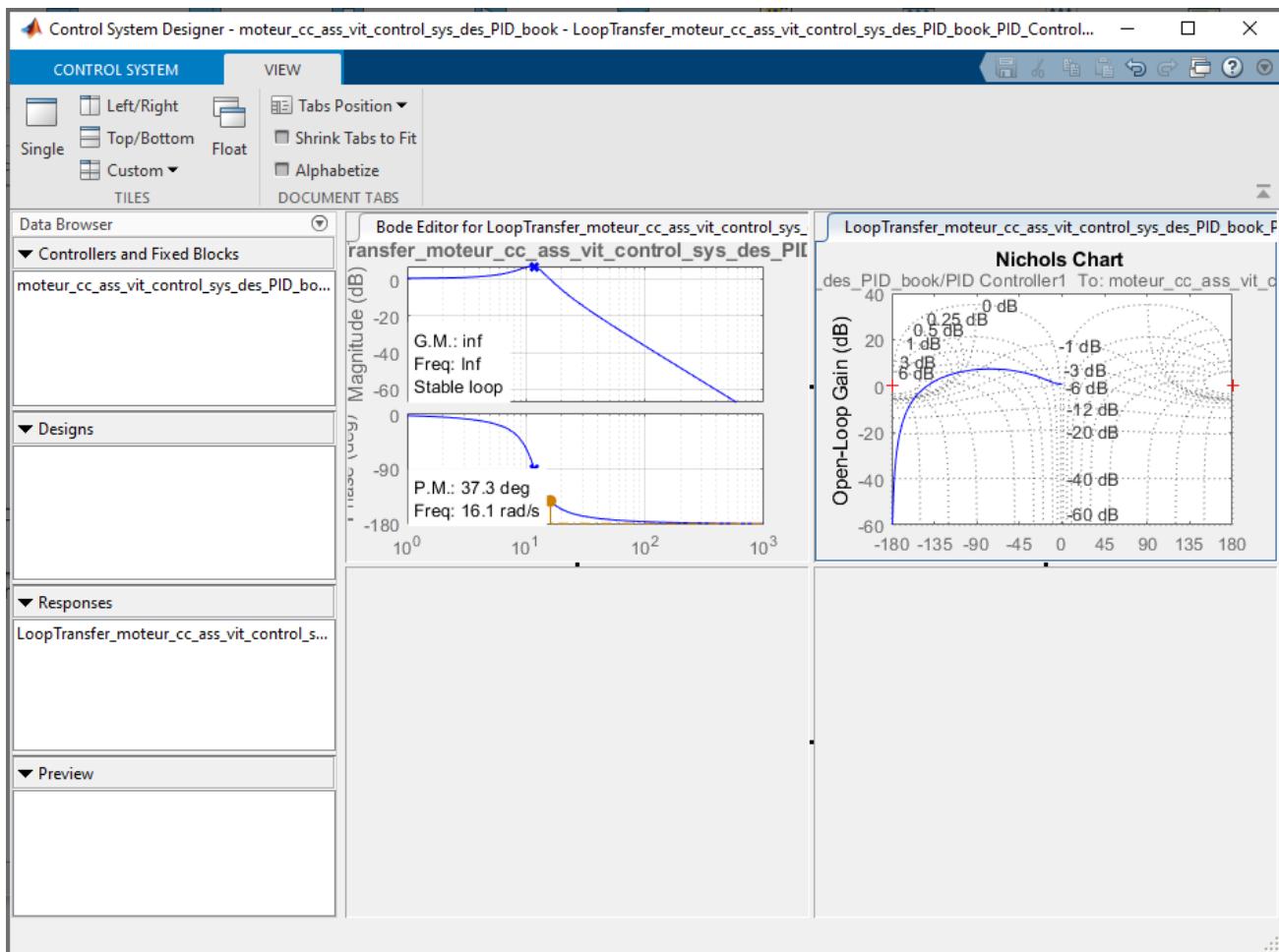


Figure 551 : modification de la configuration des fenêtres graphiques

3. Visualisation des réponses temporelles

Afin d'effectuer le réglage, il faut pouvoir visualiser les réponses temporelles vis-à-vis de la consigne et vis-à-vis de la perturbation.

Pour cela nous allons choisir les points de linéarisation pour obtenir ces deux réponses.

Cliquer sur New Plot/New Step.

Dans la fenêtre **New Step to Plot**, sélectionner **New Input-Output Transfer Response** (Figure 552).

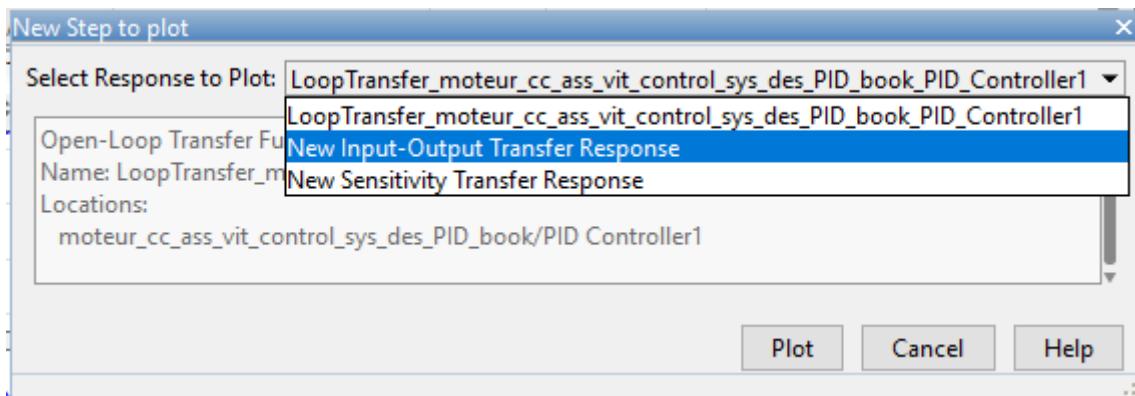


Figure 552 : choix des points de linéarisation

Dans la fenêtre qui apparaît, nommer la fonction de transfert dans le champ **Response Name** '**FTBF_consigne**'.

Dans le champ **Specify input Signals** cliquer sur **Add signal to list** et Choisir **Select Signal from Model**.

Dans le modèle Simulink de l'asservissement, cliquer sur le point d'entrée correspondant à la consigne de l'asservissement. Puis cliquer sur **Add signal** dans la fenêtre **New Step to Plot**. Cliquer alors sur le point d'entrée correspondant à la fonction de transfert en boucle fermée vis-à-vis de la consigne.

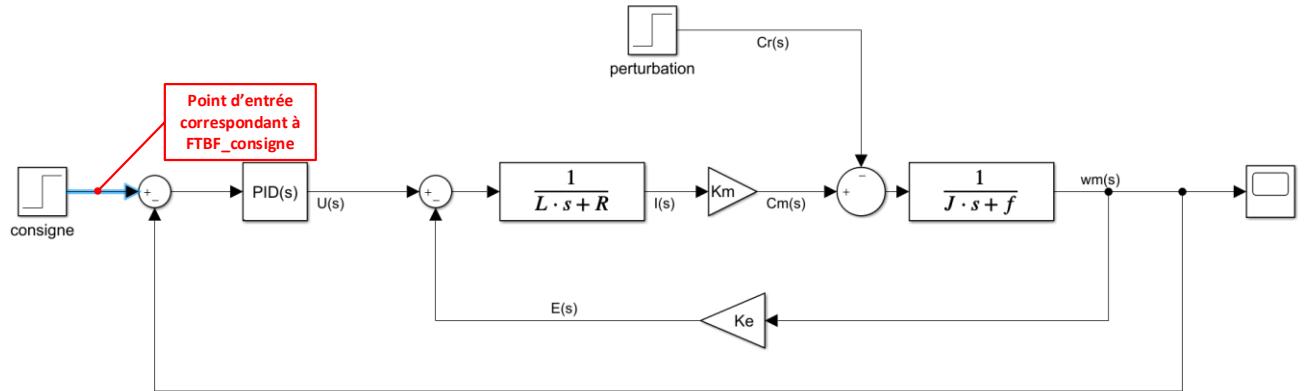


Figure 553 : choix du point d'entrée de la fonction de transfert

Cliquer alors sur **Add signal(s)** dans la fenêtre **New Step to Plot** pour valider votre choix.

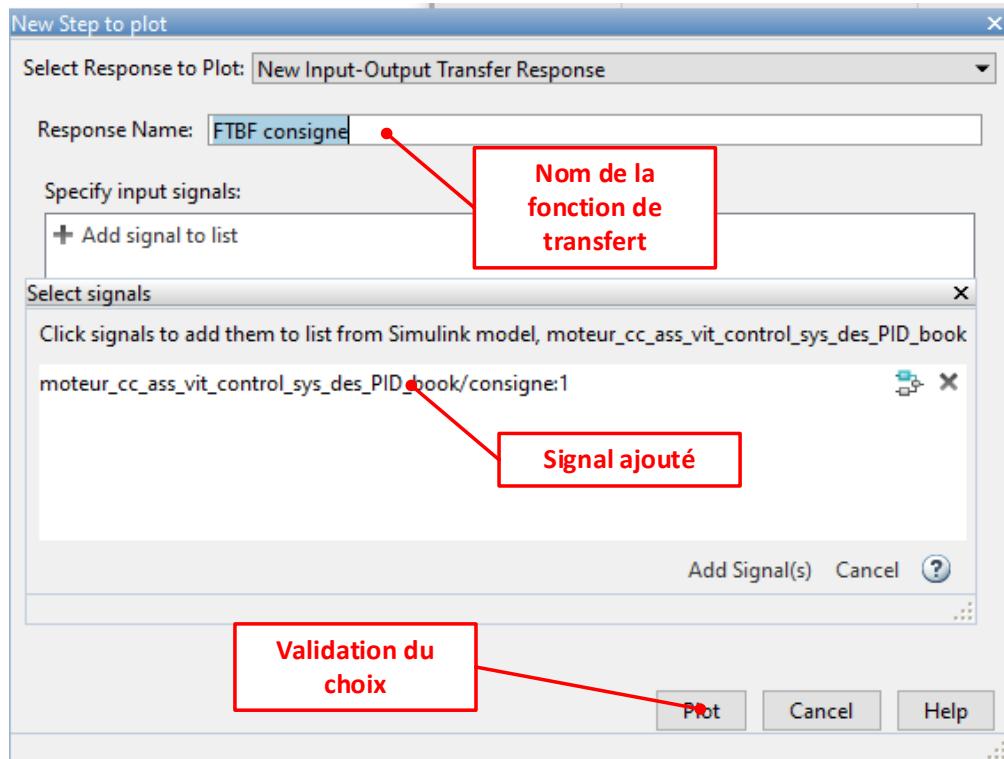


Figure 554 : fenêtre New Step to Plot

Recommencer cette opération avec le champ **Specify output Signal** pour choisir le point de sortie de la fonction de transfert et valider votre choix.

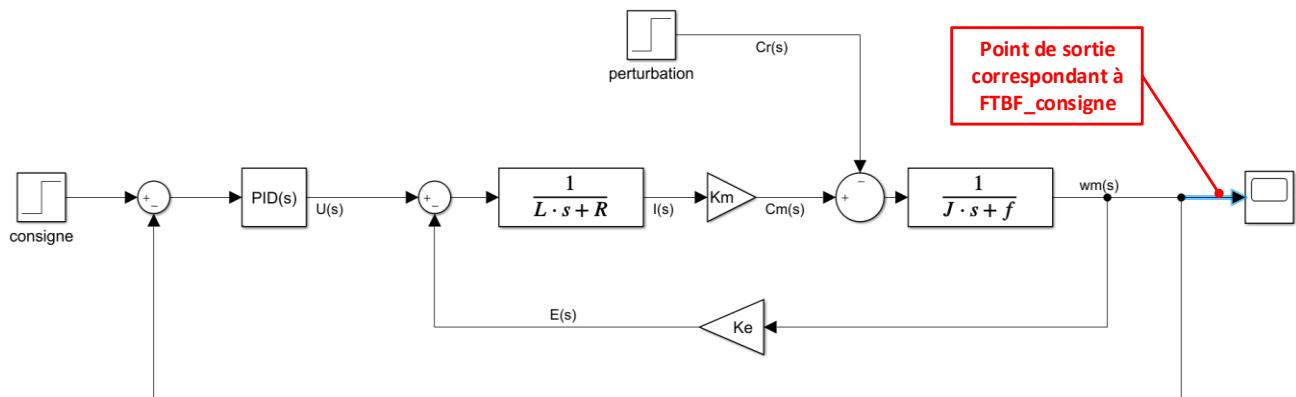


Figure 555 : choix du point de sortie de la fonction de transfert

Vous devez obtenir la configuration de la Figure 556.

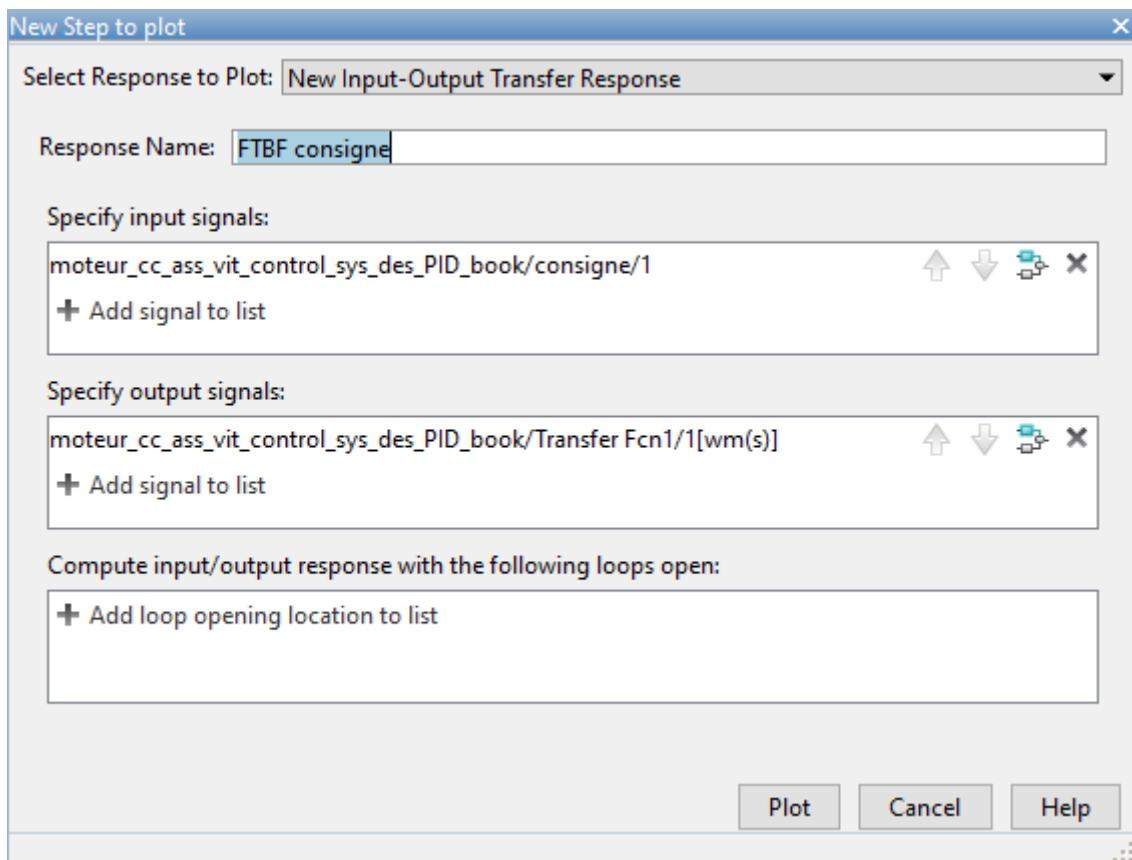
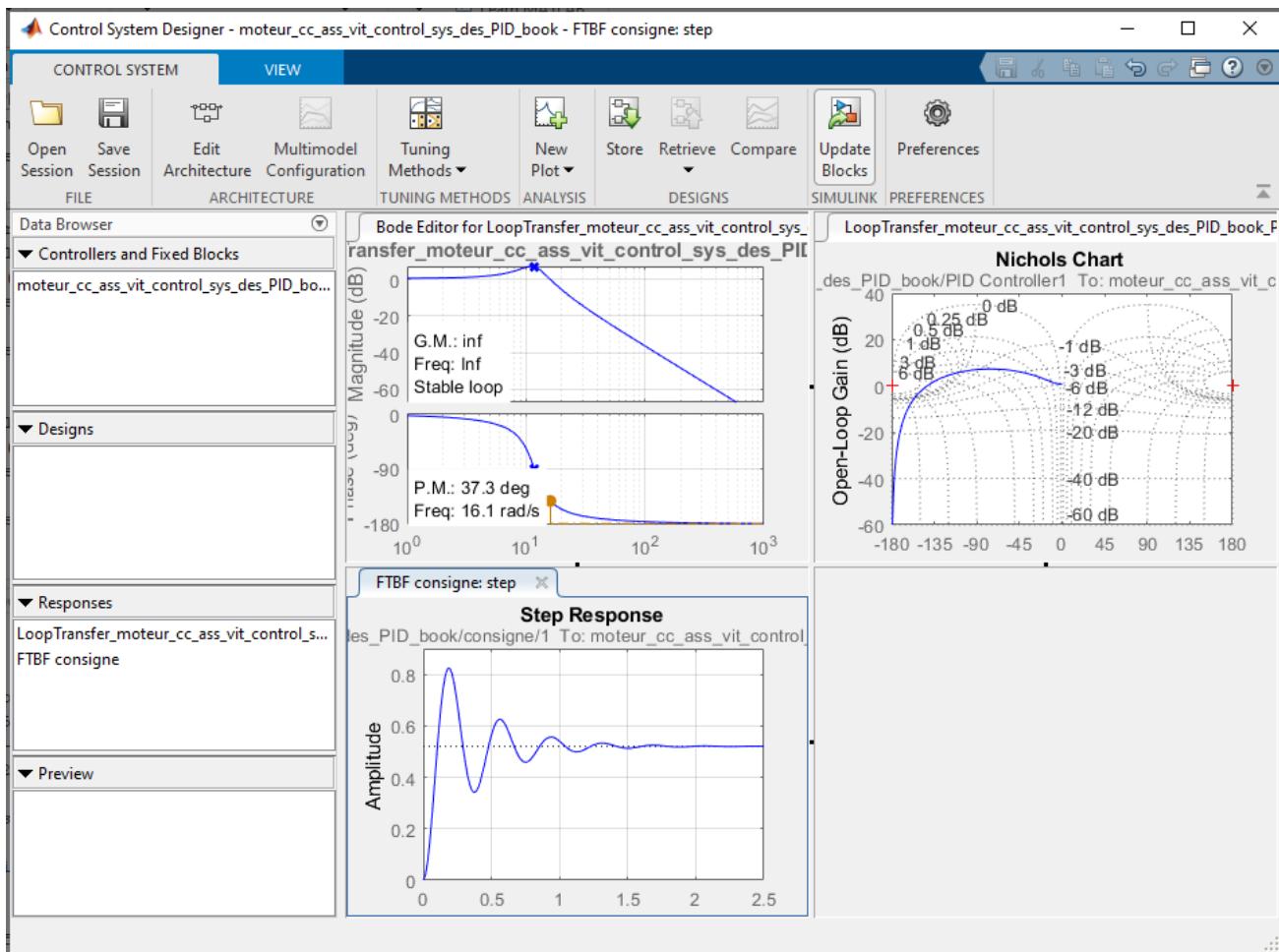


Figure 556 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la consigne

Cliquez alors sur **Plot** pour tracer la réponse temporelle souhaitée.



Recommencer cette opération pour obtenir la réponse de la fonction de transfert en boucle fermée vis-à-vis de la perturbation.

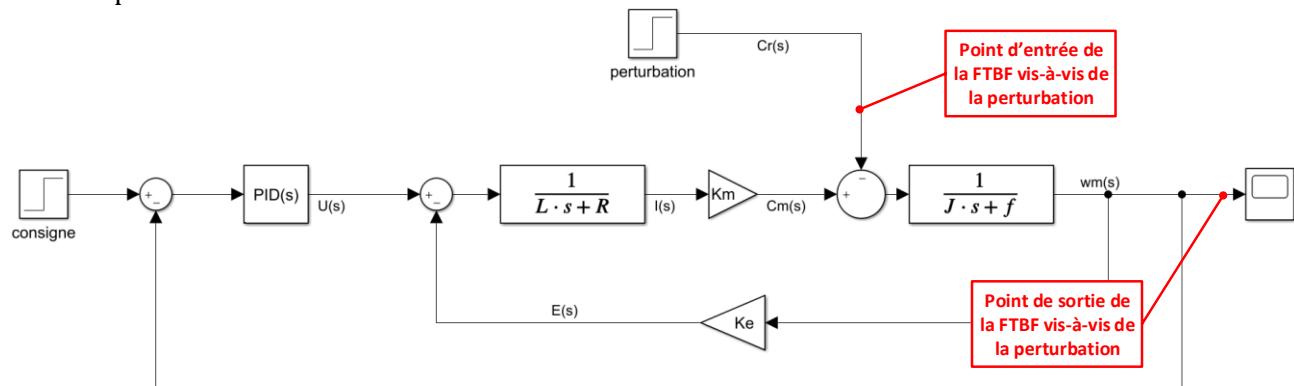


Figure 557 : point d'entrée et de sortie pour la réponse temporelle vis-à-vis de la perturbation

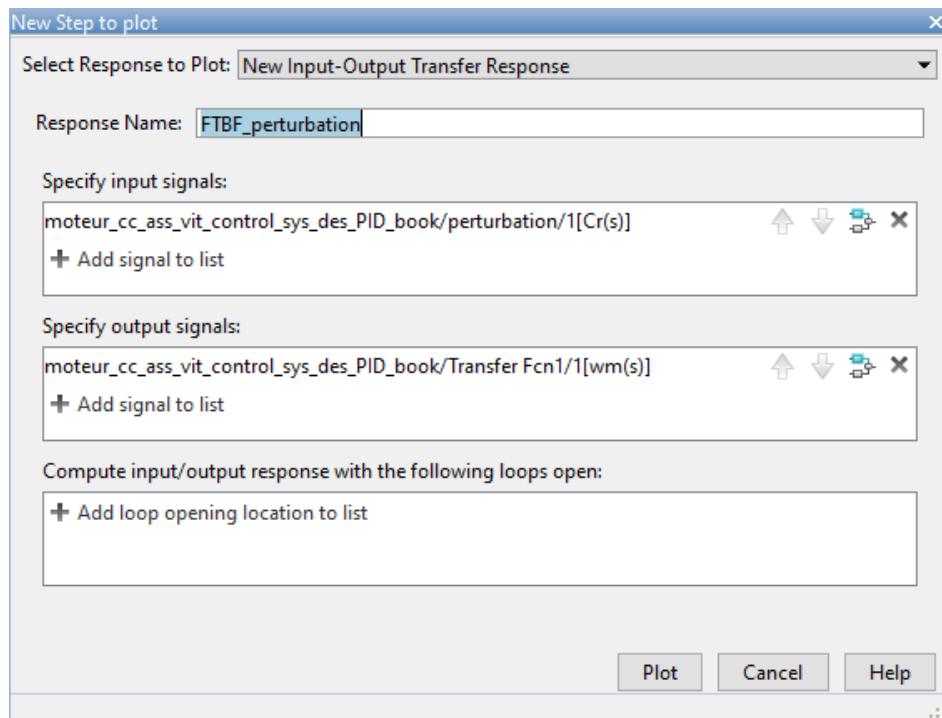


Figure 558 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la perturbation

Vous devez obtenir la configuration de la Figure 559.

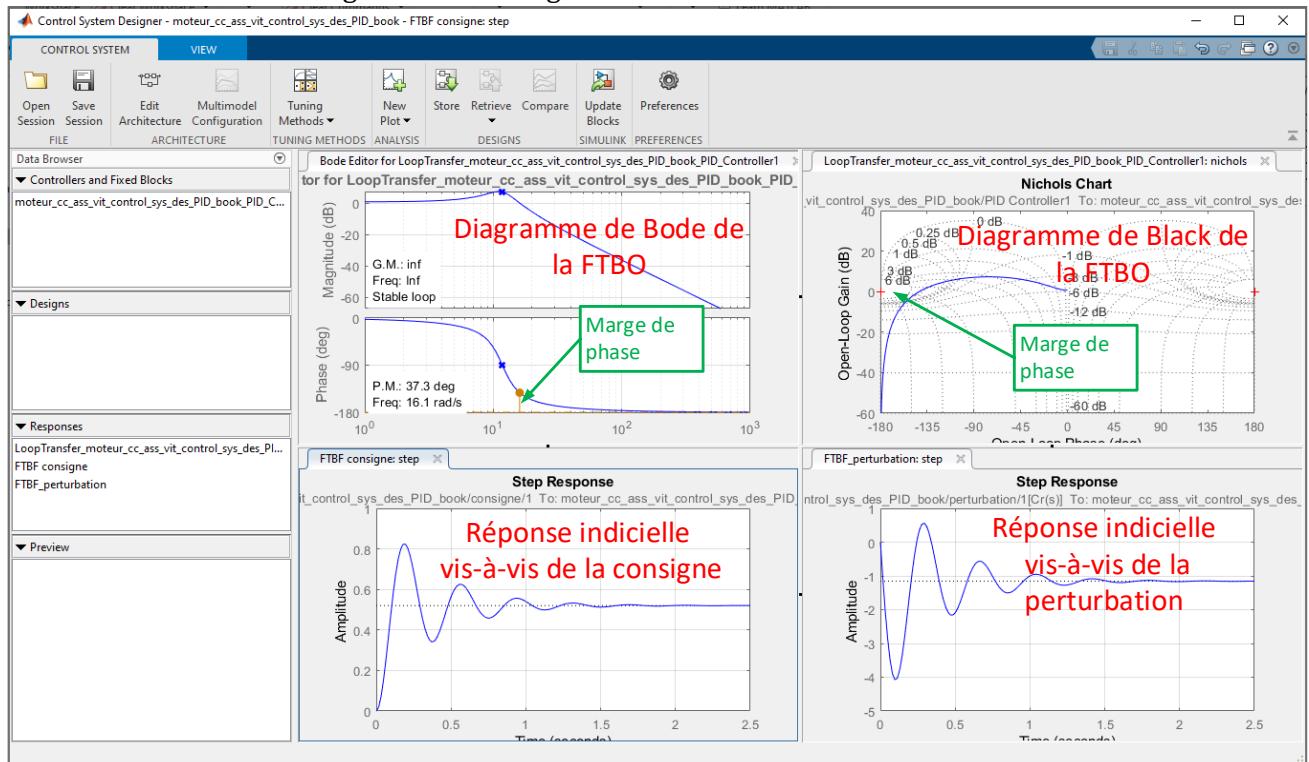


Figure 559 : visualisation des fenêtres utiles pour effectuer le réglage

4. Réglage du PID

Dans la fenêtre graphique du diagramme de Bode de la FTBO, cliquer avec le bouton droit et choisir **Edit Compensator** pour ouvrir la fenêtre de réglage du PID.

Sélectionner ensuite l'onglet **Parameter** pour pouvoir agir sur les paramètres de réglage du PID.

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 diagrammes.

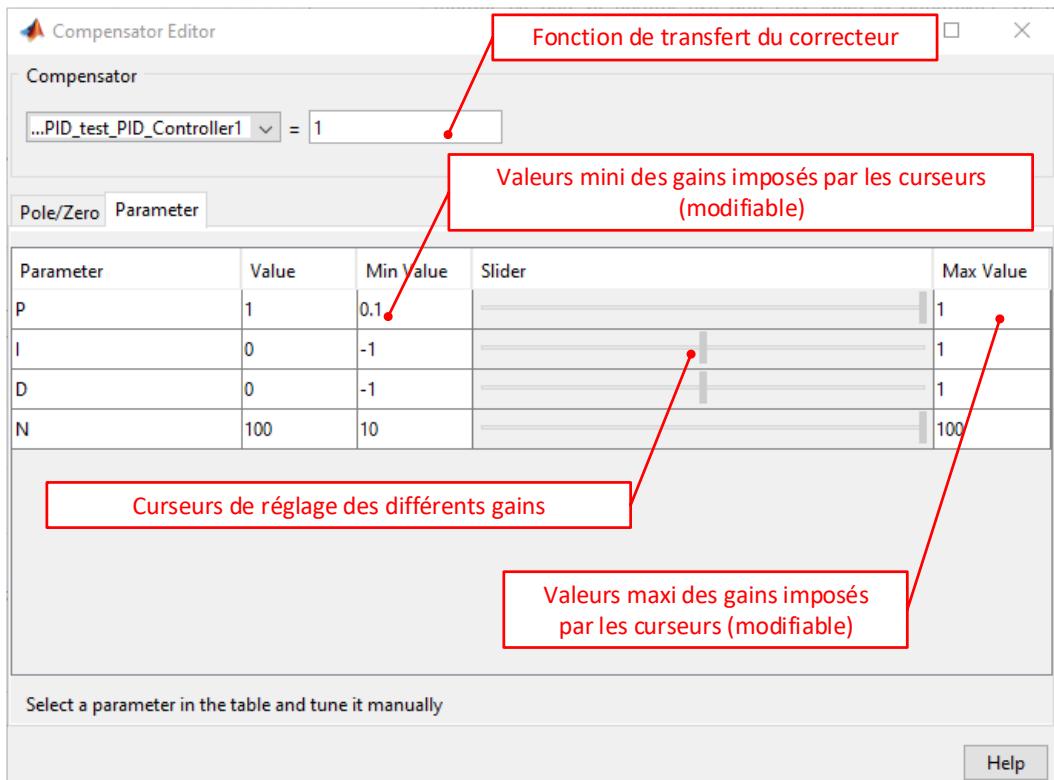


Figure 560 : les fonctions de la fenêtre de réglage du PID

5. 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 des réponses indicielles.

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF vis-à-vis de la consigne et choisir **Properties**, onglet **Options** puis définir que le critère de rapidité est le temps de réponse à 5%.

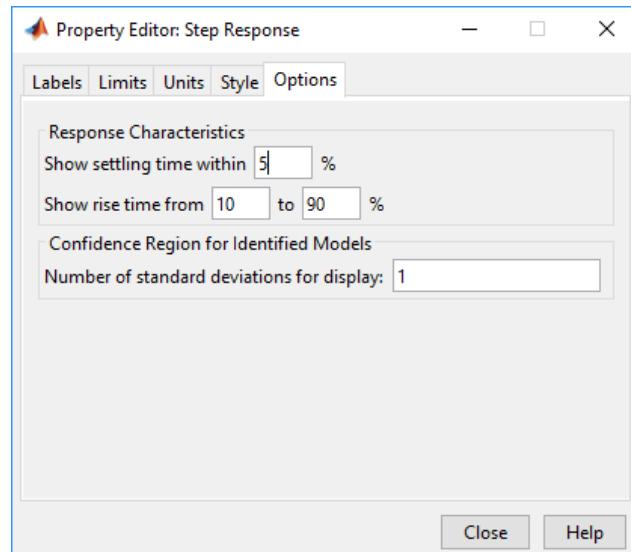


Figure 561 : définition des critères de rapidité

Clique 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% (Figure 562).

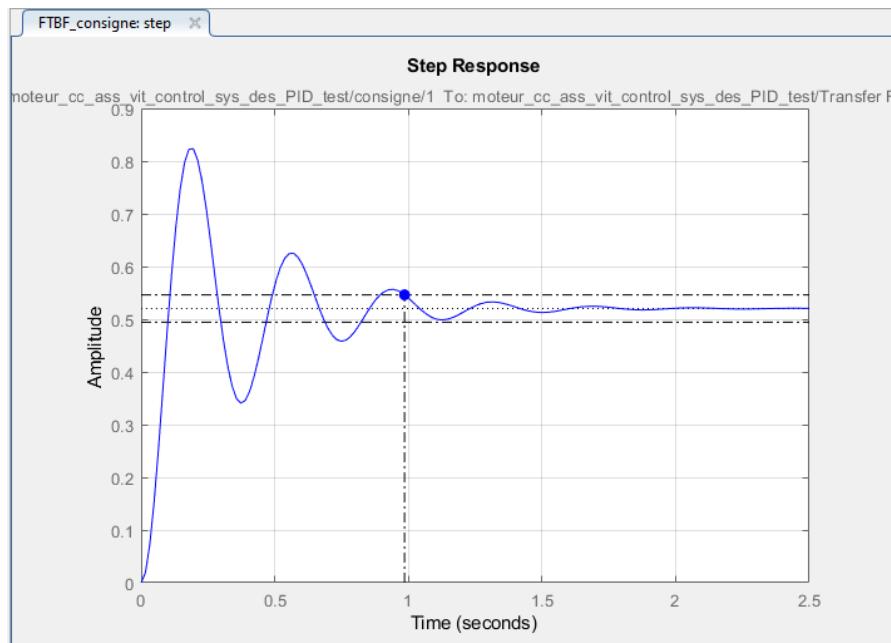


Figure 562 : visualisation du temps de réponse à 5%

Clique 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 563.

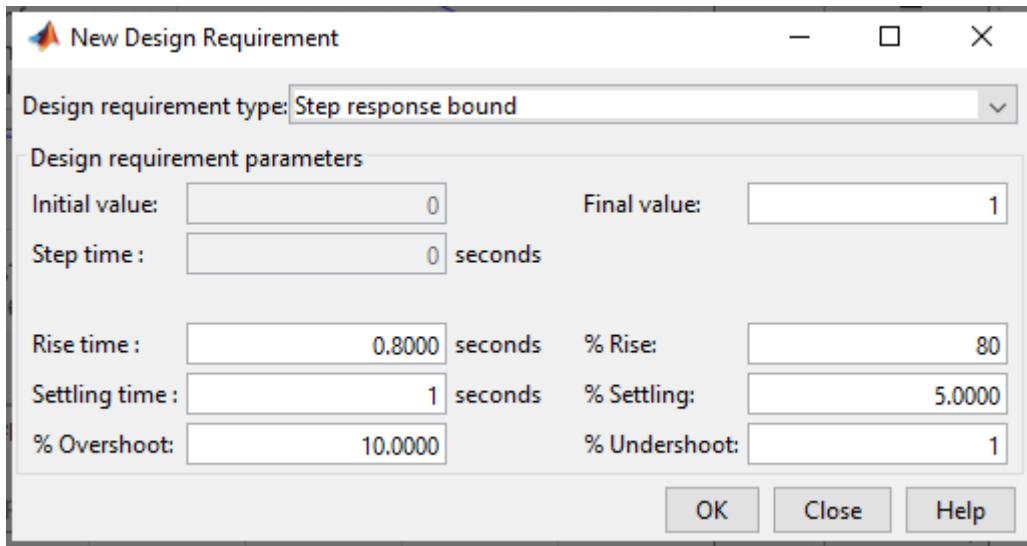


Figure 563 : définition des critères de performances

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 564).

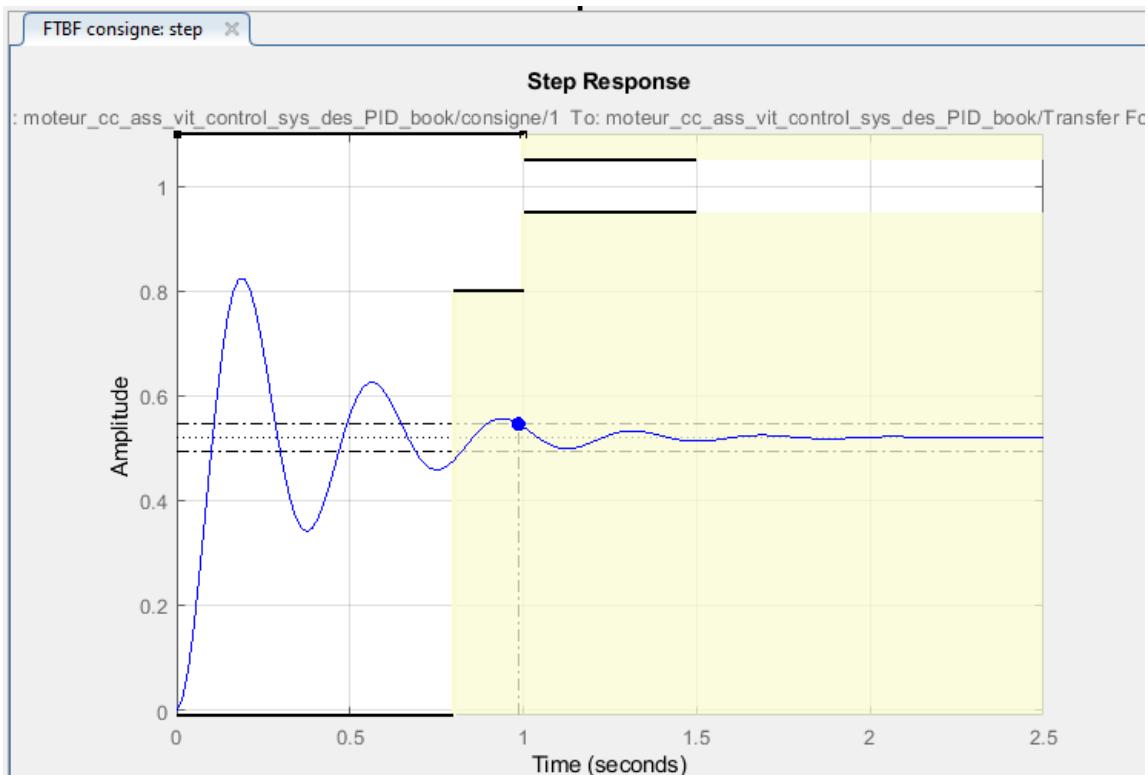


Figure 564 : visualisation des critères de performances dans la fenêtre graphique

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.

6. 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 La fenêtre de la Figure 565 montre un réglage satisfaisant du PID.

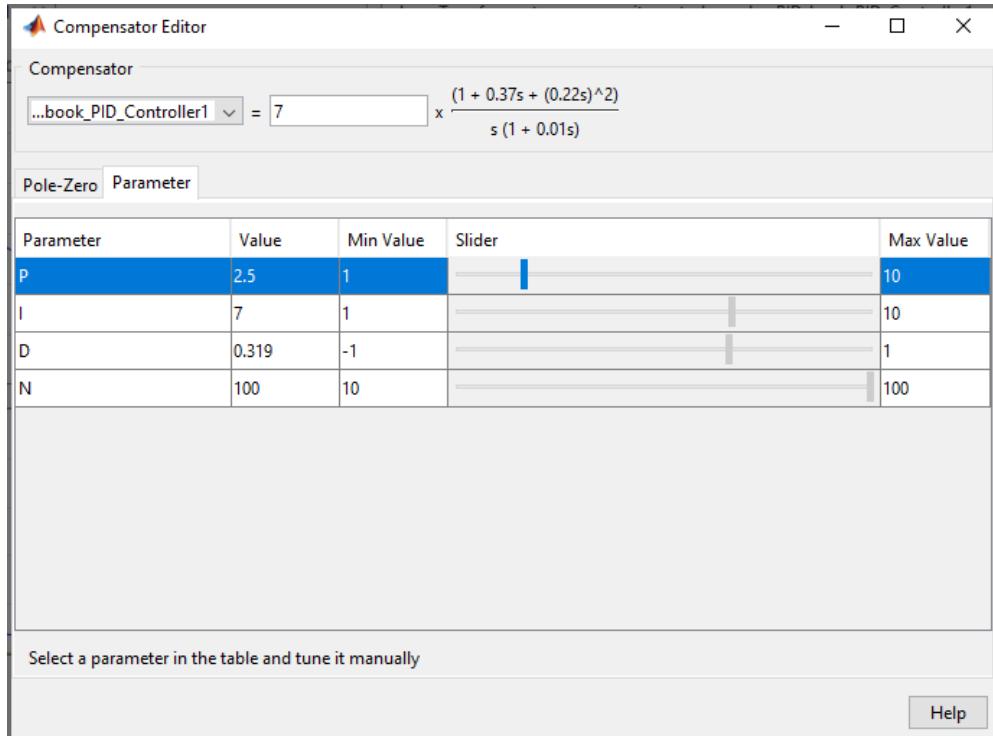


Figure 565 : réglage satisfaisant le cahier des charges du PID

La fenêtre de la Figure 566 montre la réponse indicelle corrigée de la FTBF.

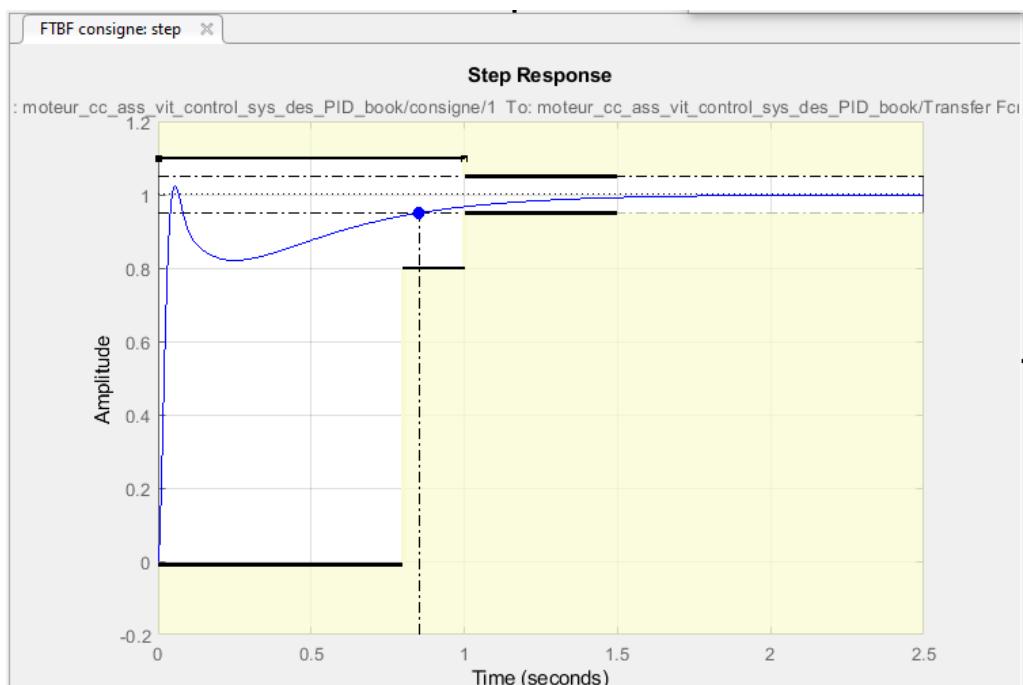


Figure 566 : résultat du réglage manuel du PID sur la réponse temporelle

Le réglage étant satisfaisant vis-à-vis de la consigne, nous pouvons maintenant vérifier le comportement vis-à-vis de la perturbation.

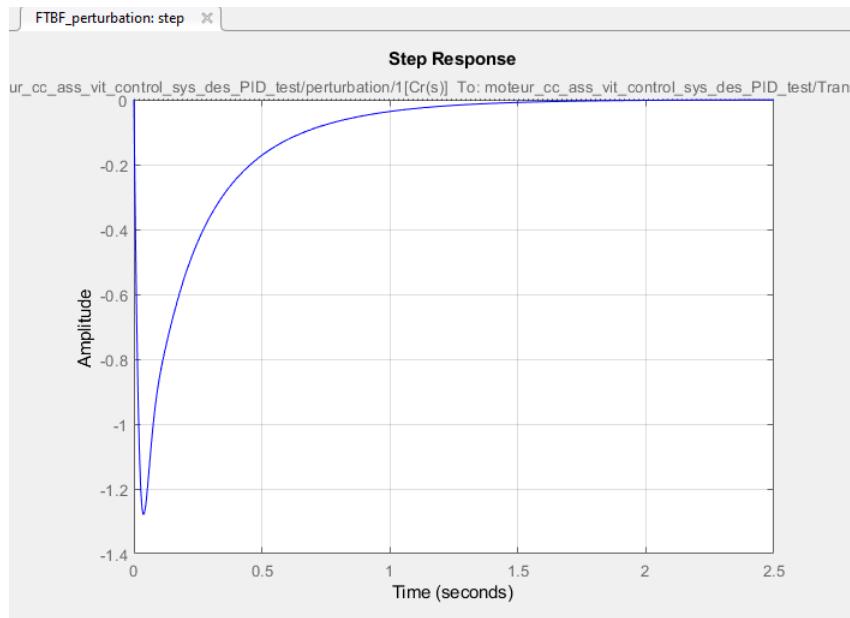


Figure 567 : réponse indicielle du système vis-à-vis d'un échelon de perturbation

Nous constatons que la perturbation est rapidement rejetée et que son influence peut être considérée comme négligeable au bout de 0.8 s . Le réglage effectué est donc satisfaisant.

Les diagrammes de Bode et de Black de la FTBO permettent de visualiser l'influence de ces réglages et d'apprécier les critères de marge de gain et de marge de phase.

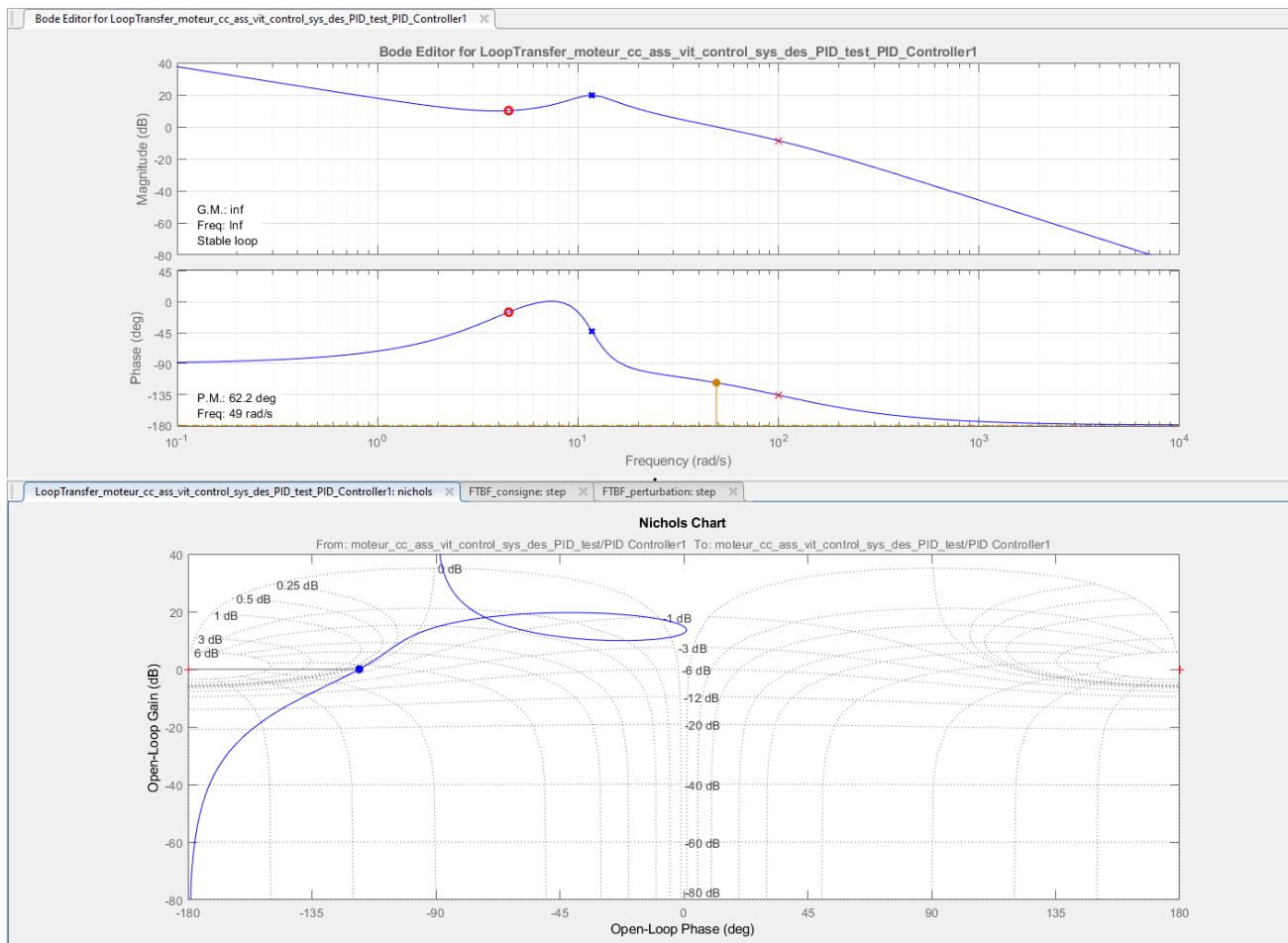


Figure 568 : influence des réglages effectués sur les diagrammes fréquentiels de la FTBO

7. Exportation du réglage dans le modèle Simulink

Cliquer maintenant sur **Update Block** dans la fenêtre de commande du **Control System Designer** pour exporter les réglages effectués dans le PID du modèle Simulink.

Retourner dans le modèle Simulink puis **Lancer** la simulation et visualiser la réponse du moteur avec le PID réglé.

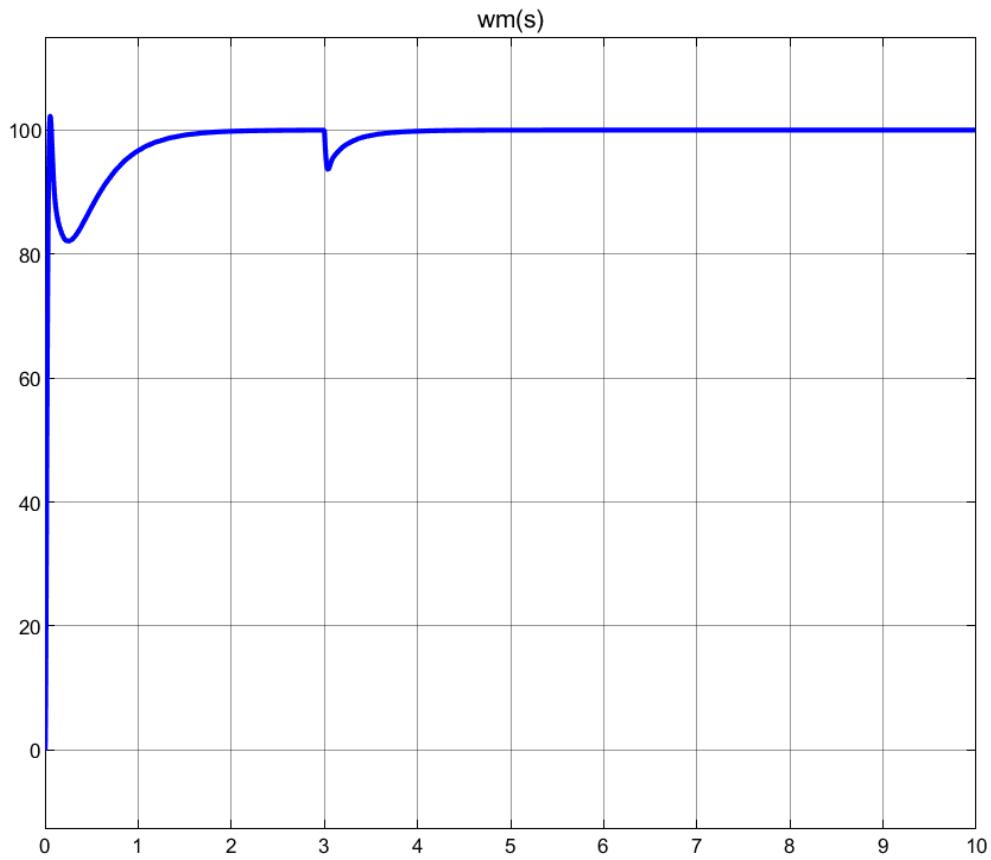


Figure 569 : réponse temporelle après réglage du PID

Nous constatons sur la Figure 569 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 570). 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.

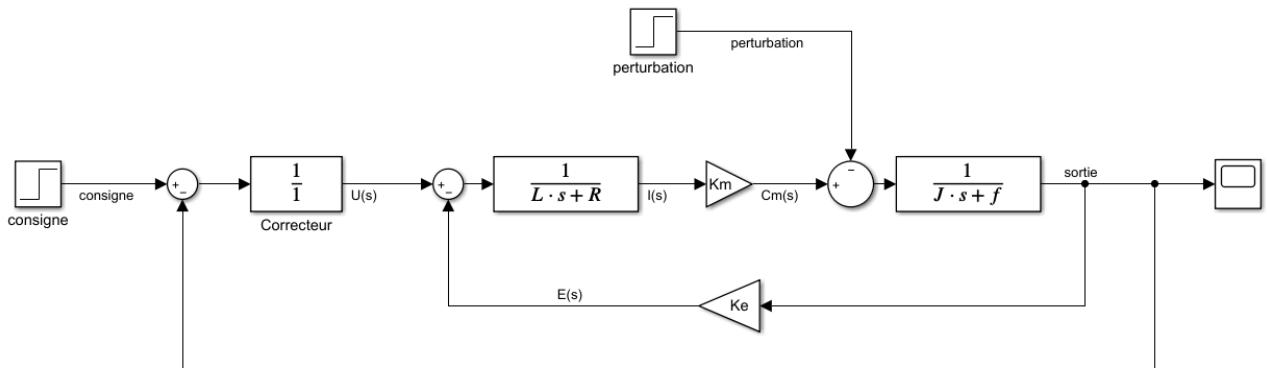


Figure 570 : modèle de l'asservissement du moteur à continu avec correction par fonction de transfert à régler

Exécuter le script ***parametres_moteur_cc.m***.

Lancer la simulation et **visualiser** la réponse dans le scope.

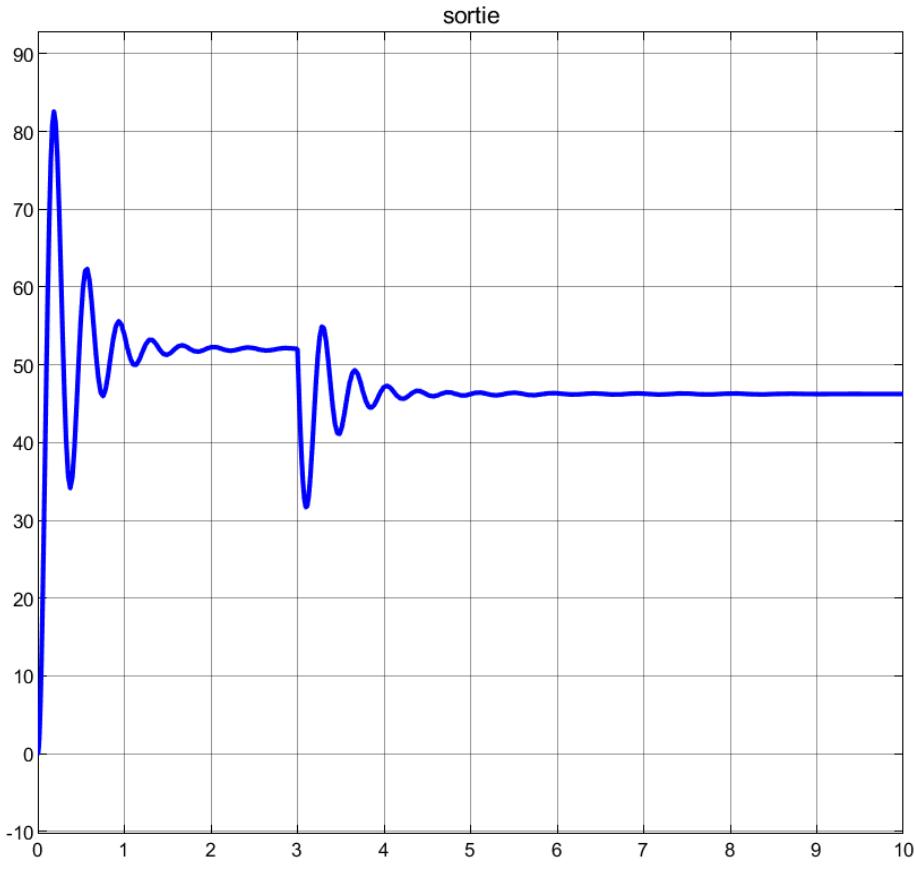


Figure 571 : réponse du système non corrigé

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 l'onglet APPS de la fenêtre de commande de MATLAB, sélectionner l'application « Control System Designer » (Figure 572).

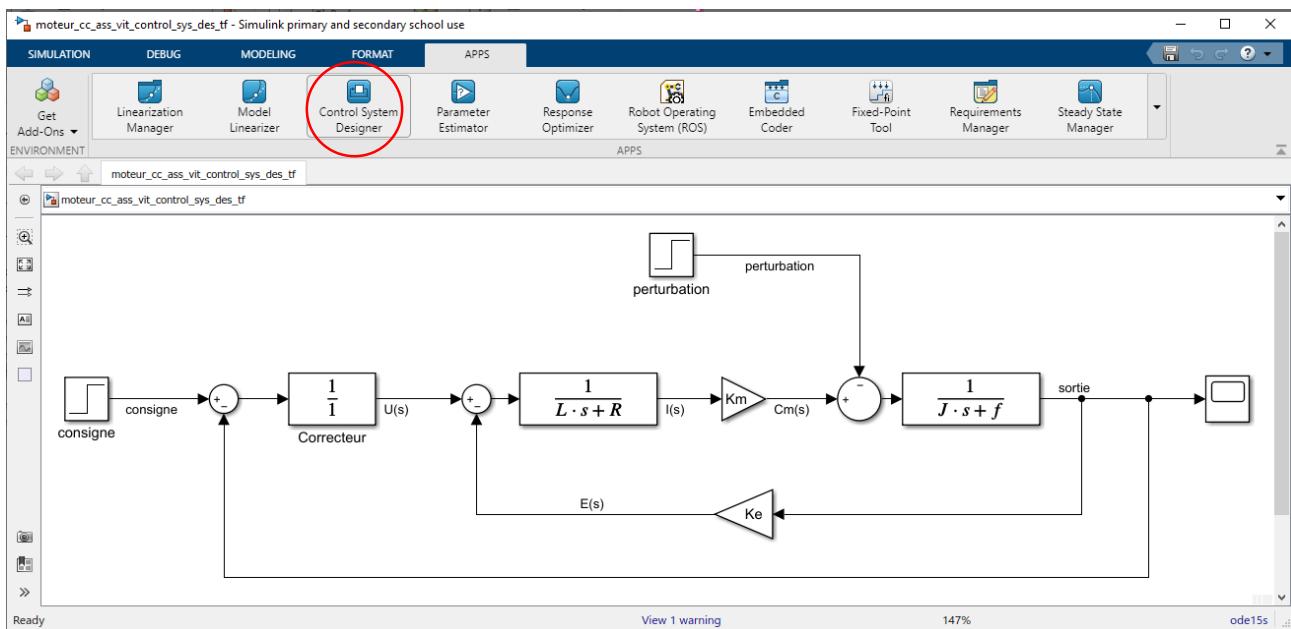


Figure 572 : lancement de l'application « Control System Designer »

Cette commande entraîne l'ouverture de la fenêtre **Control System Designer** qui propose de choisir les blocs que l'on veut régler, de visualiser les points d'entrée et de sortie de la boucle fermée.

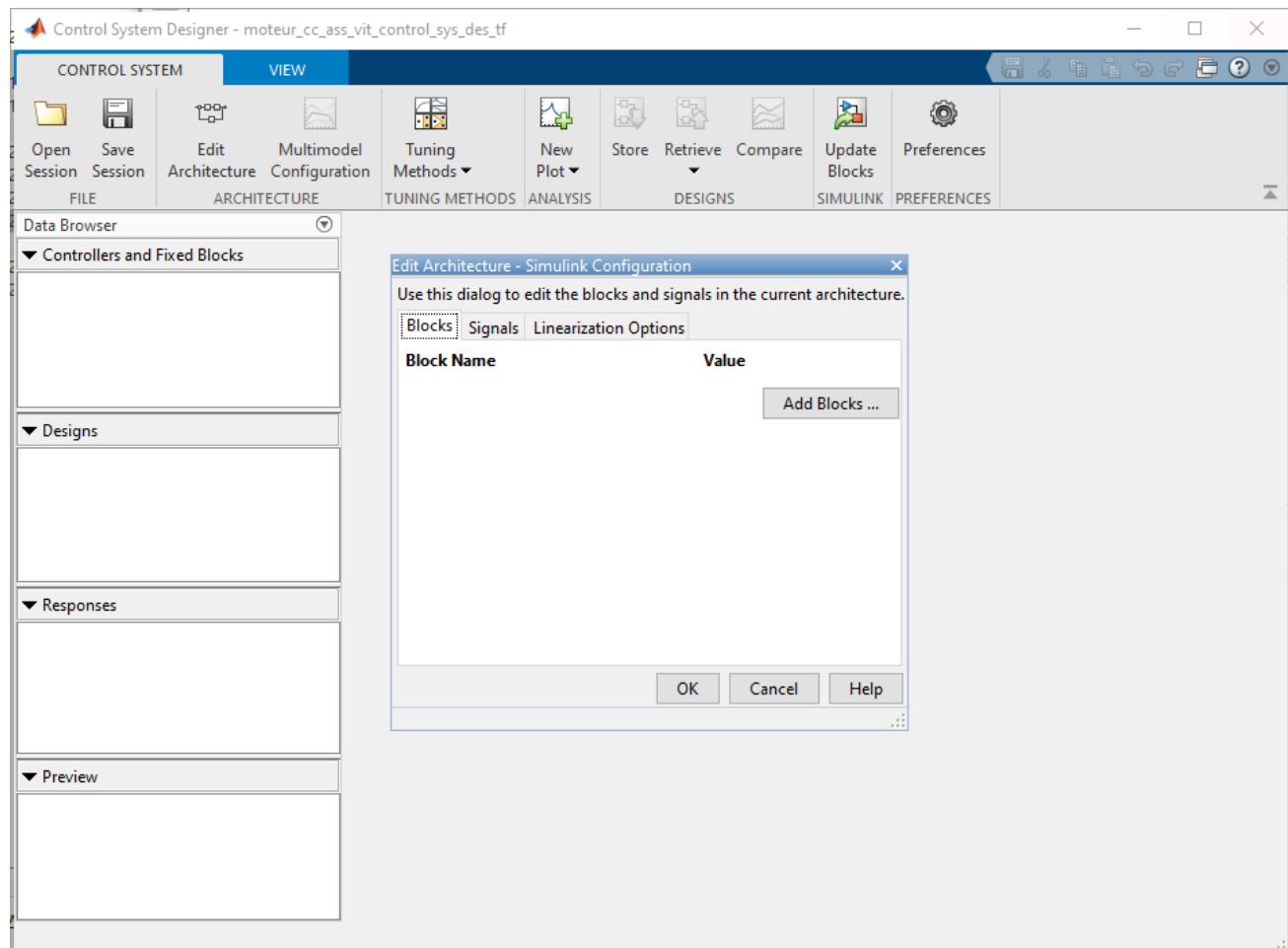


Figure 573 : Control System Designer

Afin de sélectionner le bloc à régler cliquer sur **Add Blocks** et choisir **Correcteur** comme bloc à régler, puis cliquer sur **OK**. (Figure 574).

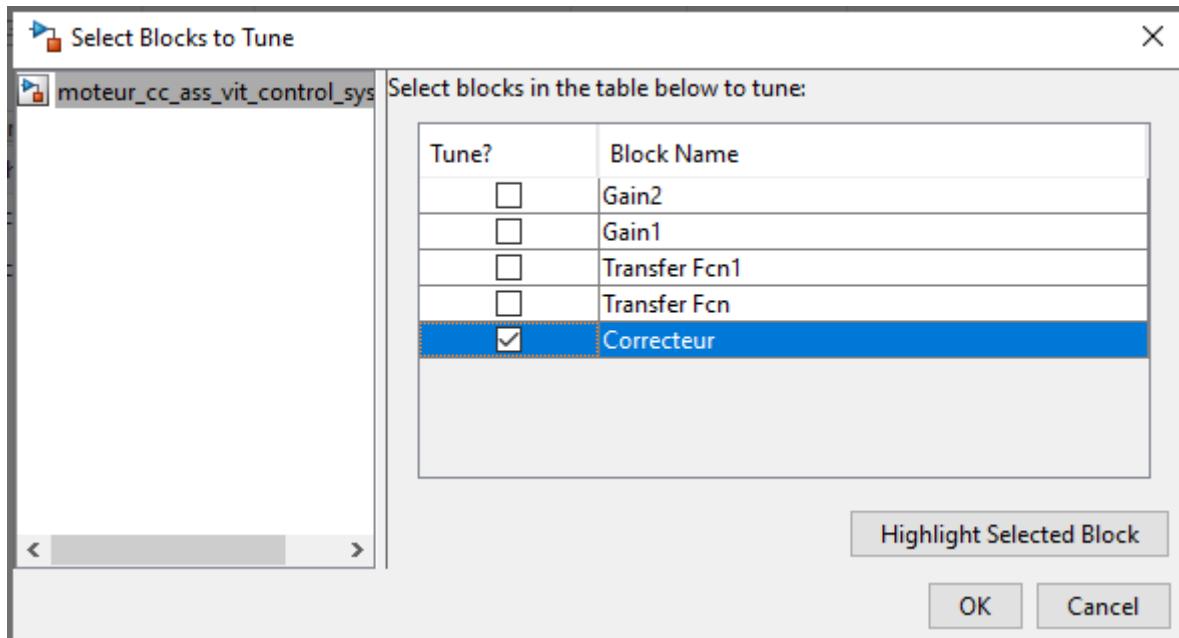


Figure 574 : choix du bloc à régler

Dans la fenêtre **Edit Architecture** (Figure 575) qui apparaît, cliquer sur OK pour valider votre choix.

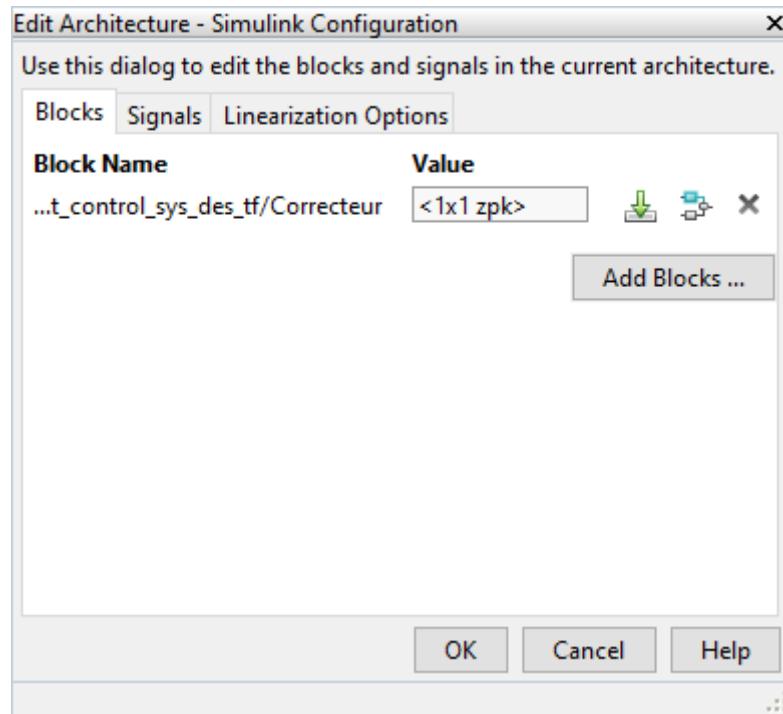


Figure 575 : fenêtre Edit Architecture

1. Diagrammes fréquentiels de la FTBO

Nous allons maintenant choisir la méthode de réglage. Dans la fenêtre du Control System Designer, sélectionner **Tuning Method/Bode Editor**.

La fenêtre **Select Response to Edit** apparaît. La fonction de transfert en boucle ouverte (FTBO) est automatiquement détectée. Les points de linéarisation sont indiqués (Figure 576).

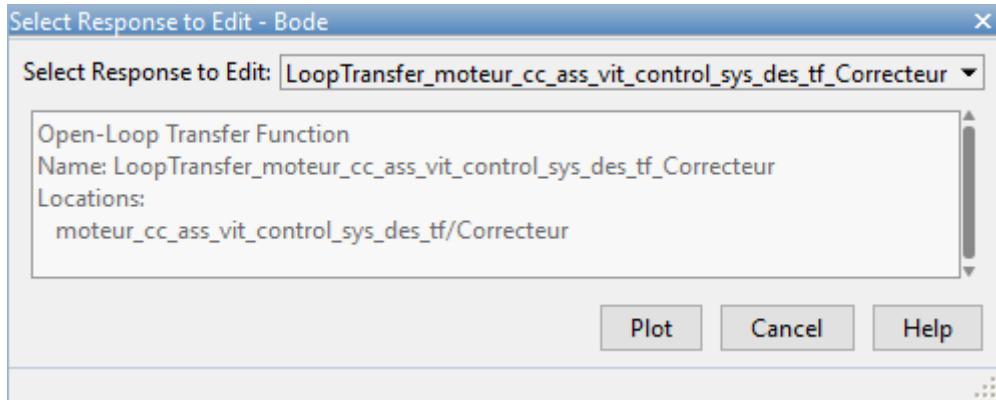


Figure 576 : choix des points de linéarisation pour la FTBO

Cliquer sur **Plot** pour tracer le diagramme de Bode de la FTBO. Pour ajouter un quadrillage, cliquer avec le bouton droit dans la fenêtre du diagramme de Bode et choisir **Grid**.

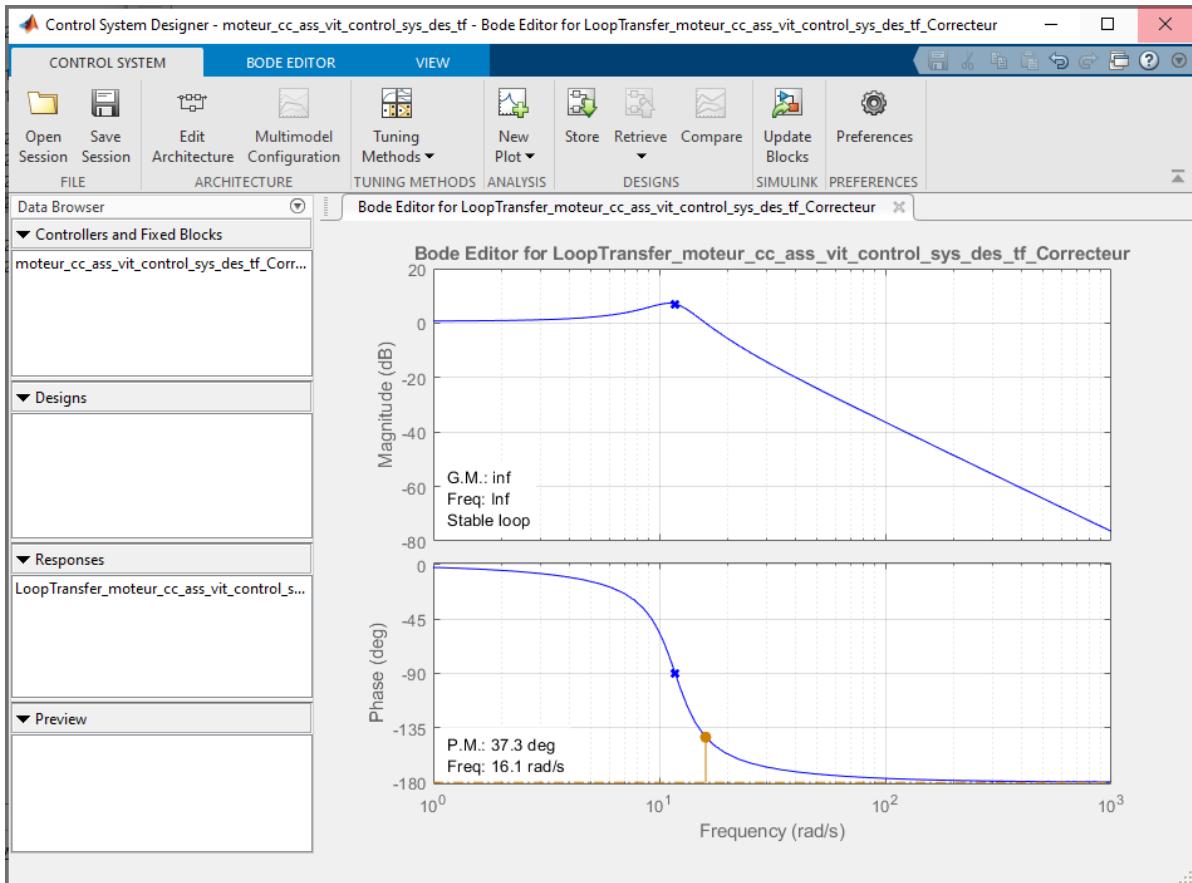


Figure 577 : diagramme de Bode de la FTBO

Il est également possible d'ajouter d'autres types de diagramme pour la FTBO.

Cliquer sur **New Plot/New Nichols** puis cliquer sur **Plot** pour faire apparaître un diagramme de Black.

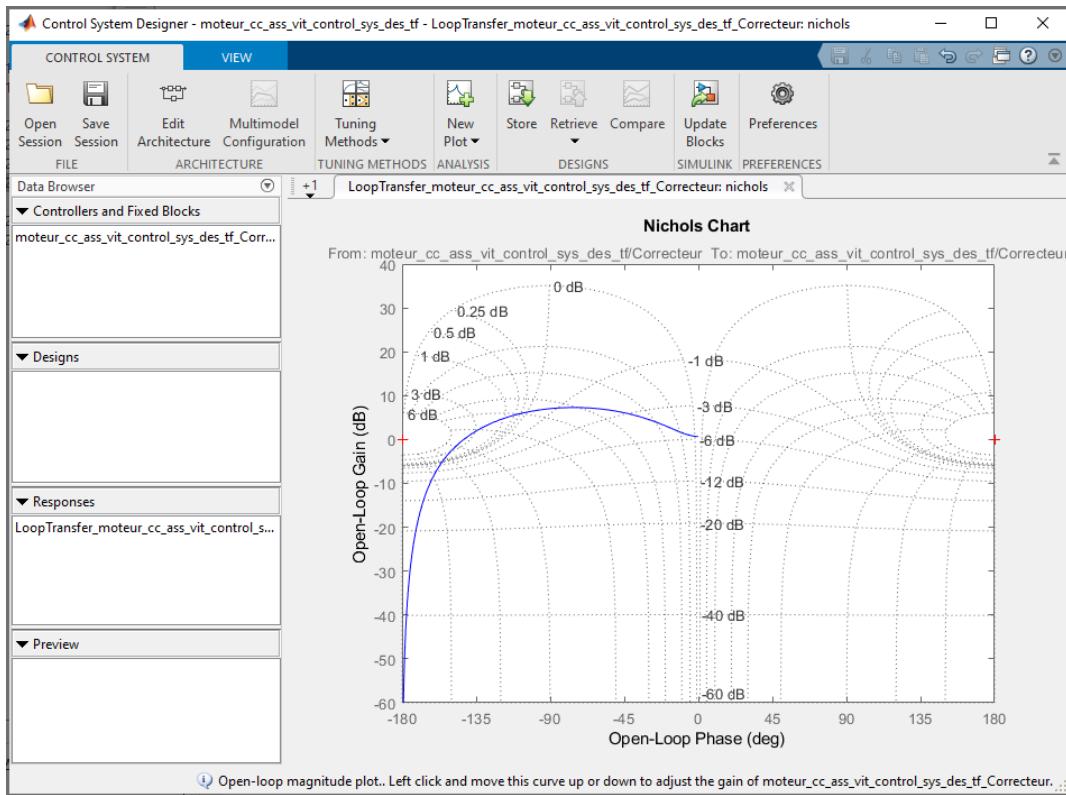


Figure 578 : diagramme de Black de la FTBO

Afin de visualiser les deux graphiques dans la même fenêtre, cliquer sur l'onglet **View** puis **Custom** pour choisir la configuration des fenêtres graphiques (Figure 579).

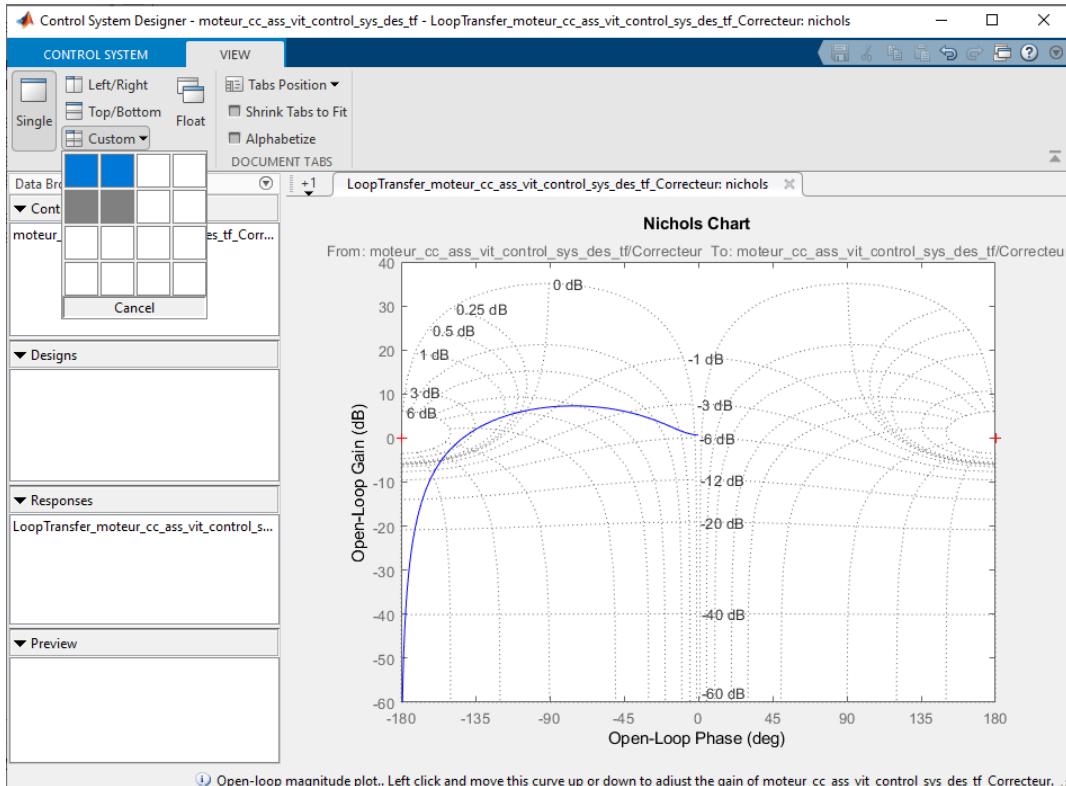


Figure 579 : choix de la configuration des fenêtres graphiques

Choisir un affichage à 4 fenêtres pour obtenir la configuration de la Figure 580.

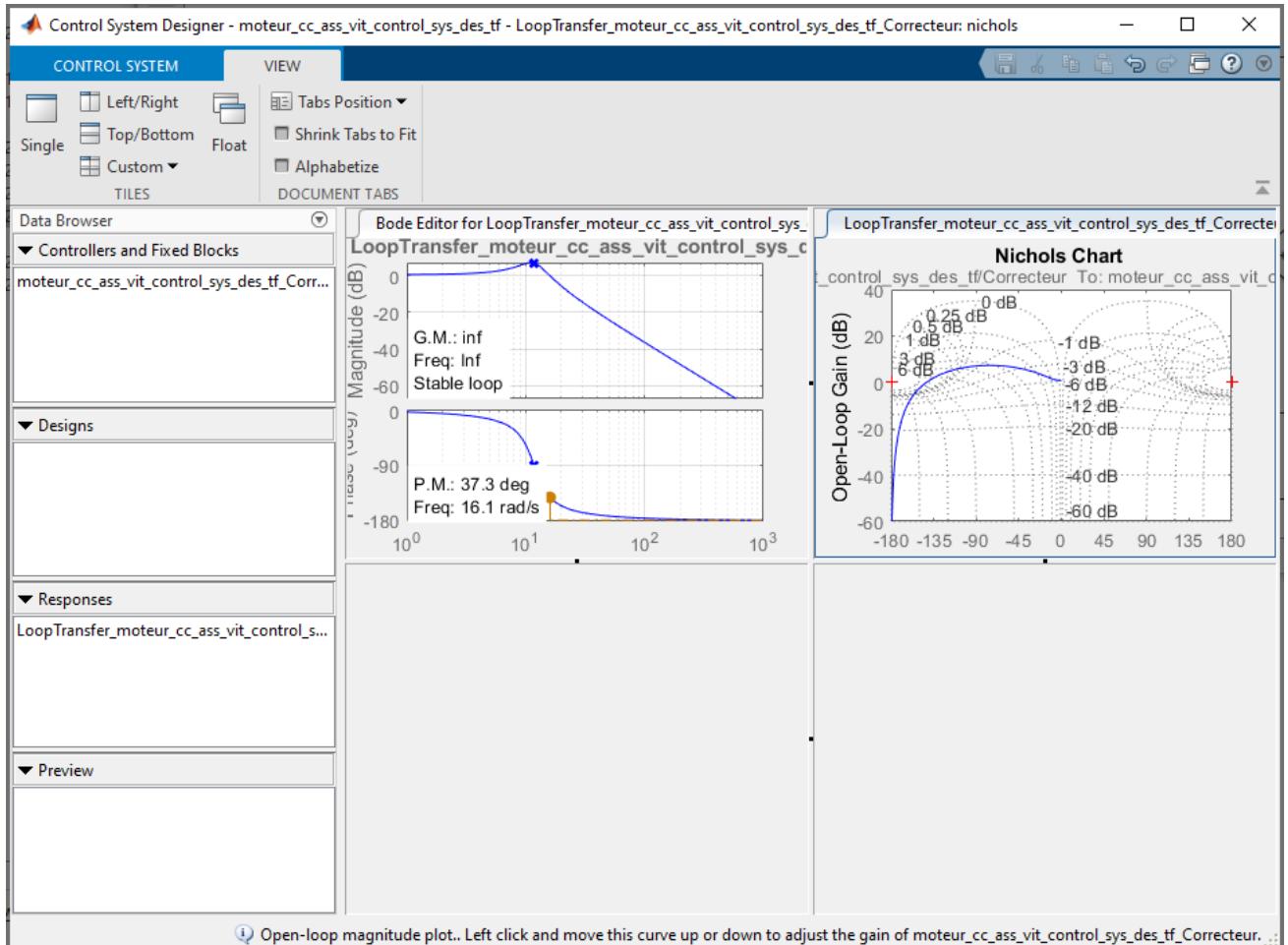


Figure 580 : modification de la configuration des fenêtres graphiques

2. Visualisation des réponses temporelle et fréquentielle de la FTBF

Afin d'effectuer le réglage, il faut pouvoir visualiser les réponses temporelles et fréquentielle de la FTBF vis-à-vis de la consigne. Nous allons donc afficher :

- La réponse indicielle de la FTBF vis-à-vis de la consigne
- Le diagramme de Bode de la FTBF vis-à-vis de la consigne afin de visualiser les éventuels problèmes de résonnance en boucle fermée qui pourraient apparaître lors des réglages.

Pour cela nous allons choisir les points de linéarisation pour obtenir ces deux réponses.

Cliquer sur **New Plot/New Step**.

Dans la fenêtre **New Step to Plot**, sélectionner **New Input-Output Transfer Response** (Figure 581).

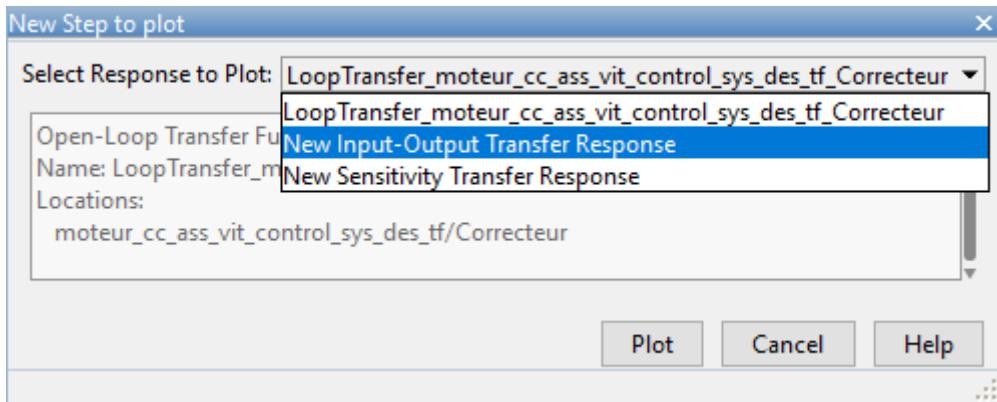


Figure 581 : choix des points de linéarisation

Dans la fenêtre qui apparaît, nommer la fonction de transfert dans le champ **Response Name** '**FTBF_consigne**' (Figure 583).

Dans le champ **Specify input Signals** cliquer sur **Add signal to list** et Choisir **Select Signal from Model**.

Dans le modèle Simulink de l'asservissement, cliquer sur le point d'entrée correspondant à la consigne de l'asservissement. Puis cliquer sur **Add signal** dans la fenêtre **New Step to Plot**. Cliquer alors sur le point d'entrée correspondant à la fonction de transfert en boucle fermée vis-à-vis de la consigne.

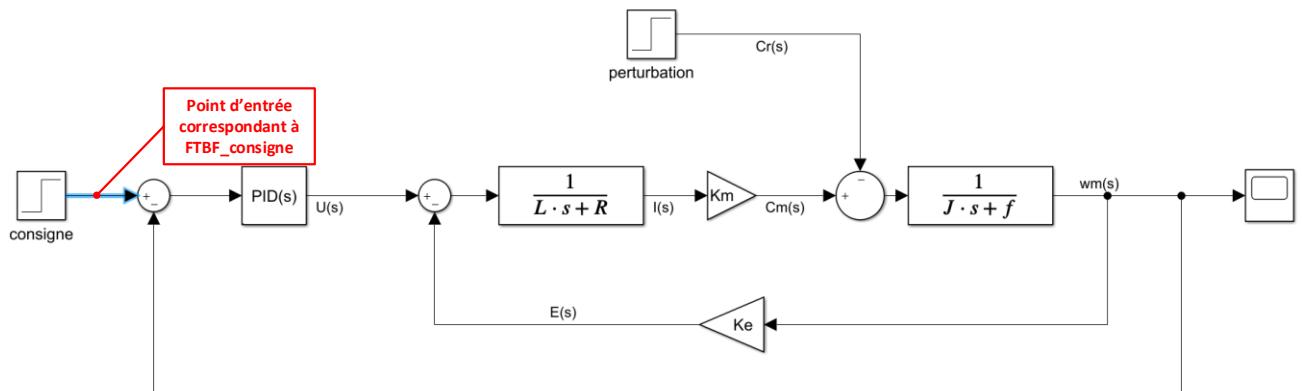


Figure 582 : choix du point d'entrée de la fonction de transfert

Cliquer alors sur **Add signal(s)** dans la fenêtre **New Step to Plot** pour valider votre choix.

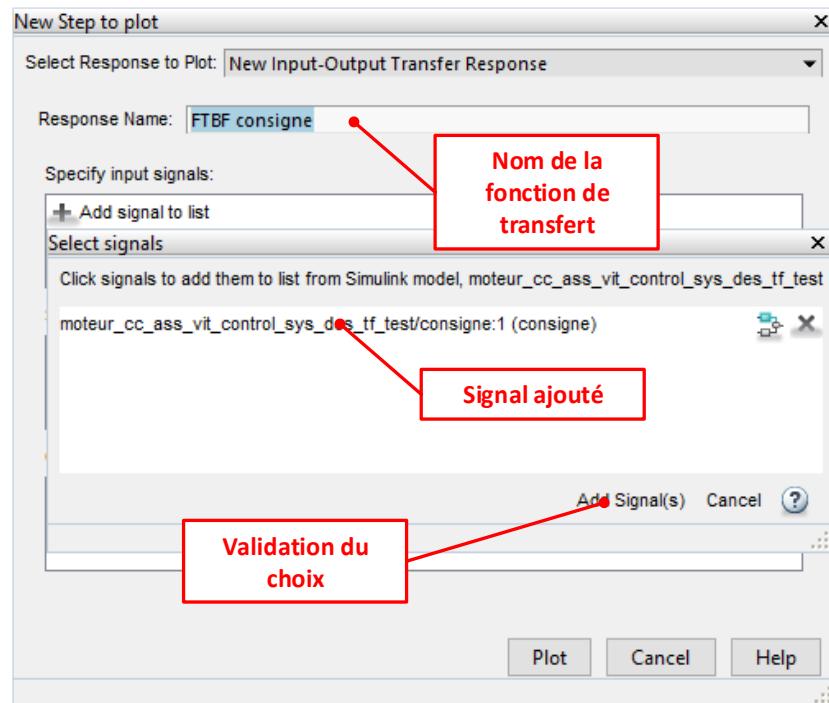


Figure 583 : fenêtre New Step to Plot

Recommencer cette opération avec le champ **Specify output Signal** pour choisir le point de sortie de la fonction de transfert et valider votre choix.

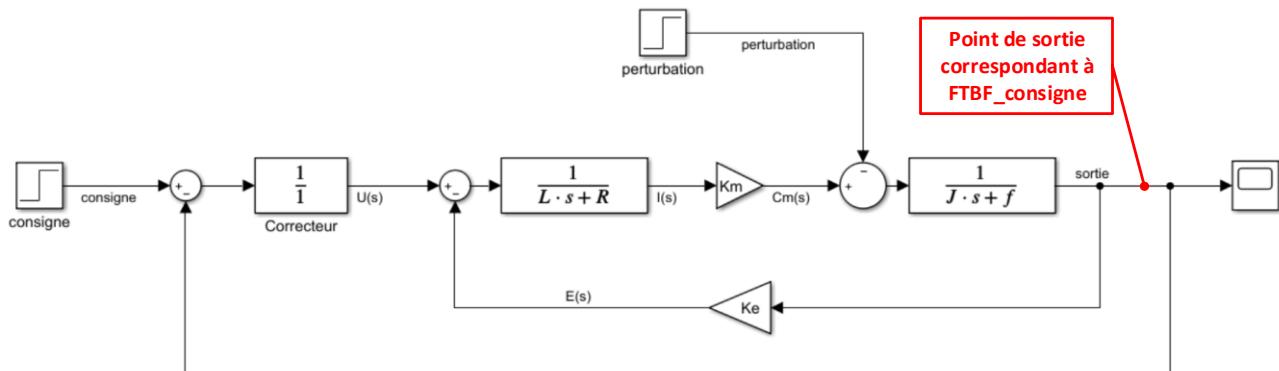


Figure 584 : choix du point de sortie de la fonction de transfert

Vous devez obtenir la configuration de la Figure 585.

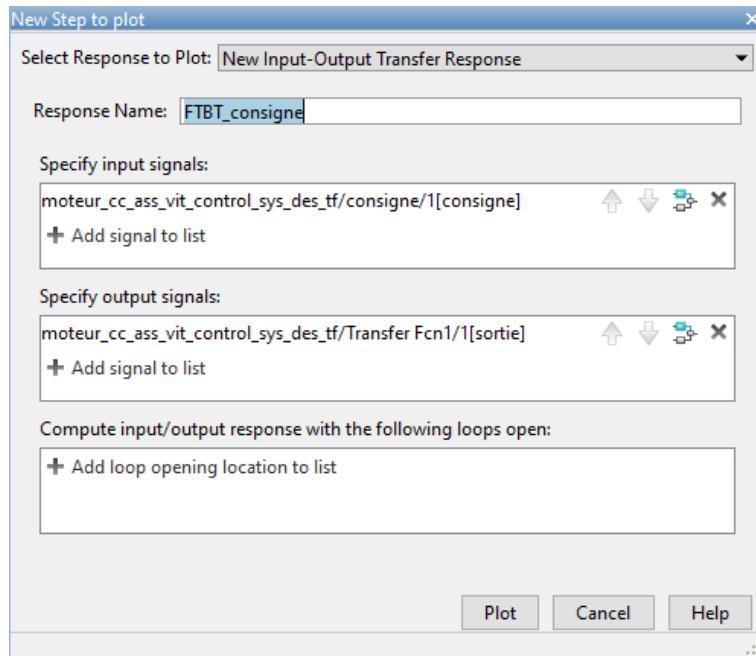


Figure 585 : définition des points de linéarisation de FTBF_consigne

Cliquer sur Plot pour obtenir le tracé de la réponse indicielle de la FTBF.

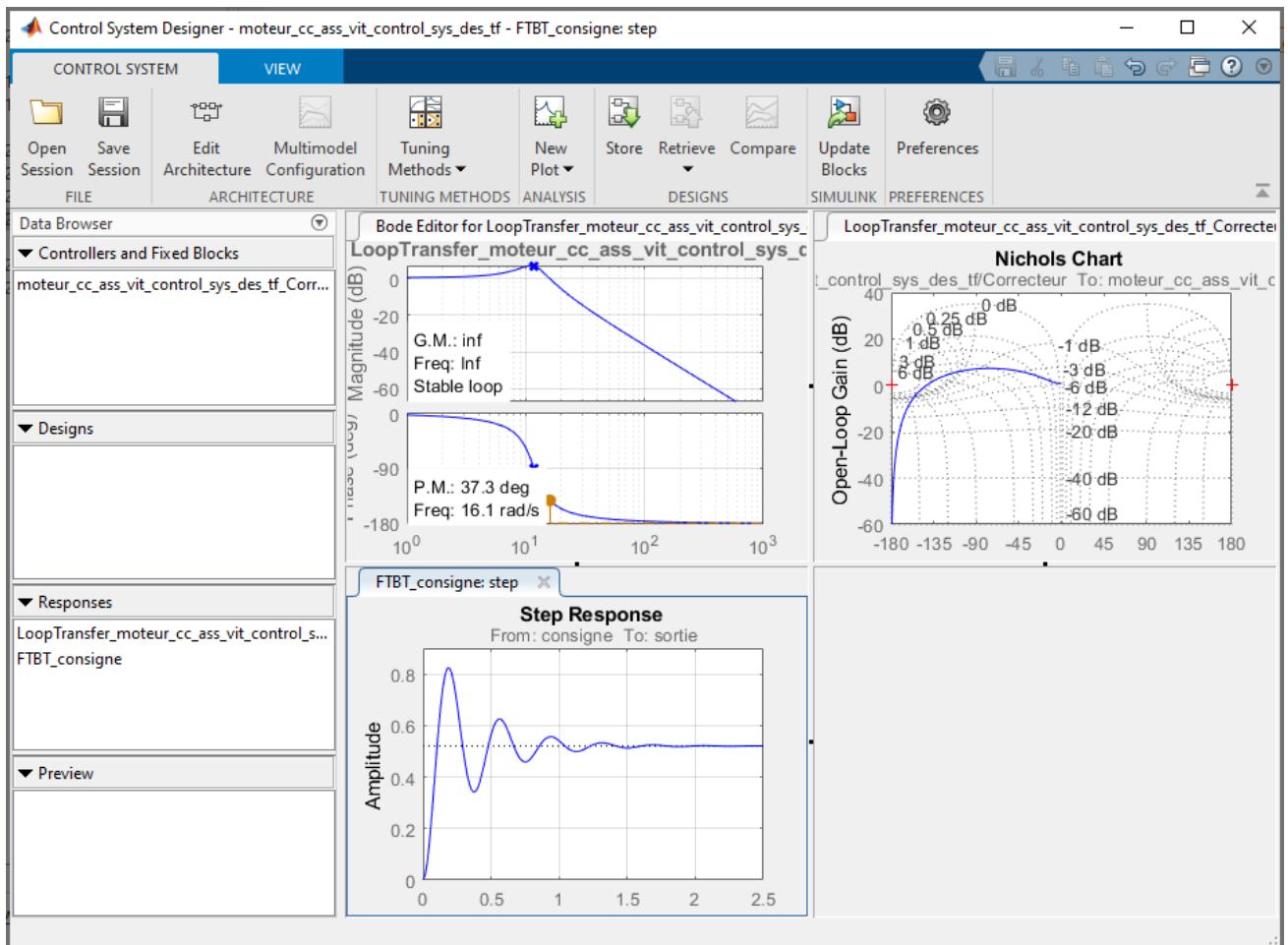


Figure 586 : ajout de la réponse indicielle de la FTBF

Nous allons maintenant tracer le diagramme de Bode de la FTBF vis-à-vis de la consigne.

Pour cela cliquer sur **New Plot/New Bode**.

Choisir la fonction de transfert FTBF_consigne (déjà définie) dans le menu déroulant de la fenêtre **New Bode to Plot** puis cliquer sur **Plot** (Figure 587).

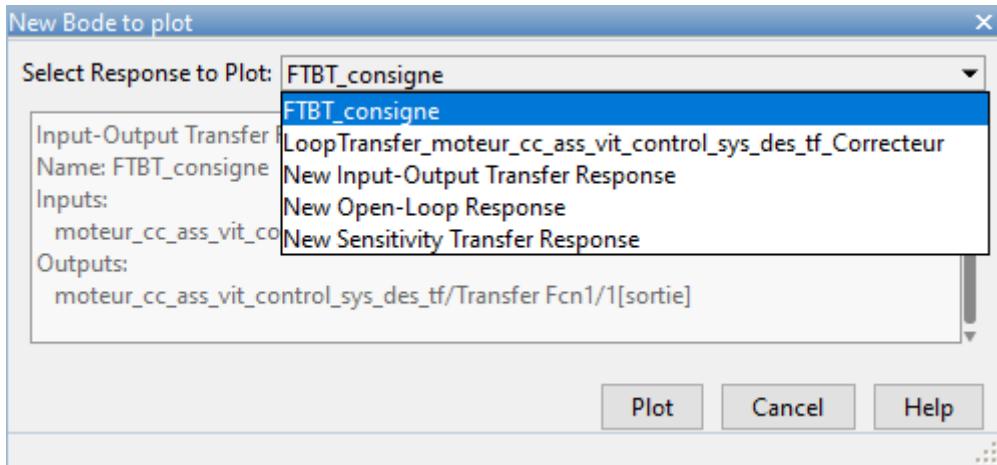


Figure 587 : tracé du diagramme de Bode de la FTBF

Vous devez obtenir la configuration de la Figure 588 qui permet de visualiser tous les diagrammes nécessaires à la conception du correcteur.

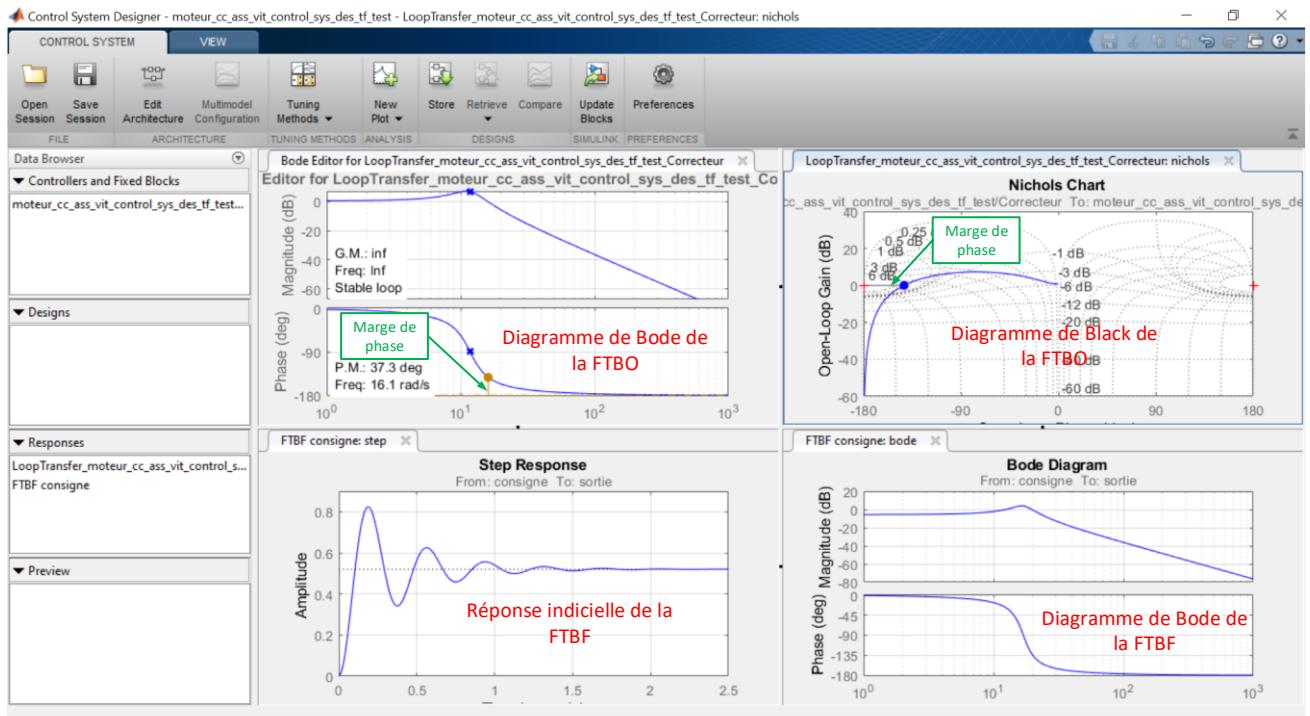


Figure 588 : configuration des diagrammes pour effectuer le réglage

3. Synthèse du correcteur

L'outil Control System Designer permet d'ajouter dans le correcteur différents éléments :

- 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**)

Pour cela plusieurs méthodes permettent d'intégrer ces fonctions de transfert au correcteur.

Cliquer dans la fenêtre graphique du diagramme de Bode de la FTBO pour faire apparaître l'onglet **Bode Editor** (Figure 589).

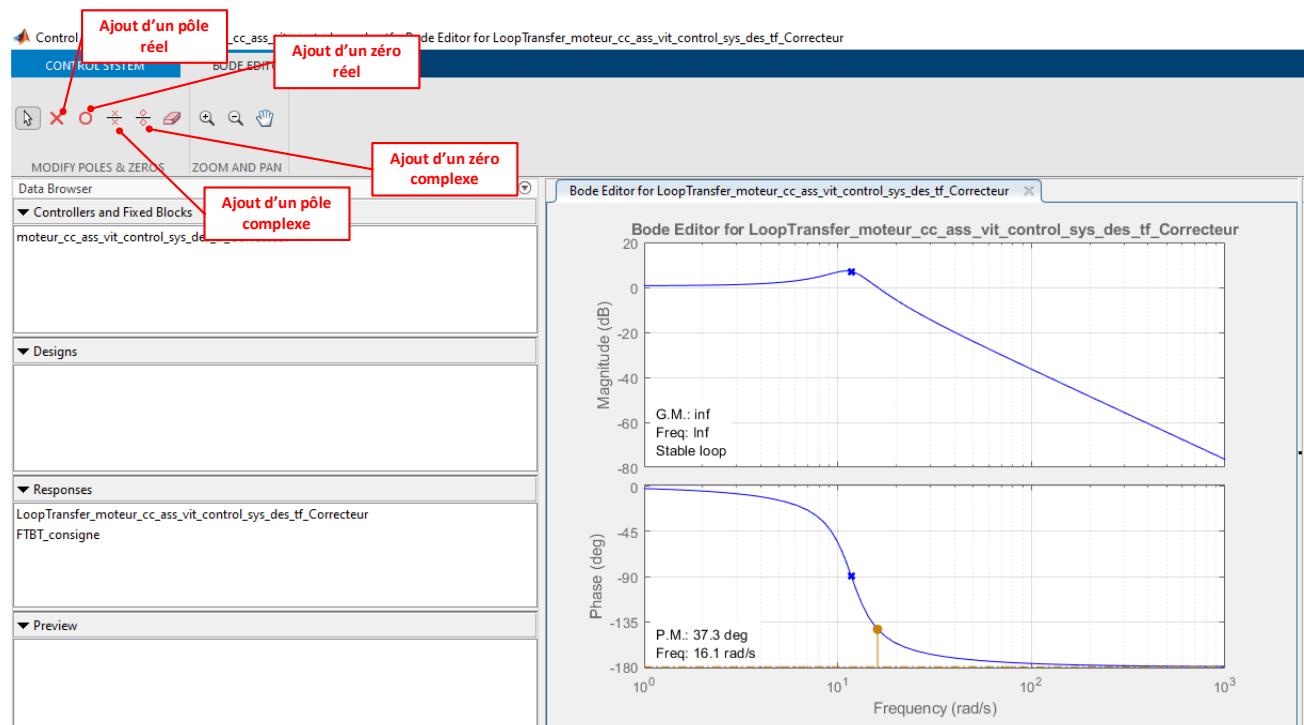


Figure 589 : la barre de commande du Bode Editor

Il est également possible d'ajouter des éléments au correcteur en cliquant avec le bouton droit dans la fenêtre graphique du diagramme de Bode de la FTBO, puis Add Pole/zero pour faire apparaître un menu déroulant qui permet de choisir les éléments à ajouter au correcteur.

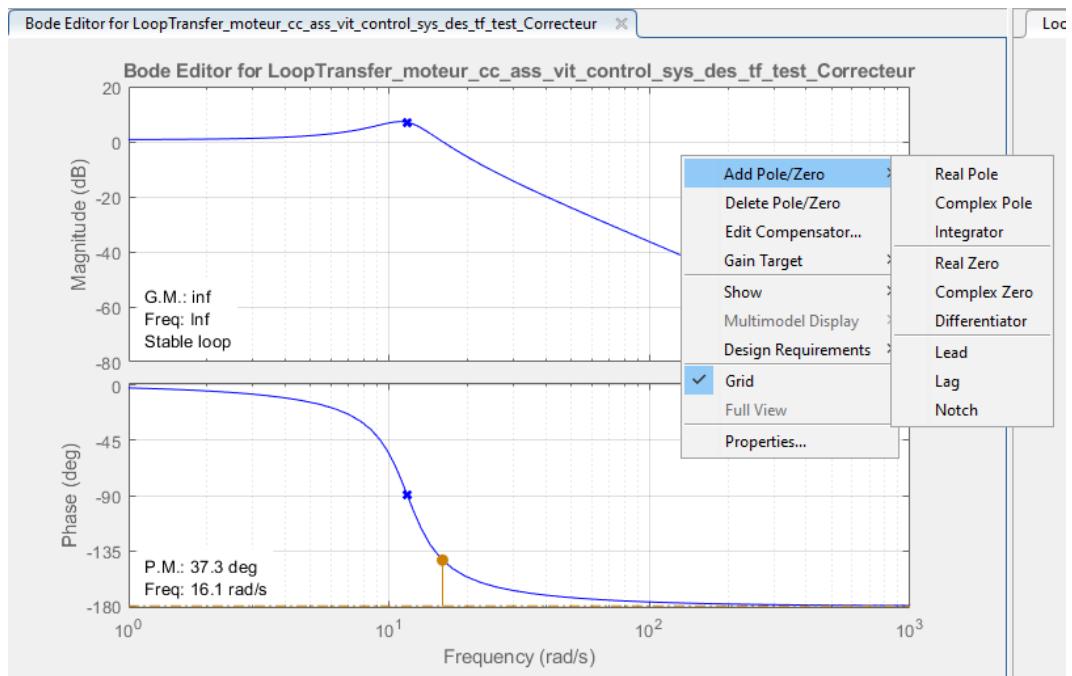


Figure 590 : ajouts d'éléments au correcteur

4. Visualisation de l'influence du gain de la FTBO

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 graphique du 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.

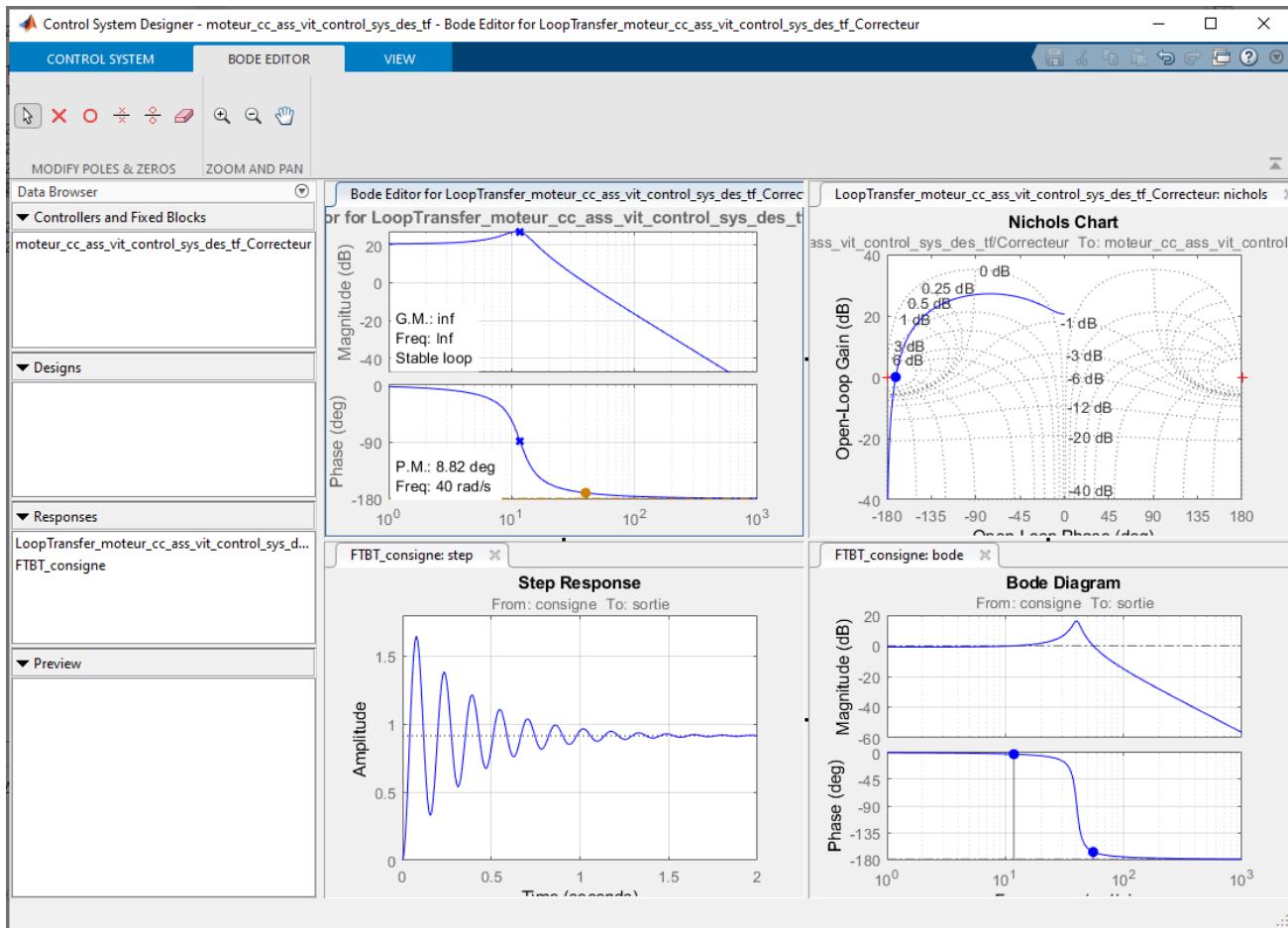


Figure 591 : variation du gain de la fonction de transfert en boucle ouverte

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. 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 de la FTBF, on observe une augmentation de la bande passante à -3dB (amélioration de la rapidité) et une résonnance plus importante.

Il est également possible de visualiser la nouvelle fonction du correcteur dans la fenêtre. Pour cela cliquer avec le bouton droit dans la fenêtre du diagramme de Bode de la FTBO et choisir **Edit Compensator** dans le menu contextuel.

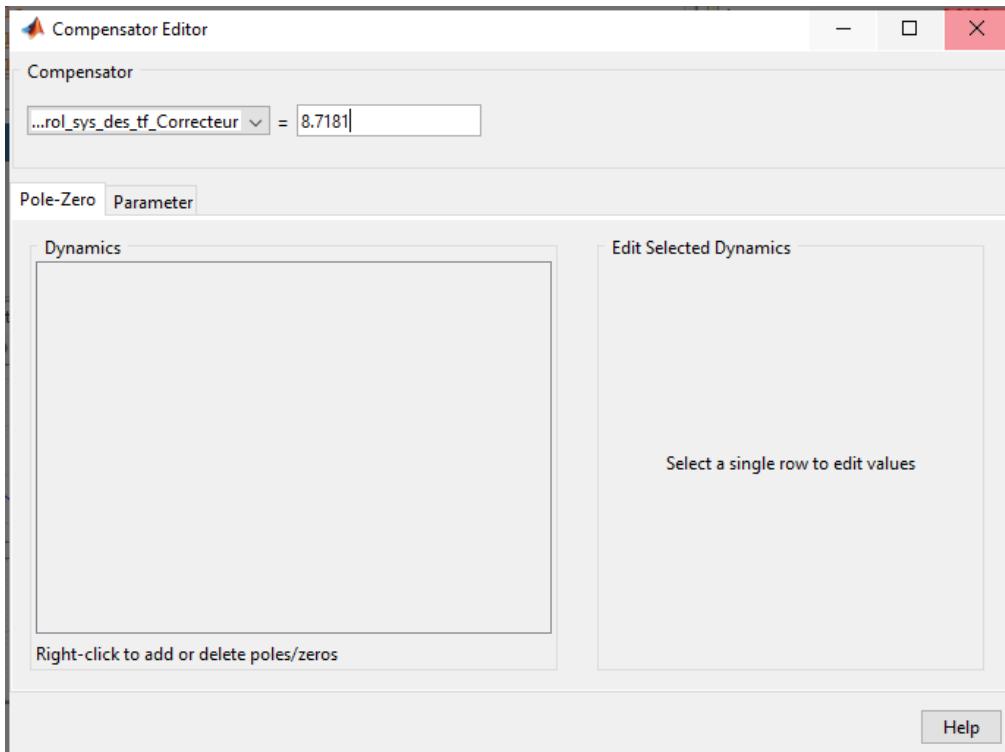


Figure 592 : Edition de la fonction de transfert su correcteur

Il apparaît que le correcteur se limite ici à un correcteur proportionnel de gain **8,7181**.

Il est également possible de modifier le gain du correcteur en utilisant d'autres procédés :

- En faisant translater verticalement le tracé de Nichols
- En indiquant directement la valeur du gain dans la fenêtre **Compensator Editor**.

Dans tous les cas la mise à jour des autres diagrammes est réalisée automatiquement.

5. 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 **graphique du diagramme de Bode de la FTBO** puis sélectionner **Add Pole zero/Integrator** (Figure 593). Un intégrateur est ajouté à notre correcteur et toutes les courbes sont mises à jour.

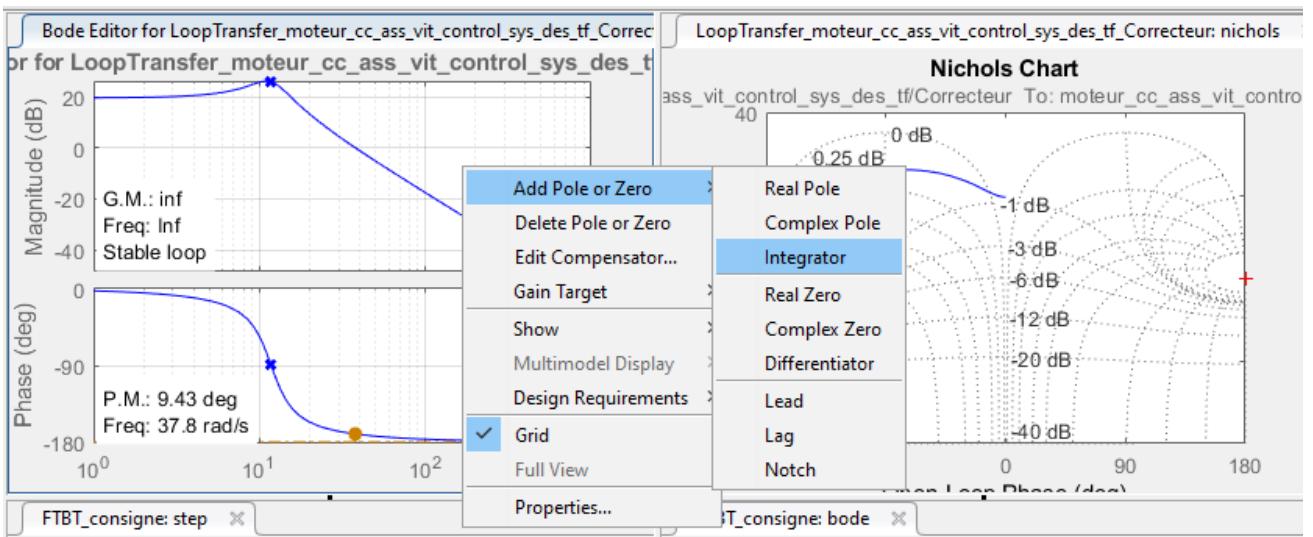


Figure 593 : ajout d'un intégrateur dans le correcteur

L'ajout d'un intégrateur à rendu l'asservissement 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 594).

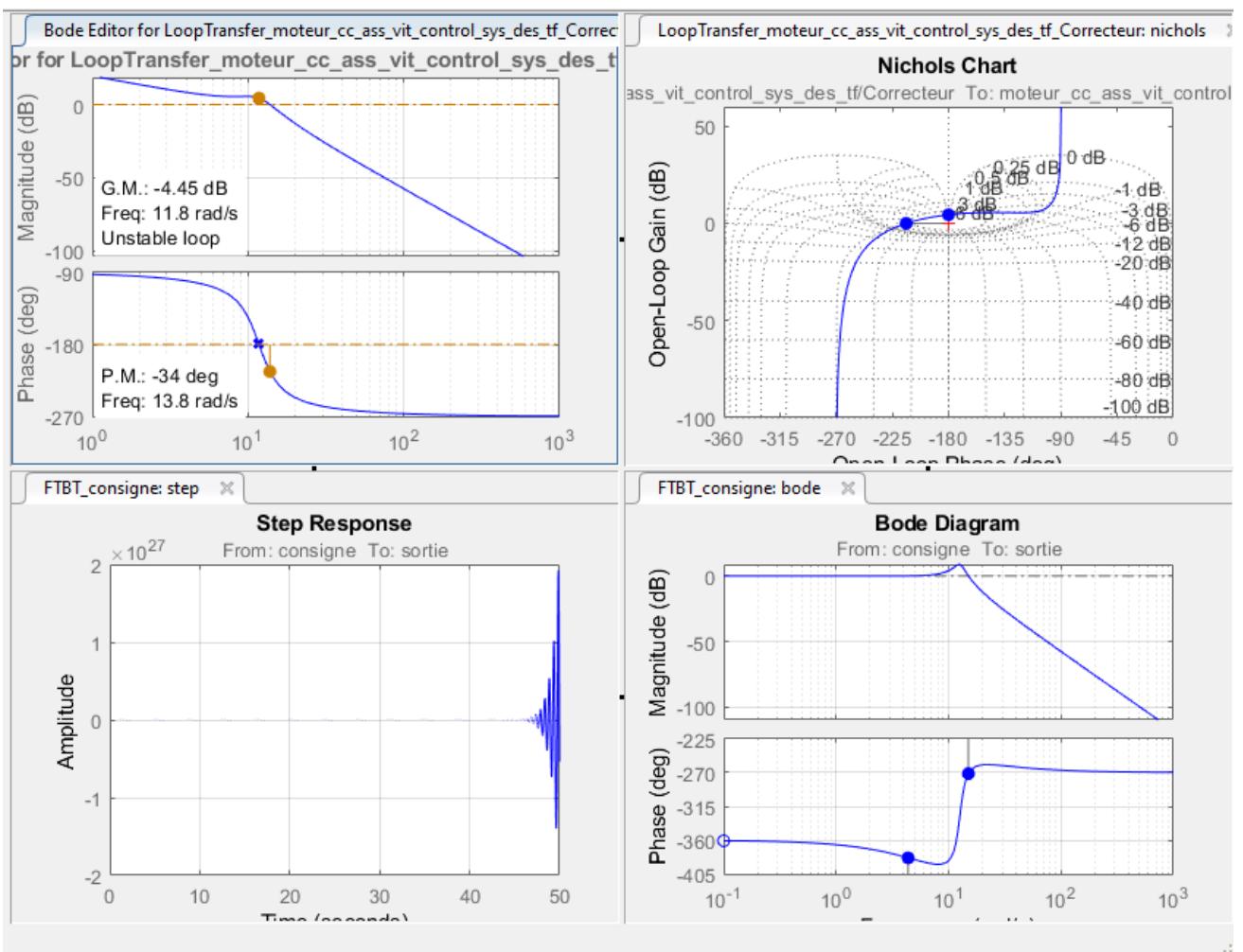


Figure 594 : visualisation de l'instabilité de la réponse indicielle

Il est possible de visualiser la nouvelle fonction de transfert du correcteur (Figure 595).

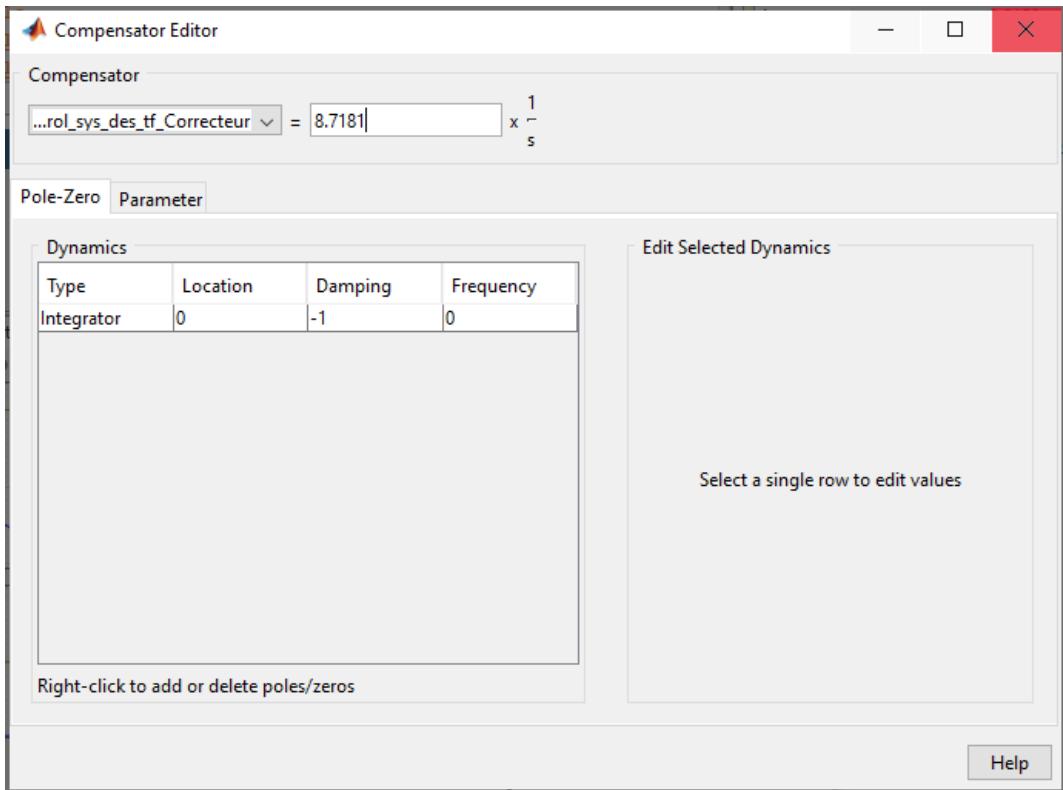


Figure 595 : visualisation de la fonction de transfert du correcteur

6. 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 596).

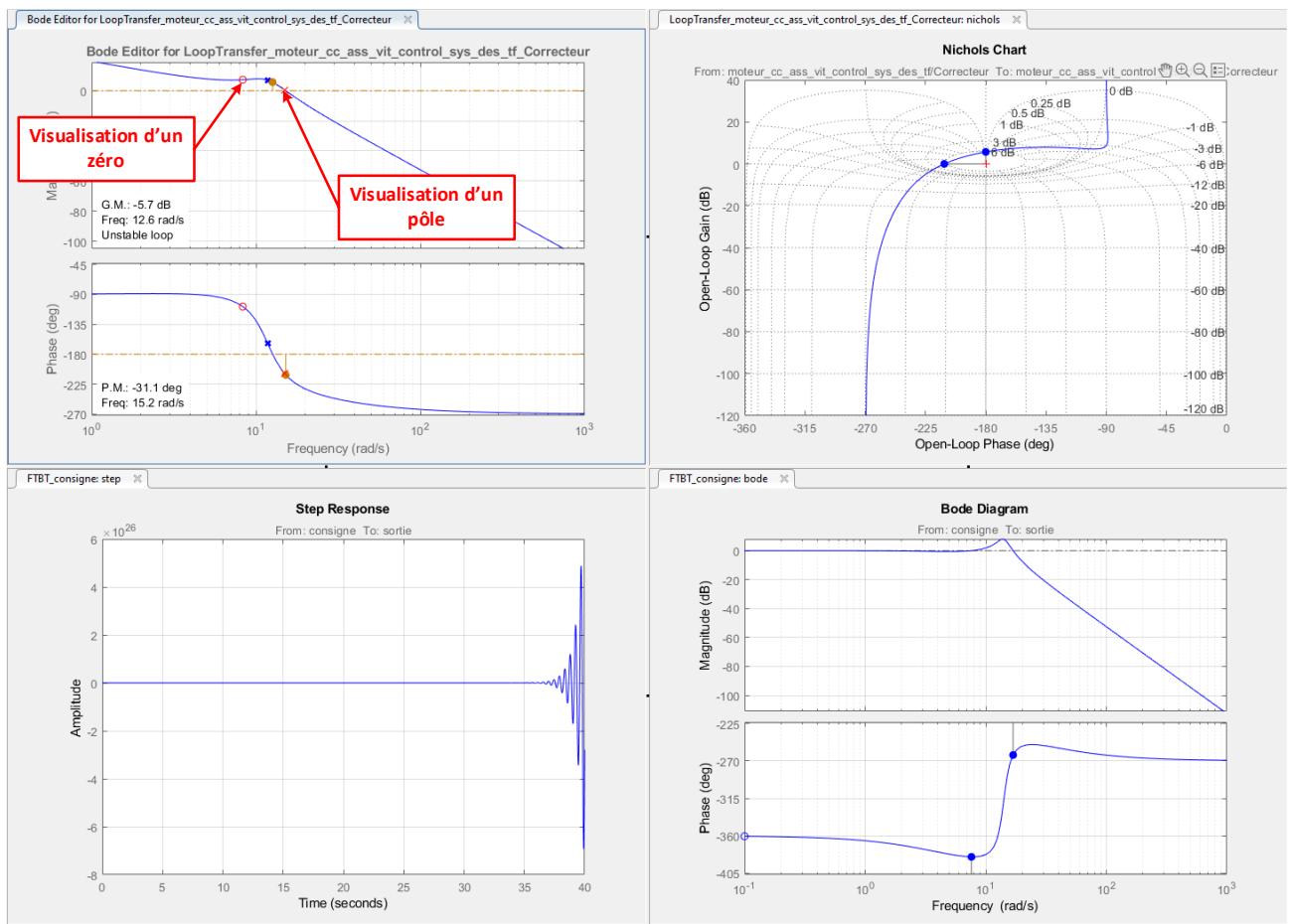


Figure 596 : visualisation des pôles et des zéros sur les courbes

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 (Figure 597).

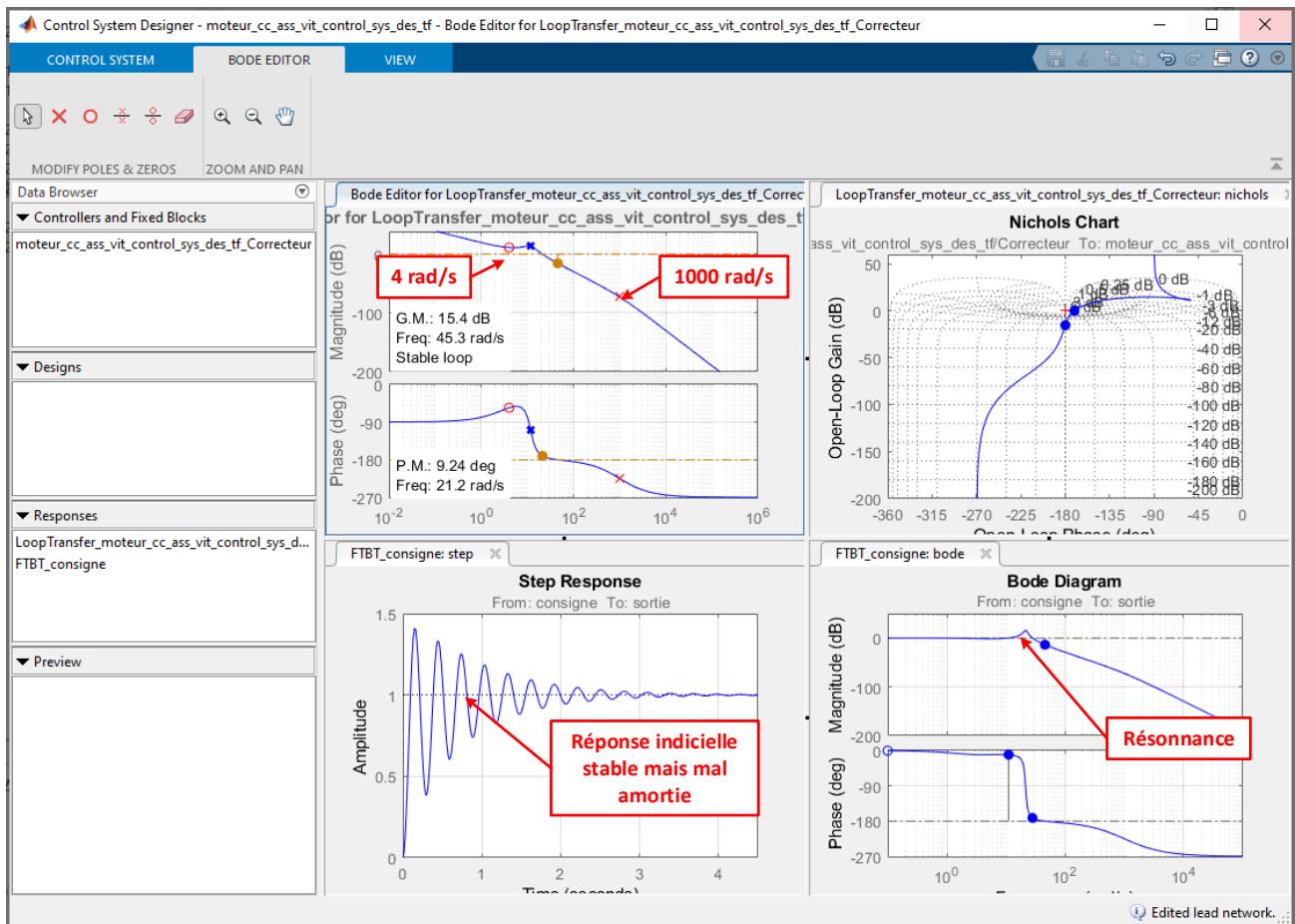


Figure 597 : stabilisation du système par ajout d'un correcteur à avance de phase

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 597).

Il est possible de visualiser la nouvelle fonction de transfert du correcteur (Figure 598).

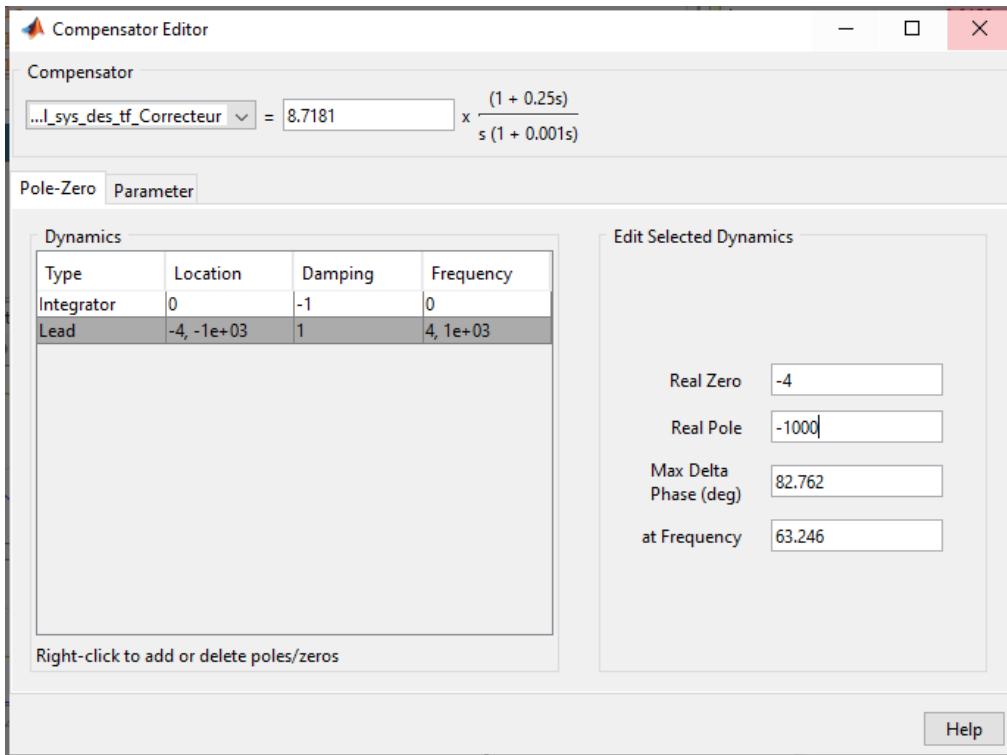


Figure 598 : visualisation de la fonction de transfert du correcteur

7. 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**.

Cliquez ensuite sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte légèrement en retrait du pic de résonance 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 599).

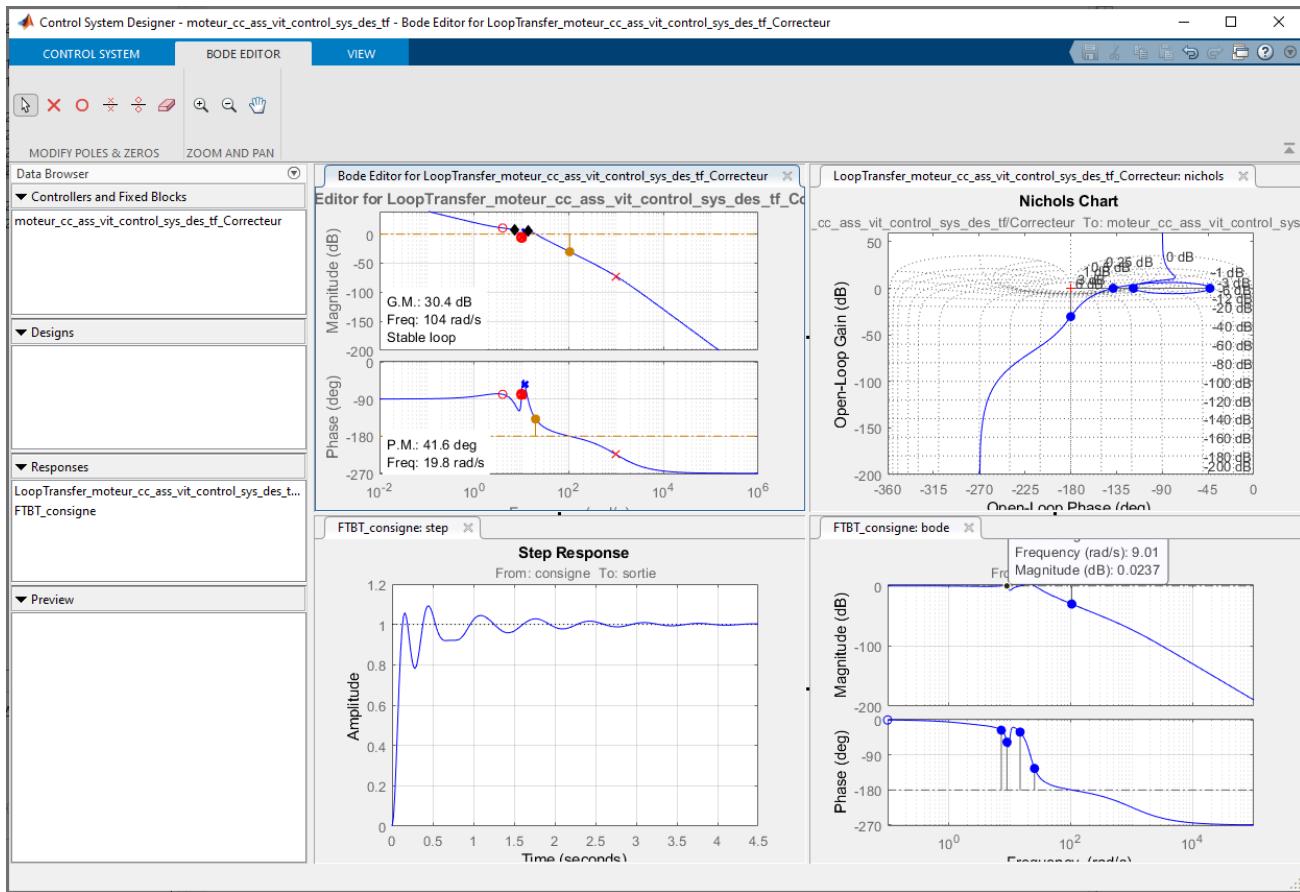


Figure 599 : ajout d'un filtre rejecteur au correcteur

Nous pouvons constater l'influence du filtre rejecteur sur la réponse indicielle et sur le diagramme de Bode de la boucle fermée. Il apparaît cependant 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 sur la réponse indicielle.

8. 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**)

Ces paramètres sont visualisables dans la fenêtre **Compensator Editor** et peuvent être modifier directement (Figure 600).

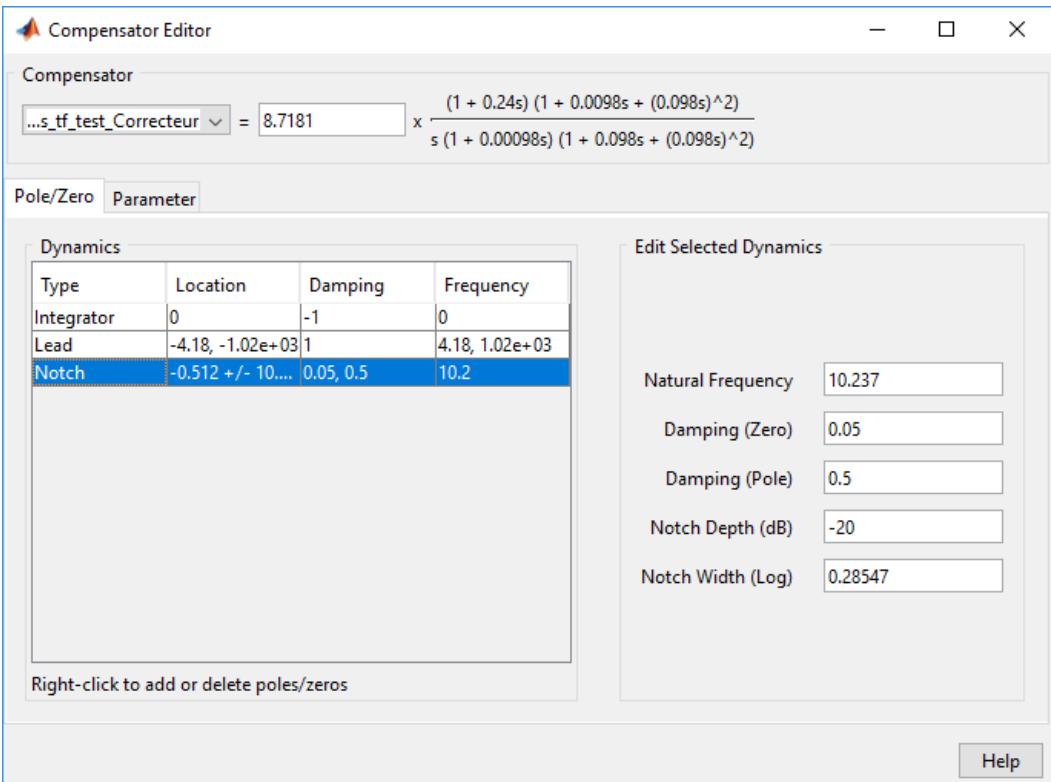


Figure 600 : visualisation des paramètres de réglage du filtre rejecteur

Cependant, ces paramètres peuvent être réglés de manière dynamique 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 601.

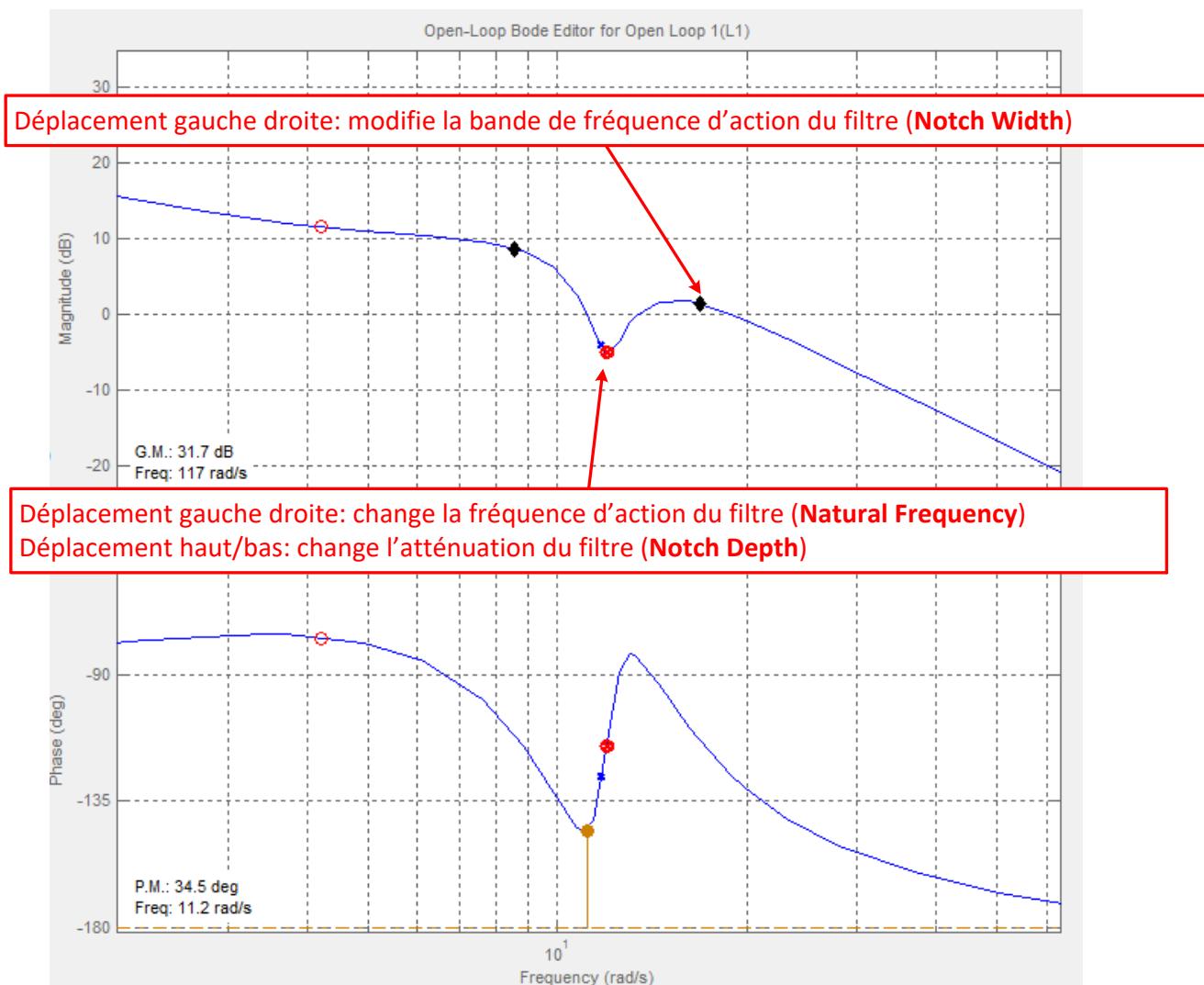


Figure 601 : réglage du filtre rejecteur

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 602).

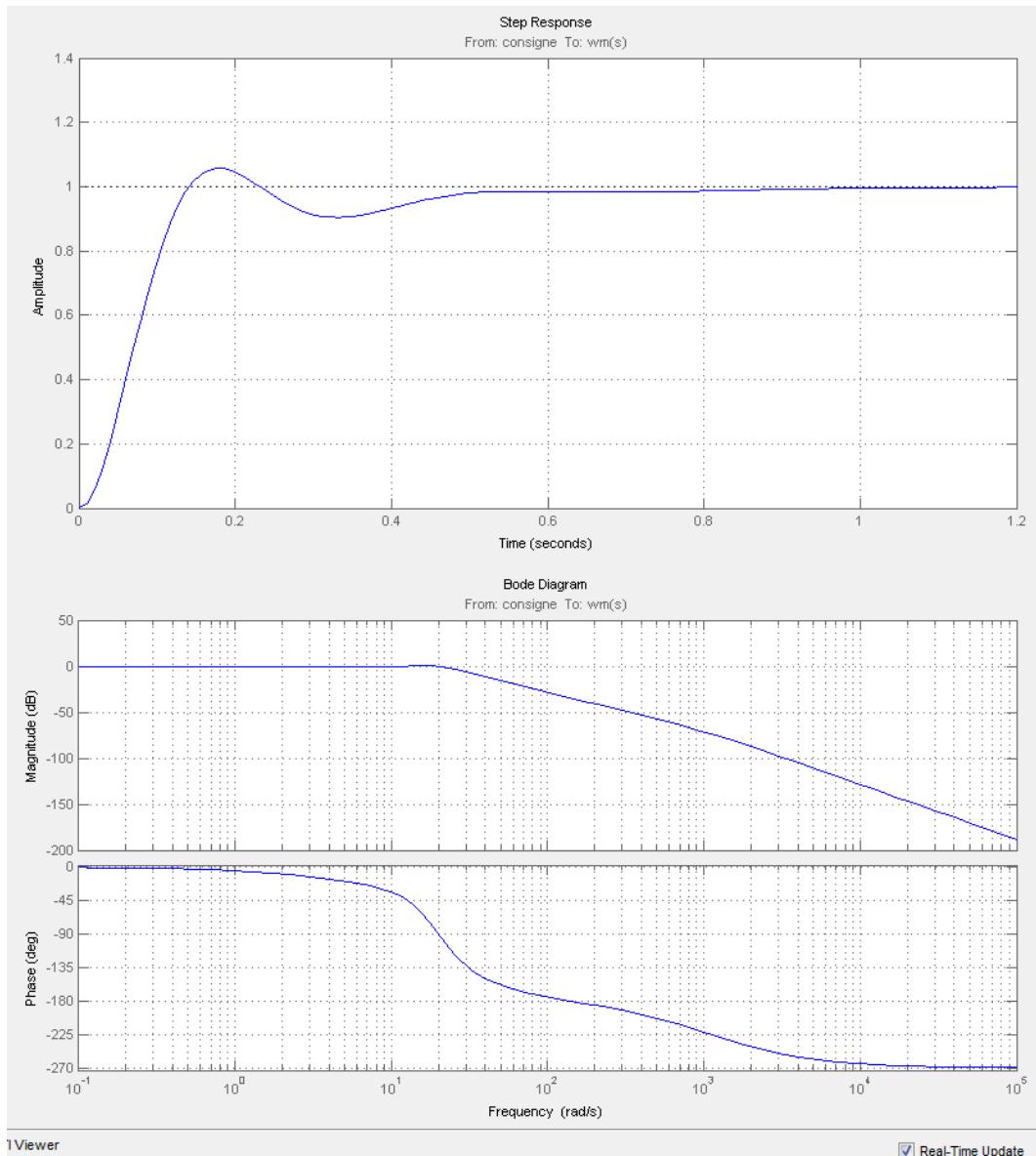


Figure 602 : performances de la boucle fermée après réglage du filtre

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 dans la fenêtre du **Compensator Editor** (Figure 603).

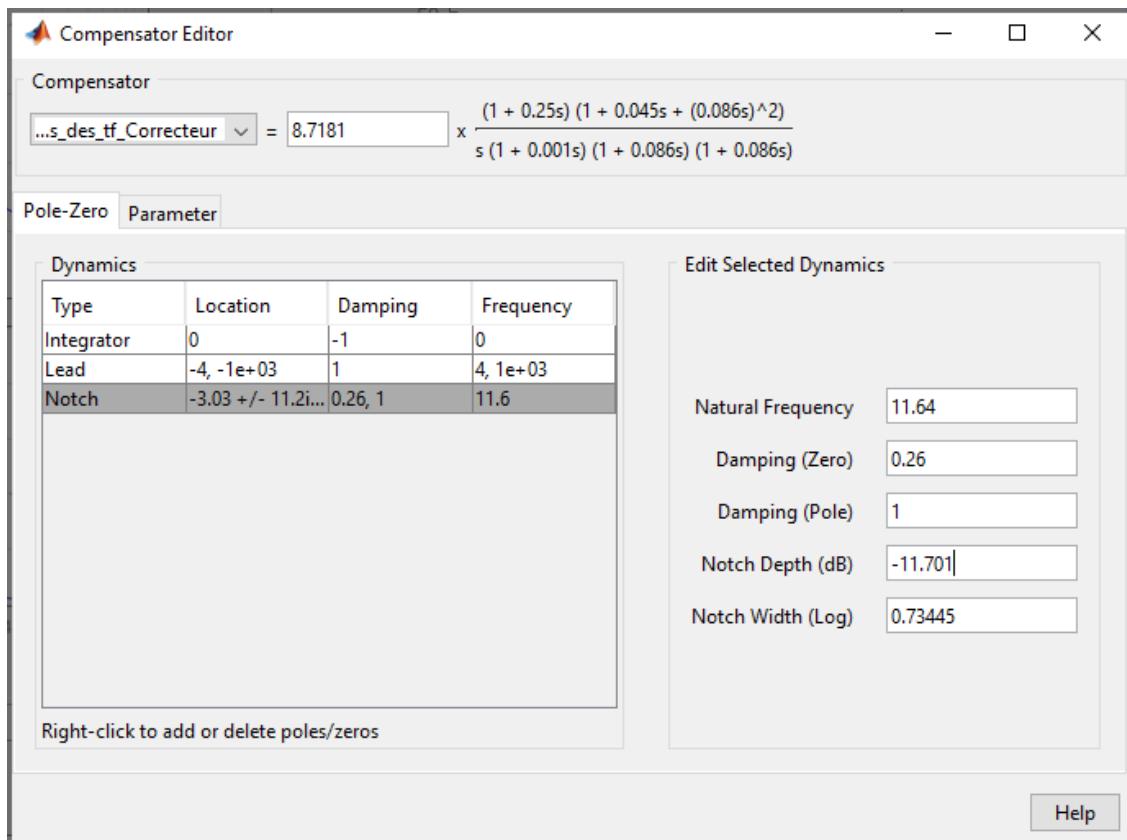


Figure 603 : visualisation de la fonction de transfert du correcteur

Il est également possible de voir l'influence du filtre rejecteur sur l'ensemble des courbes représentant le comportement de la boucle ouverte (Figure 604).

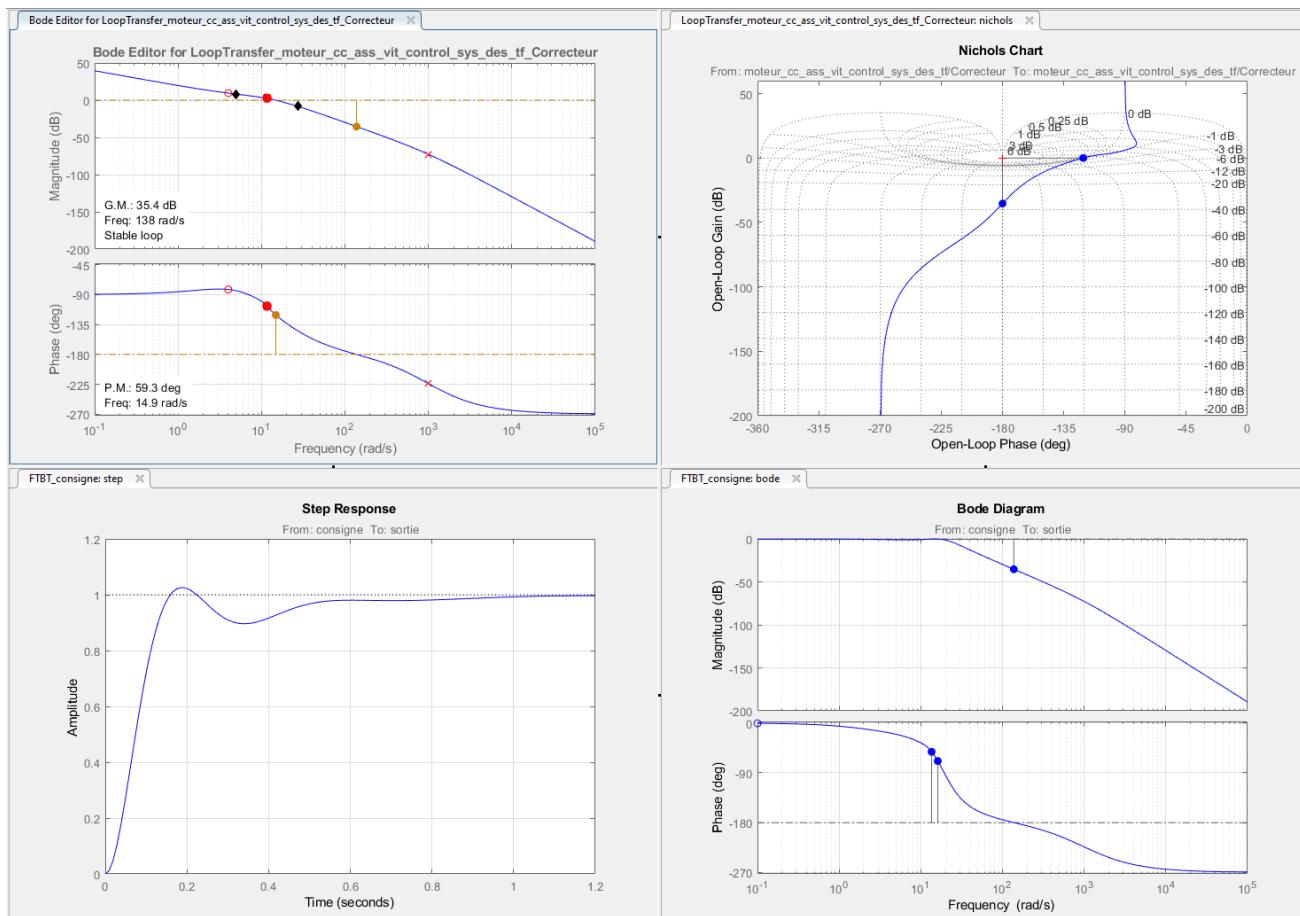


Figure 604 : influence du filtre sur le comportement de la boucle ouverte

9. Exportation de la fonction de transfert du correcteur vers le modèle Simulink

Cliquer maintenant sur Update Block dans la barre de commande du **Control System Designer** 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.

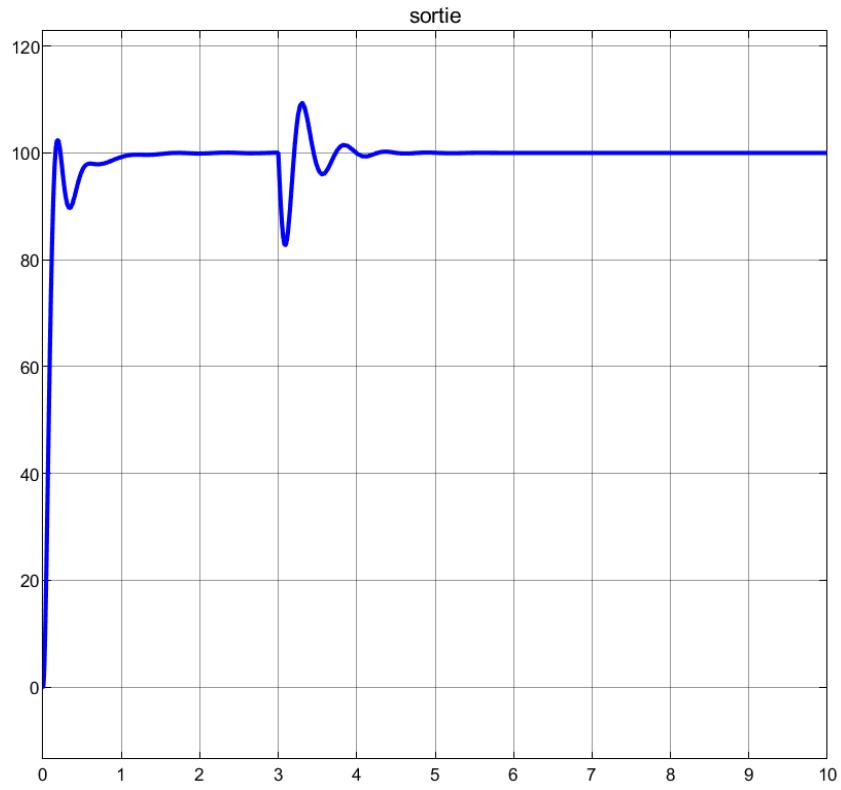


Figure 605 : réponse indicelle corrigée

La réponse indicelle corrigée (Figure 605) 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 (Figure 606).

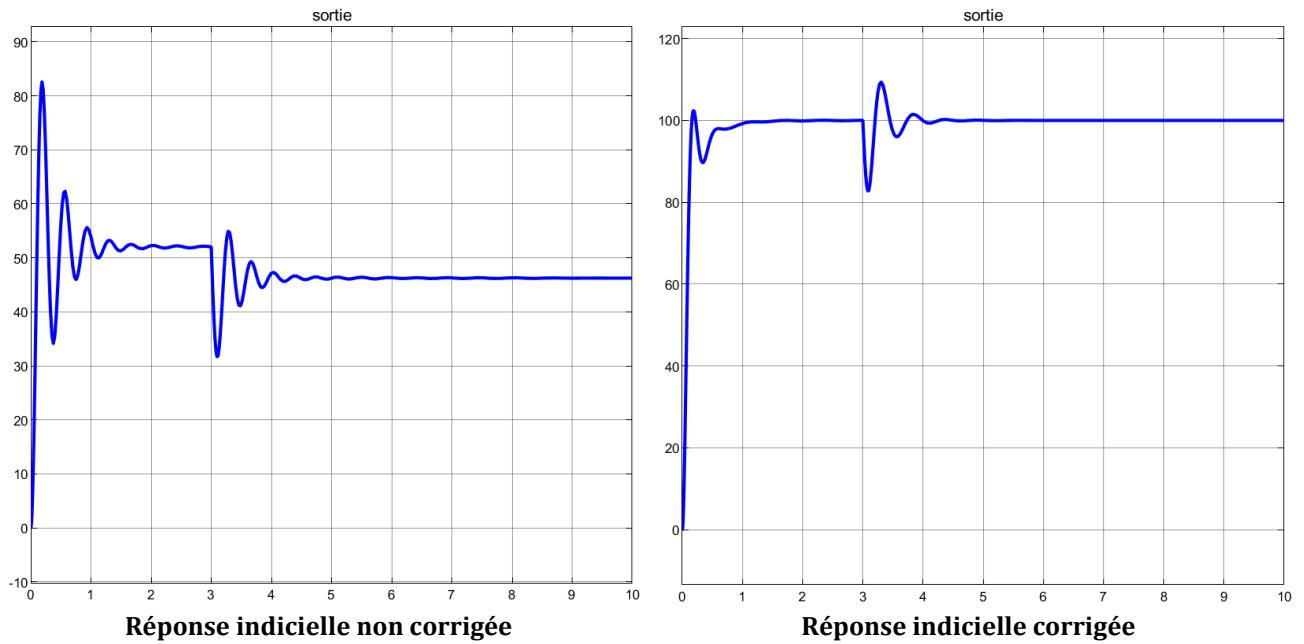


Figure 606 : comparaison des réponses indicielles corrigée et non corrigée

Chapitre 11 : Ingénierie numérique avec MATLAB

I. Introduction

Dans un processus de conception ou de validation des performances d'un système, l'ingénieur est amené à mettre en place des modèles qui lui permettent de décrire le comportement des éléments qui composent le système. La phase de modélisation va permettre d'établir des équations qui caractérisent le comportement d'un système. Sauf cas particuliers, la résolution de ces équations n'est en général pas possible en utilisant un processus de résolution analytique. La démarche de résolution est donc le plus souvent menée en utilisant un processus de calcul numérique. L'ingénieur doit être en mesure de coder avec un langage performant et adapté le processus de résolution qui correspond au problème traité. Ce chapitre présente les méthodes et les algorithmes qui sont à la base des compétences en ingénierie numérique :

- Utilisation et gestion des fonctions
- Dérivation numérique
- Résolution des équations algébriques
- Intégration numérique
- Filtrage des signaux
- Résolution des équations différentielles
- Résolution des systèmes d'équations...

La maîtrise et l'utilisation courante de ces algorithmes permettra au futur ingénieur de renforcer ses compétences et d'être en mesure de résoudre une grande partie des problèmes auxquels il sera confronté.

II. Les fonctions

A. Organisation et structure

La bonne utilisation des fonctions est un élément déterminant de la conception des algorithmes en langage MATLAB. Il faut donc commencer ce chapitre par définir les modalités de création et d'utilisation des fonctions. Il existe en langage MATLAB de très nombreuses fonctions prédéfinies qui peuvent être utilisées directement en se référant à l'aide du logiciel afin de prendre connaissance des entrées (arguments) et des sorties de la fonction.

Il est par contre indispensable de définir ses propres fonctions pour les besoins d'un algorithme. L'utilisation des fonctions permet de structurer le code et le rendre plus lisible. Ainsi un script principal pourra faire appel durant son exécution à des fonctions définies préalablement. En langage MATLAB, il existe plusieurs manières de définir et d'utiliser les fonctions.

La première méthode consiste à définir les fonctions dans des fichiers différents du script principal. Sur la Figure 607, le script **Main_script_1** fait appel aux fonctions **function_1** et **function_2** durant son exécution. Le script **Main_script_2** fait appel aux fonctions **function_3**, **function_4** et **function_2** durant son exécution. L'intérêt de coder les fonctions dans des scripts séparés permet donc à des scripts différents de faire appel à une même fonction ou d'appeler une fonction directement à partir de la fenêtre de commande. C'est très pratique et c'est en général la méthode à privilégier. Pour pouvoir être appelée, une fonction doit se trouver dans le « **path** » de MATLAB, il est donc inutile d'indiquer le chemin d'accès à la fonction. L'inconvénient est que deux fonctions différentes ne pourront pas porter le même nom.

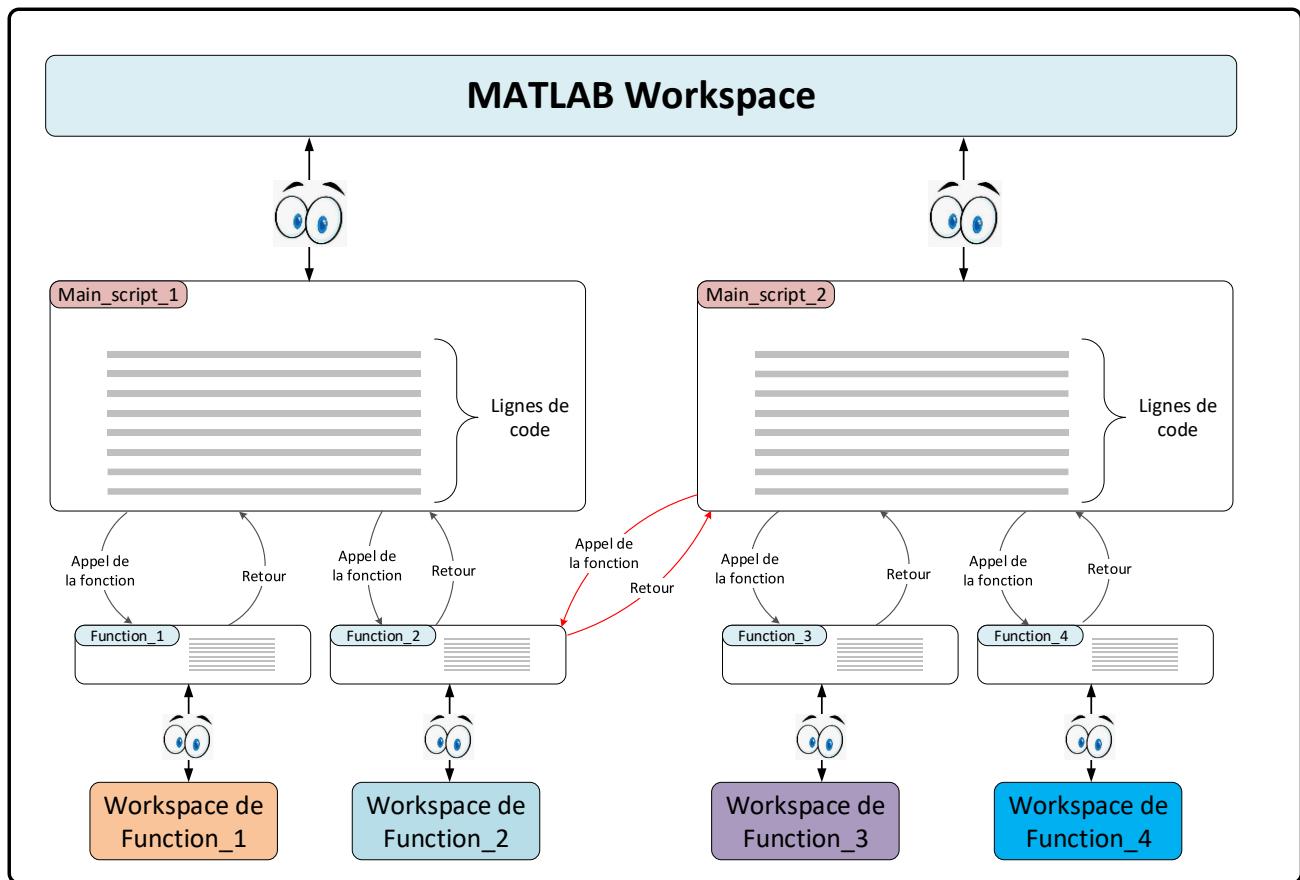


Figure 607 : organisation des scripts et des fonctions, méthode 1

La seconde méthode consiste à définir les fonctions directement dans les scripts dans lesquels elles seront utilisées. Ces fonctions sont obligatoirement placées à la fin du script.

Sur la Figure 608, le script **Main_script_1** fait appel aux fonctions **function_1** et **function_2** durant son exécution. Le script **Main_script_2** fait appel aux fonctions **function_3**, **function_4** durant son exécution.

Mainscript_2 ne pourra pas faire appel aux fonctions **function_1** ou **function_2**. Ce type de structure est adaptée à des fonctions qui n'auront d'intérêt que pour le script dans lequel elles sont définies.

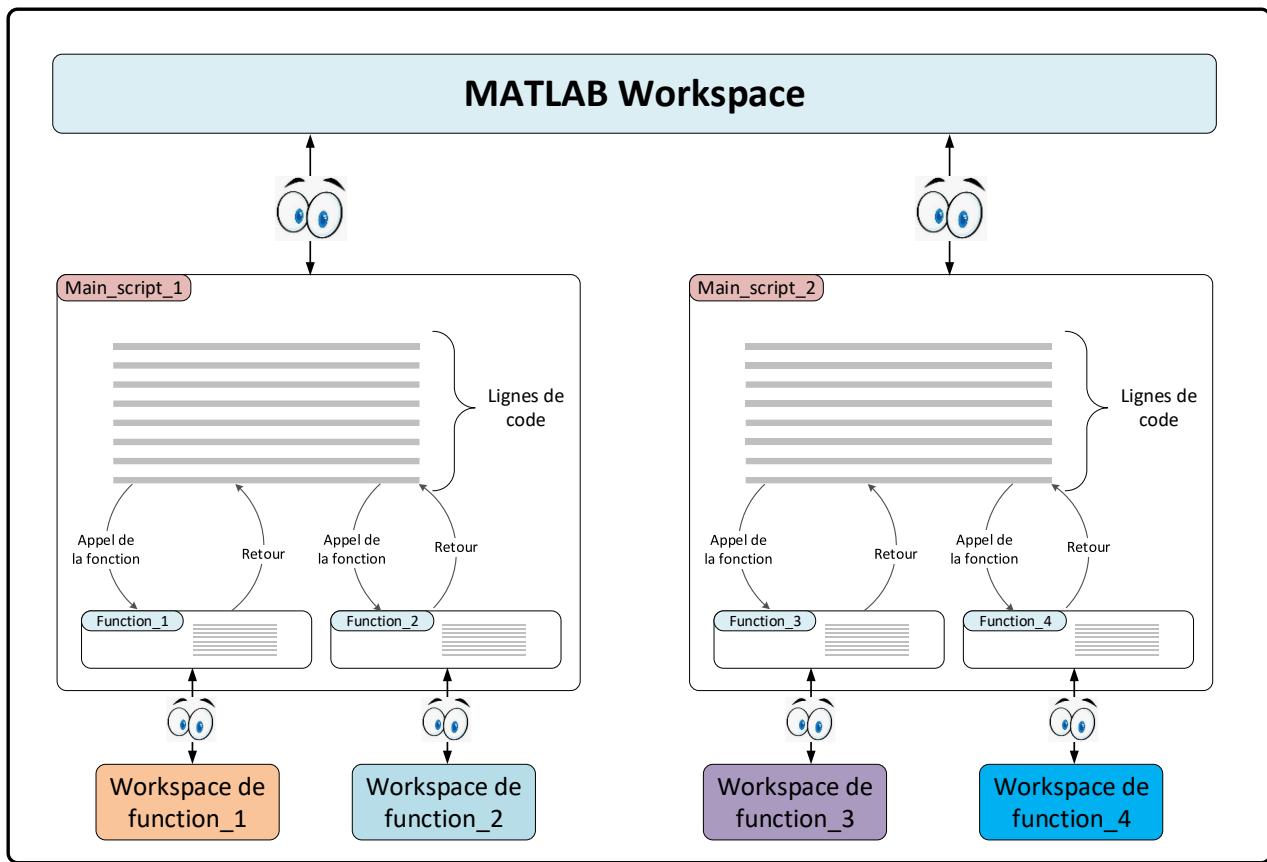


Figure 608 : organisation des scripts et des fonctions, méthode 2

B. Portée des variables

D'une manière générale, chaque fonction possède son propre **workspace** et utilise ses propres variables. Que ce soit dans le cas de la Figure 607 ou dans le cas de la Figure 608 les variables du **workspace** de MATLAB ne seront pas accessibles dans les différentes fonctions. De même les variables utilisées dans les fonctions n'apparaitront pas dans le **workspace** de MATLAB et ne pourront pas être utilisées dans le script. Si des variables doivent être partagées entre le script et les fonctions, des commandes spécifiques permettent de le préciser, mais cela n'aura pas lieu par défaut.

C. Crédation et définition d'une fonction

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Fonctions**.

Prenons l'exemple d'une fonction élémentaire **my_0_function** (Figure 609) permettant de réaliser un calcul simple en prenant un nombre **x** comme argument et renvoyant la valeur calculée de **y** en fonction de **x**.

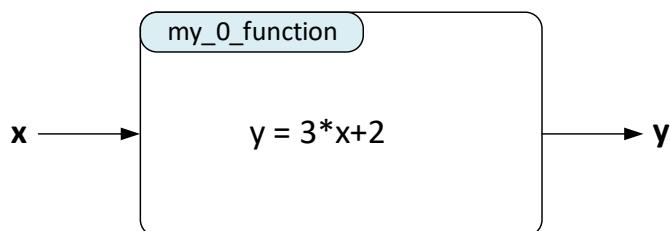


Figure 609 : my_0_function

La première méthode consiste à créer la fonction dans un fichier dédié à la fonction. Ce fichier doit respecter les règles suivantes (Figure 610) :

- commencer par le mot clé **function**
- préciser le nom de la fonction : **my_0_function**
- définir l'ensemble des paramètres d'entrée (arguments) : **x**
- définir l'ensemble des paramètres de sortie : **y**
- Le nom du fichier doit correspondre au nom de la fonction : **my_0_function.m**
- Se terminer par le mot réservé **end**

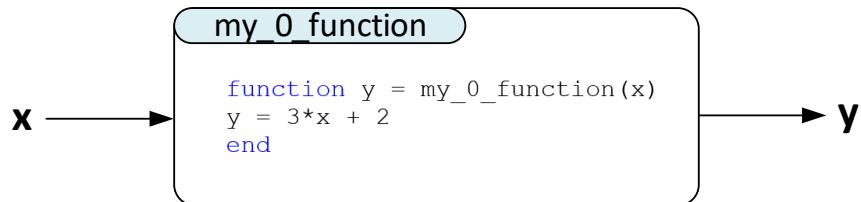


Figure 610 : codage de la fonction my_0_function

Créer le dossier « Mes_fonctions » que vous placerez dans le **path** de MATLAB. Choisir ce dossier comme dossier courant dans MATLAB.

Ouvrir un nouveau script et le sauvegarder sous le nom « **my_0_function.m** ».

Taper les lignes de code suivante :

```
function y = my_0_function(x)
y = 3*x + 2
end
```

Sauvegarder votre saisie.

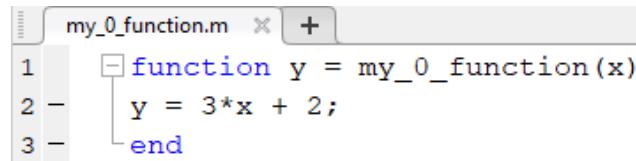


Figure 611 : codage de la fonction my_0_function

La fonction **my_0_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.

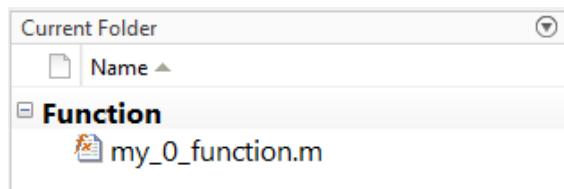


Figure 612 : visualisation de la fonction my_0_function dans le répertoire courant

Dans la fenêtre de commande, faire un appel à la fonction en spécifiant une valeur pour l'argument d'entrée.

```
>> my_0_function(5)
```

```
ans =  
17
```

Il est également possible de stocker le contenu du résultat dans une variable :

```
>> a = my_0_function(5)  
  
a =  
17
```

Cette fonction peut maintenant être utilisée depuis la fenêtre de commande ou être appelée par un script durant son exécution.

Une fonction peut posséder plusieurs entrées et plusieurs sorties. C'est le cas de la fonction **my_1_function** de la Figure 613.

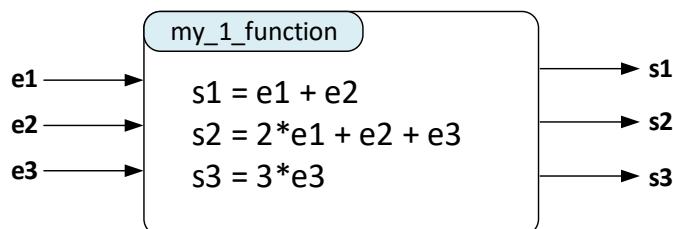


Figure 613 : my_1_function

La fonction sera codée comme indiqué sur la Figure 614.

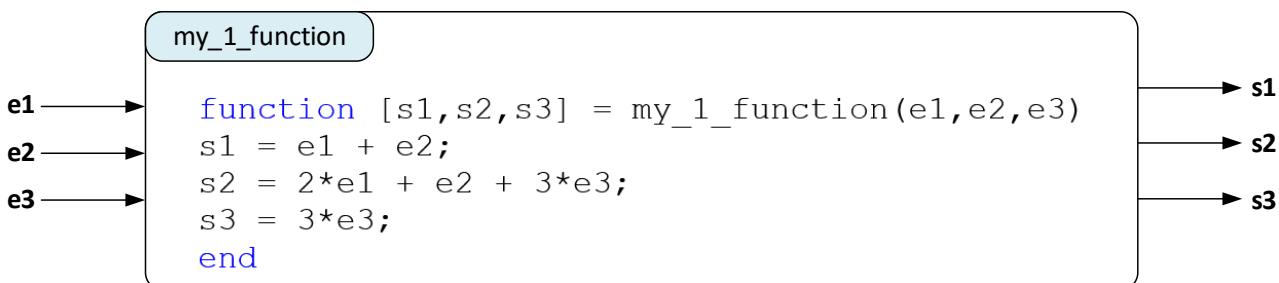


Figure 614 : codage de my_1_function

Ouvrir un nouveau script et le sauvegarder dans le dossier **Mes_fonctions** sous le nom « **my_1_function.m** ».

Taper les lignes de code suivante :

```
function [s1,s2,s3] = my_1_function(e1,e2,e3)  
s1 = e1 + e2;  
s2 = 2*e1 + e2 + 3*e3;  
s3 = 3*e3;  
end
```

Sauvegarder votre saisie.

```

my_0_function.m
my_1_function.m
+ 

1 function [s1,s2,s3] = my_1_function(e1,e2,e3)
2 
3 s1 = e1 + e2;
4 s2 = 2*e1 + e2 + 3*e3;
5 s3 = 3*e3;
6 end

```

Figure 615 : codage de la fonction my_1_function

La fonction **my_1_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.

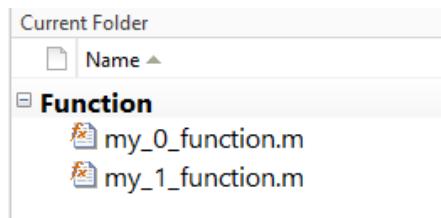


Figure 616 : visualisation de la fonction my_1_function dans le répertoire courant

Dans la fenêtre de commande, faire un appel à la fonction **my_1_function** en spécifiant une valeur pour chaque argument d'entrée.

```

>> my_1_function(1,2,3)

ans =
    3

```

Nous constatons que par défaut la fonction ne renvoie pas toutes ses sorties mais uniquement la première sortie **s1**. Il est possible de lui demander la sortie que l'on souhaite en tapant la commande suivante :

```

>> [~,~,a] = my_1_function(1,2,3)
a =
    9

```

Enfin, il est également possible de stocker le contenu des trois sorties dans des variables :

```

>> [a,b,c] = my_1_function(1,2,3)
a =
    3

b =
    13

c =
    9

```

Cette fonction peut maintenant être utilisée depuis la fenêtre de commande ou être appelée par un script durant son exécution.

D. Utilisation d'une fonction par un script

Nous allons maintenant examiner comment un script fait appel à une fonction et utilise le résultat de cette fonction. L'objectif est de créer un script permettant d'obtenir la représentation graphique d'une fonction avec la variation d'un paramètre. La fonction sera créée dans un fichier séparé **my_2_function.m**.

La fonction à tracer est définie par $y(x) = \sin(x) \cdot e^{-\frac{x}{a}}$

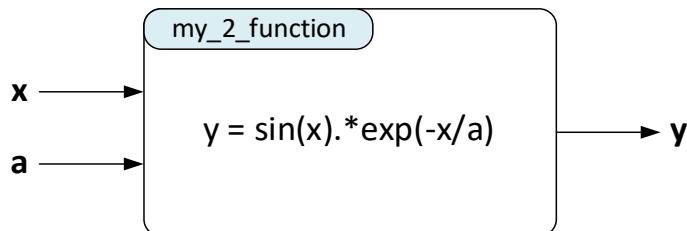


Figure 617 : my_2_function

La fonction sera codée comme indiqué sur la Figure 618.

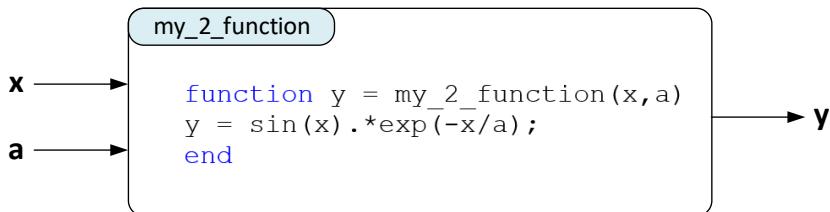


Figure 618 : codage de my_2_function

Ouvrir un nouveau script et le sauvegarder dans le dossier **Mes_fonctions** sous le nom « **my_2_function.m** ».

Taper les lignes de code suivante :

```
function y = my_2_function(x,a)
y = sin(x).*exp(-x/a);
end
```

Sauvegarder votre saisie.

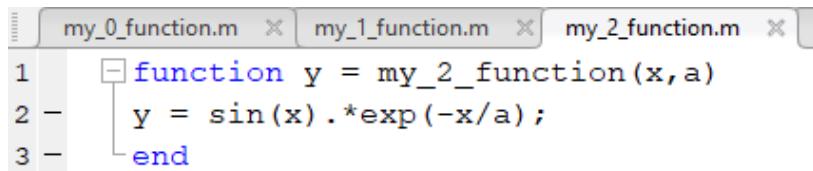


Figure 619 : codage de la fonction my_2_function

La fonction **my_2_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.



Figure 620 : visualisation de la fonction my_2_function dans le répertoire courant

Créer un nouveau script et saisir les lignes de commandes suivantes ou **ouvrir** directement le fichier « **script_0.m** ».

```
%% script_0
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction my_2_function.m

% création d'une nouvelle fenêtre graphique
figure;
hold on;

% création du vecteur des temps
t = [0:0.001:60];

% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my_2_function(t,k), 'LineWidth',2); % appel de la fonction
end

% affichage de la grille
grid on;
grid minor;

% titre et étiquettes des axes
title('sin(t)*exp(-t/a)', 'Color', 'red');
xlabel('Temps', 'Color', 'blue');
ylabel('Valeur du sinus', 'Color', 'blue');
```

Figure 621 : script_0.m

Ici lors de l'appel à la fonction **plot**, la fonction qui sera tracée sera la fonction contenue dans **my_2_function**. Le premier argument de **my_2_function** sera le temps **t** et le second argument sera le paramètre **k** qui va varier pour chacune des courbes.

Sauvegarder et exécuter le script.

Le graphique obtenu est visualisable sur la Figure 622.

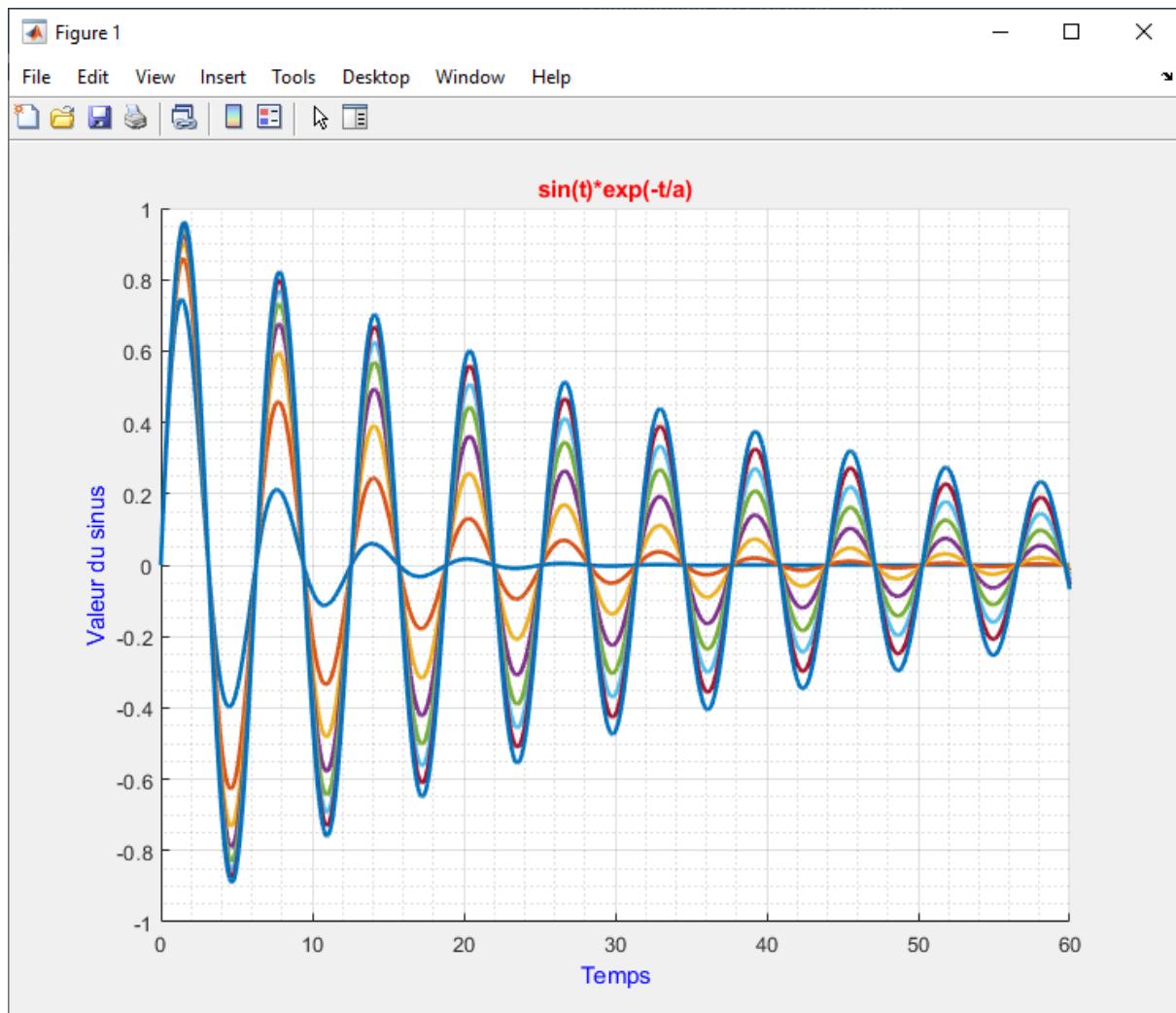


Figure 622 : résultat obtenu suite à l'exécution de script_0.m

Les différentes courbes apparaissent, mais il est impossible de relier une courbe particulière à la valeur du paramètre **a** correspondant, ce qui rend cette série de courbes inexploitables. Nous allons apporter quelques améliorations à ce script en lui permettant de générer automatiquement une légende afin de relier les courbes aux différentes valeurs du paramètre **a**. Nous utiliserons également l'interpréteur Tex afin d'écrire l'équation de manière plus lisible.

Pour créer la légende nous utiliserons un tableau de type **cell** permettant de stocker tout type de données et en particulier des chaînes de caractères. À chaque itération, une ligne de la légende est générée puis stockée dans la cellule sous la forme d'une chaîne de caractères. La légende est ensuite ajoutée au graphique à l'aide de la commande **legend** qui permet d'afficher directement le contenu de la cellule.

Pour utiliser l'interpréteur Tex, nous ferons appel à la commande **texlabel**.

Modifier le script conformément à la Figure 623 ou **ouvrir** le fichier « **script_0_legende.m** ».

```
%% script_0_legend
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
```

```

% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% La légende est générée automatiquement
% L'équation de la courbe est stockée dans la fonction my_2_function.m

% création d'une nouvelle fenêtre graphique
figure;
hold on;

% création du vecteur des temps
t = [0:0.001:60];

% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my_2_function(t,k), 'LineWidth',2); % appel de la fonction
end

% affichage de la grille
grid on;
grid minor;

% titre et étiquettes des axes
title('sin(t)*exp(-t/a)', 'Color', 'red');
xlabel('Temps', 'Color', 'blue');
ylabel('Valeur du sinus', 'Color', 'blue');

% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction my_2_function.m

% création d'une nouvelle fenêtre graphique
figure;
hold on;

% création du vecteur des temps
t = [0:0.001:60];

% création d'une cellule pour stocker les lignes de la légende
Titre_Legende = cell(1,1);
ind_Legende = 1;

% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my_2_function(t,k), 'LineWidth',2);
Titre_Legende{1,ind_Legende} = 'a=' + string(k); % création de la ligne de la légende
ind_Legende = ind_Legende + 1;
end

% affichage de la légende
legend(Titre_Legende);

% affichage de la grille
grid on;
grid minor;

% titre et étiquettes des axes
titre = texlabel('sin(t).e^((-t/a)) en fonction de t');
title(titre, 'Color', 'red');
xlabel('Temps t en (s)', 'Color', 'blue');
ylabel('Valeur du sinus amorti', 'Color', 'blue');

```

Figure 623 : ajout d'une légende automatiquement au graphique

Sauvegarder, puis **exécuter** le script pour obtenir la représentation graphique de la Figure 624.

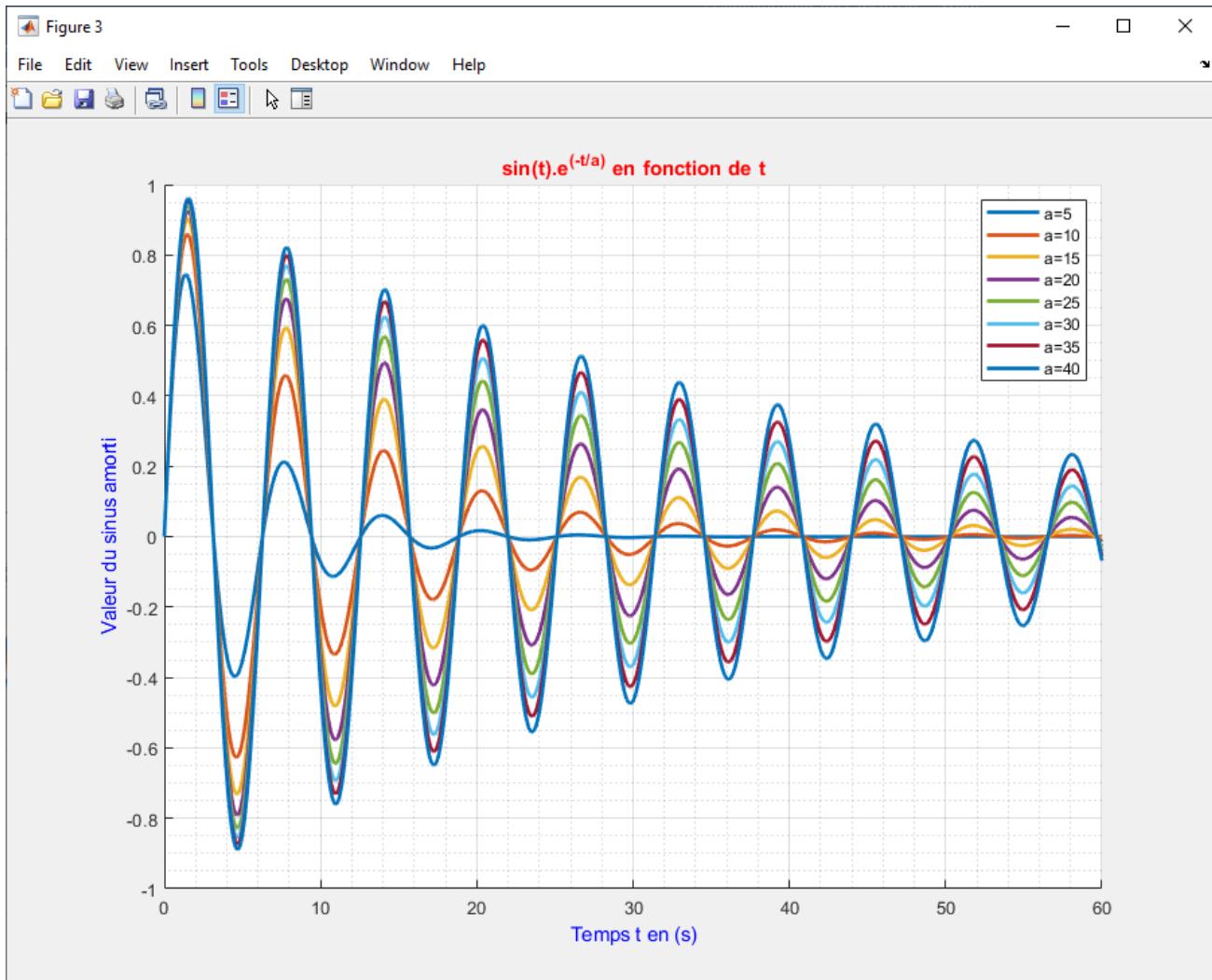


Figure 624 : représentation graphique avec légende automatique

Il est maintenant possible de relier chaque courbe à la valeur du paramètre **a** correspondant. L'équation du titre apparaît de manière plus explicite.

Il est également possible d'intégrer la fonction à tracer directement dans le script. Un exemple est donné dans le fichier « **script_0_legend_all_in_one.m** » (Figure 625). L'exécution de ce script donne un résultat rigoureusement identique.

```
%> script_0_legend_all_in_one.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction définie en fin de
% fichier

% création d'une nouvelle fenêtre graphique
figure;
hold on;

% création du vecteur des temps
t = [0:0.001:60];
```

```

%création d'une cellule pour stocker les lignes de la légende
Titre_Legende = cell(1,1);
ind_Legende = 1;

% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,f(t,k),'LineWidth',2);
Titre_Legende{1,ind_Legende} = 'a=' + string(k); % création de la ligne de la légende
ind_Legende = ind_Legende + 1;
end

% affichage de la légende
legend(Titre_Legende);

% affichage de la grille
grid on;
grid minor;

% titre et étiquettes des axes
titre = texlabel('sin(t).e^((-t/a)) en fonction de t');
title(titre,'Color','red');
xlabel('Temps t en (s)', 'Color', 'blue');
ylabel('Valeur du sinus amorti', 'Color', 'blue');

% définition de la fonction
function y=f(x,a)
y = sin(x).*exp(-x/a);
end

```

Figure 625 : script_0_legend_all_in_one

E. Création d'une fonction prenant en argument une autre fonction

Il est très fréquent d'avoir besoin de créer une fonction qui prendra en argument une autre fonction. A titre d'exemple, nous allons créer une fonction qui correspond à un besoin fréquent des utilisateurs de MATLAB. Afin de pouvoir formater rapidement et efficacement les graphiques que l'on doit tracer il peut être intéressant de créer une fonction qui permettra d'obtenir la représentation graphique d'une courbe en imposant un formatage du graphique prédéfini et qui sera donc le même pour toutes les courbes que l'on souhaite tracer. Pour cela nous allons créer une fonction **my_plot_function** qui prendra en argument une autre fonction **my_3_function** qui contient l'équation de la courbe.

La fonction **my_plot_function** (Figure 626) prendra plusieurs arguments en entrée :

- **a** : limite inférieure pour les abscisses
- **b** : limite supérieure pour les abscisses
- **nb_points** : nombres de points à calculer pour le tracé
- **f** : fonction à tracer
- **titre**: titre du graphique
- **Xtitle**: titre de l'axe des abscisses
- **Ytitle**: titre de l'axe des ordonnées

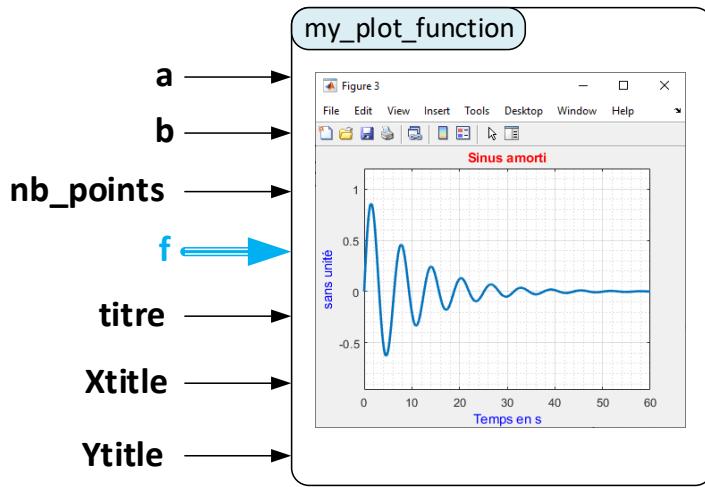


Figure 626 : my_plot_function

Nous pouvons remarquer que cette fonction ne renvoie aucune variable. Son action sera limitée à la génération d'une fenêtre graphique contenant la courbe à tracer.

Ouvrir le fichier contenant la fonction « **my_plot_function.m** » (Figure 627)

```
%% my_plot_function.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Trace une courbe avec une mise en forme prédéfinie
%
% my_plot_function(a,b,nb_points,f,titre,Xtitle,Ytitle)
%
% a : limite inférieure pour les abscisses
% b : limite supérieure pour les abscisses
% nb_points : nombres de points à calculer pour le tracé
% f : fonction à tracer
% titre: titre du graphique
% Xtitle: titre de l'axe des abscisses
% Ytitle: titre de l'axe des ordonnées

function my_plot_function(a,b,nb_points,f,titre,Xtitle,Ytitle)
% Création d'une fenêtre graphique
figure;
% création du vecteur des temps
t = [a:(b-a)/nb_points:b];
% tracé de la courbe et utilisation de la fonction prise en argument
plot(t,f(t), 'LineWidth',2);
%génération de la grille
grid on;
grid minor;
% titre et étiquette des axes
title(titre, 'Color', 'red');
```

```

xlabel(Xtitle,'Color','blue');
ylabel(Ytitle,'Color','blue');
% spécification des limites sur l'axe des abscisses
xlim([a,b]);
% spécification des limites sur l'axe des ordonnée
% axis donne les limites ymin et ymax
% les limites sur l'axe des ordonnées sont fixées à 20% au dessus des
% valeurs mini et maxi de la courbe
l = axis;
ymin = l(3); ymax = l(4);
ylim([1.2*ymin,1.2*ymax]);
end

```

Figure 627 : my_plot_function

Saisir la commande suivante dans la fenêtre de commande.

```

>> help my_plot_function

my_plot_function.m
Ivan LIEBGOTT @ Novembre 2019
Modélisation et Simulation des Systèmes Multi-Physiques
avec MATLAB - Simulink
Trace une courbe avec une mise en forme prédéfinie
my_plot_function(a,b,nb_points,f,titre,Xtitle,Ytitle)

a : limite inférieure pour les abscisses
b : limite supérieure pour les abscisses
nb_points : nombres de points à calculer pour le tracé
f : fonction à tracer
titre: titre du graphique
Xtitle: titre de l'axe des abscisses
Ytitle: titre de l'axe des ordonnées

```

La fenêtre de commande renvoie le commentaire placé en début de fonction de précédé du signe `%%`. Ce commentaire constitue une aide pour l'utilisateur de la fonction. Il est très important de bien documenter sa fonction en vue de son utilisation ultérieure.

Il faut maintenant créer la fonction **my_3_function** qui sera utilisée en argument : $y(x) = \sin(x).e^{-\frac{x}{10}}$

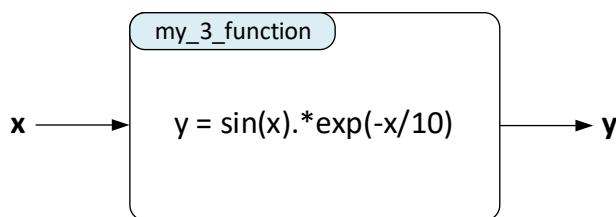


Figure 628 : my_3_function

Ouvrir un nouveau script et créer la fonction **my_3_function**.

Sauvegarder cette fonction sous le nom « **my_3_function.m** ».

```

+1 my_1_function.m x my_2_function.m x my_3_function.m x
1 function y = my_3_function(x)
2 y = sin(x).*exp(-x/10);
3 end

```

Figure 629 : my_3_function

Quand une fonction que l'on a codée utilise en argument une autre fonction, il est impératif de placer le signe « @ » devant la fonction qui est utilisée en argument. Pour utiliser la fonction **my_plot_function** avec **my_3_function** en argument, il faut taper la commande suivante dans la fenêtre de commande ou dans un script. La fonction sera tracée sur l'intervalle [0;60] et contiendra **1000 points**.

```
>> my_plot_function(0,60,1000,@my_3_function,'Sinus amorti','Temps en s','sans unité')
```

On obtient la courbe de la Figure 630.

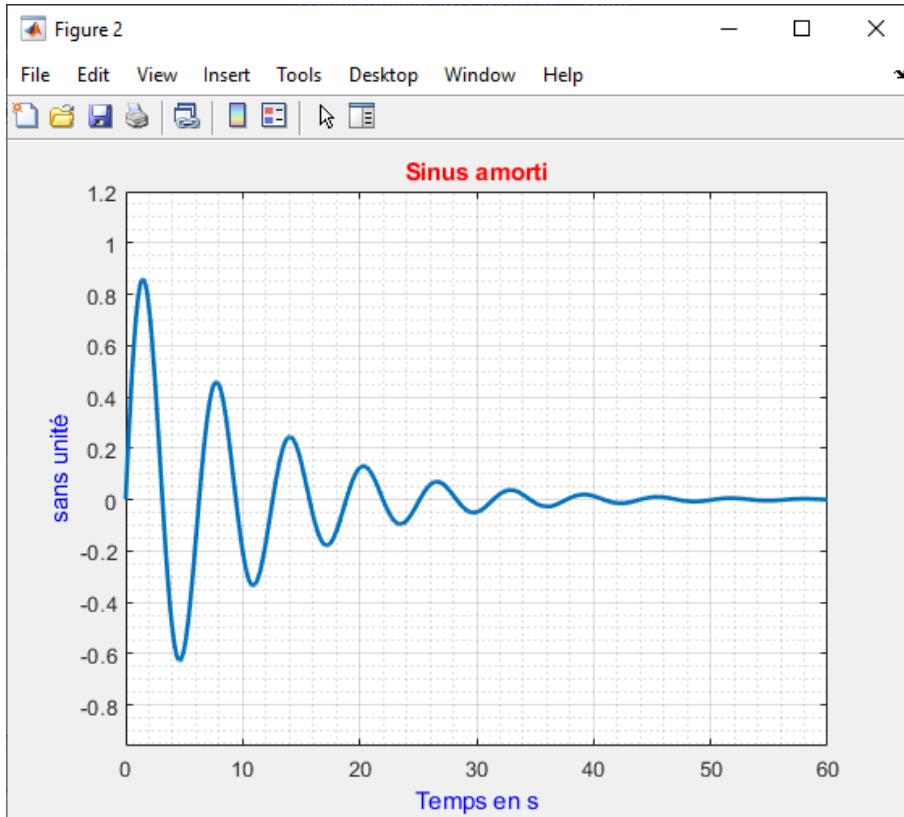


Figure 630 : appel de la fonction my_plot_function

Si l'on souhaite tracer une autre courbe en gardant la même mise en forme, il suffit de modifier l'équation contenu dans **my_3_function** et de faire un nouvel appel à la fonction **my_plot_function**.

Si l'on souhaite tracer : $y(x) = \sin(x) \cdot \cos^2(x)$, il faut modifier **my_3_function** (Figure 631).

```
+1 my_1_function.m x my_2_function.m x my_3_function.m x
1 function y = my_3_function(x)
2 y = sin(x).*cos(x).^2;
3 end
```

Figure 631 : modification de my_3_function

On fait alors un nouvel appel à **my_plot_function** pour un tracé de la courbe sur l'intervalle [0;20] avec **1000 points**.

```
>> my_plot_function(0,20,1000,@my_3_function,'Sinus amorti','Temps en s','sans unité')
```

On obtient alors le graphique de la Figure 632.

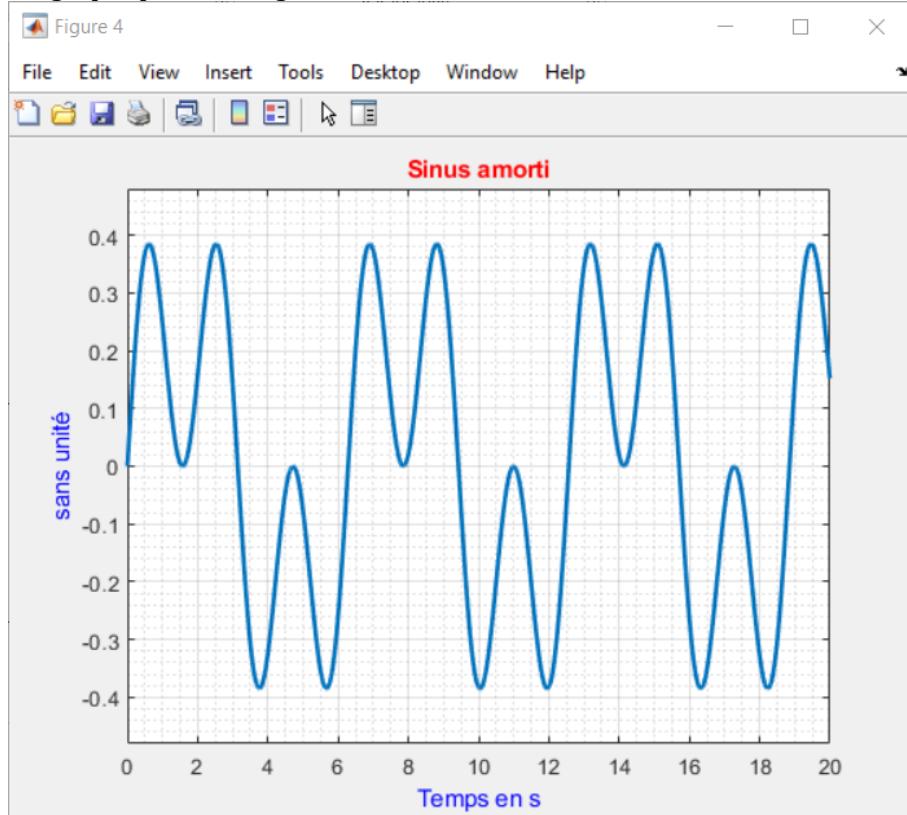


Figure 632 : appel de la fonction my_plot_function

Les mises en forme de la Figure 630 et de la Figure 632 sont strictement identiques et permettent d'uniformiser la présentation tout en accélérant le processus de construction des graphiques.

Il est également possible de créer une fonction ne prenant aucun argument en entrée et ne générant aucune variable en sortie. C'est le cas de **my_4_function** (Figure 633) dont le code est donné Figure 634.

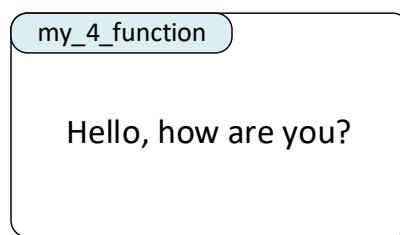


Figure 633 : my_4_function

```
+2 my_2_function.m x my_3_function.m x my_4_function.m x
1 function my_4_function
2 disp('hello, how are you?')
3 end
```

Figure 634 : codage de my_4_function

En appelant la fonction dans la fenêtre de commande, on obtient juste l'affichage d'un message.

```
>> my_4_function
hello, how are you?
```

F. Appel d'une fonction à plusieurs arguments

Supposons que l'on veuille trouver les racines du polynôme $y(x) = x^2 - 3$ en utilisant la fonction **my_5_function** (Figure 635) qui donne la définition générale d'un polynôme du second degré. Cette fonction qui prend 4 arguments est codée sur la Figure 636.

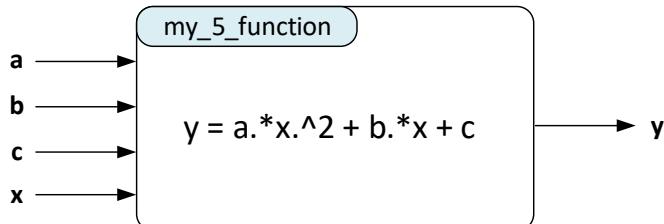


Figure 635 : my_5_function

```

+3 my_3_function.m x my_4_function.m x my_5_function.m x
1 function y = my_5_function(x,a,b,c)
2     y = a.*x.^2 + b.*x + c;
3 end
  
```

Figure 636 : codage de my_5_function

Ouvrir un nouveau script et créer la fonction **my_5_function**.
Sauvegarder cette fonction sous le nom « **my_5_function.m** ».

Nous utiliserons la fonction **fzero** (**fonction,x0**). La fonction **fzero** (Figure 637) donne la valeur du zéro de la **fonction** qui est donnée en premier argument au voisinage d'une valeur **x0**. La valeur de **x0** est donnée en second argument.

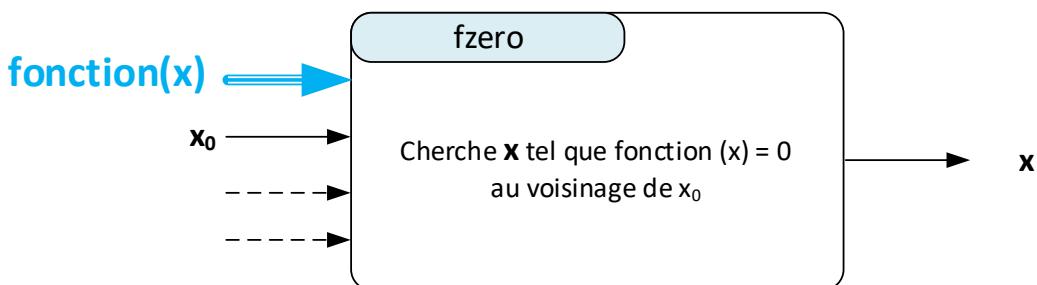


Figure 637 : la fonction fzero

La fonction **fzero** attend en argument une **fonction** à un seul argument (à une seule variable). Or **my_5_function** possède 4 arguments et si rien n'est précisé le calcul ne pourra pas être mené car **fzero** ne pourra pas faire la différence entre ces 4 arguments et ne pourra donc pas trouver la variable.

Il faut donc transformer **my_5_function** en fonction à un seul argument (**x**) en spécifiant tous les autres arguments à une valeur donnée (**a,b,c**). On utilisera encore le symbole **@**.

Taper dans la fenêtre de commande les instructions suivantes afin de trouver les racines de $y(x) = x^2 - 3$ au voisinage de $x_0=2$ et au voisinage de $x_0=-2$

```
>> racine1 = fzero(@(x)my_5_function(x,1,0,-3),2)
racine1 =
1.7321
>> racine2 = fzero(@(x)my_5_function(x,1,0,-3),-2)
racine2 =
-1.7321
```

Le symbole `@(x)` précise ici que l'unique argument pour la fonction **my_5_function** est la variable **x**, les autres variables étant fixées à une valeur donnée.

III. Déivation numérique

A. Aspects théoriques

La Figure 638 montre la courbe représentative de la fonction $f(x)$. La valeur de la dérivée en x_0 de la fonction $f(x)$ notée $f'(x_0)$ est donnée par la pente de la tangente à la courbe représentative de la fonction au point x_0 (repéré 1 sur la figure).

$$f'(x_0) = \frac{\Delta y}{\Delta x}$$

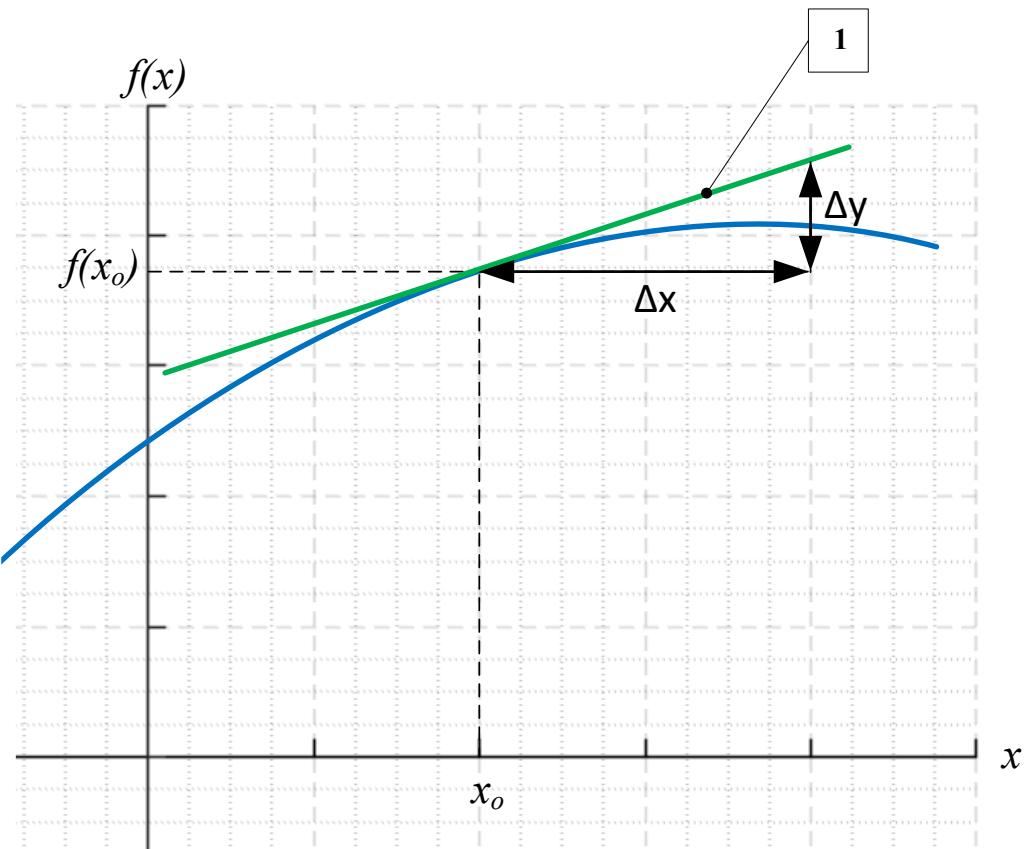


Figure 638 : calcul de la dérivée numérique au point x_0

Afin d'avoir une estimation numérique du coefficient directeur de cette tangente, il est nécessaire de choisir 2 points proches de x_0 et de calculer le coefficient directeur de la droite qui passe par ces deux points. Plusieurs choix sont alors possibles pour initier un processus de dérivation numérique.

1. Différence finie progressive

On choisit les points M_0 et M_+ (Figure 639) pour déterminer le coefficient directeur.

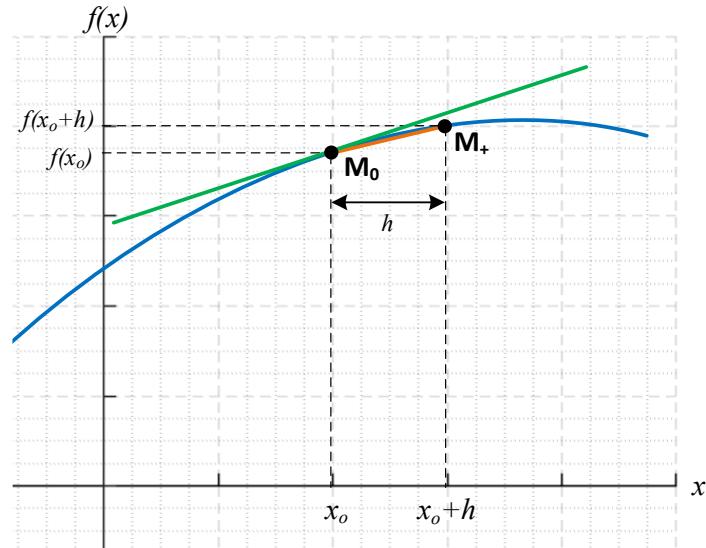


Figure 639 : différence finie progressive

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie progressive :

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} \quad (\text{h tend vers } 0)$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x_0 .

2. Différence finie rétrograde

On choisit les points M_- et M_0 (Figure 640) pour déterminer le coefficient directeur.

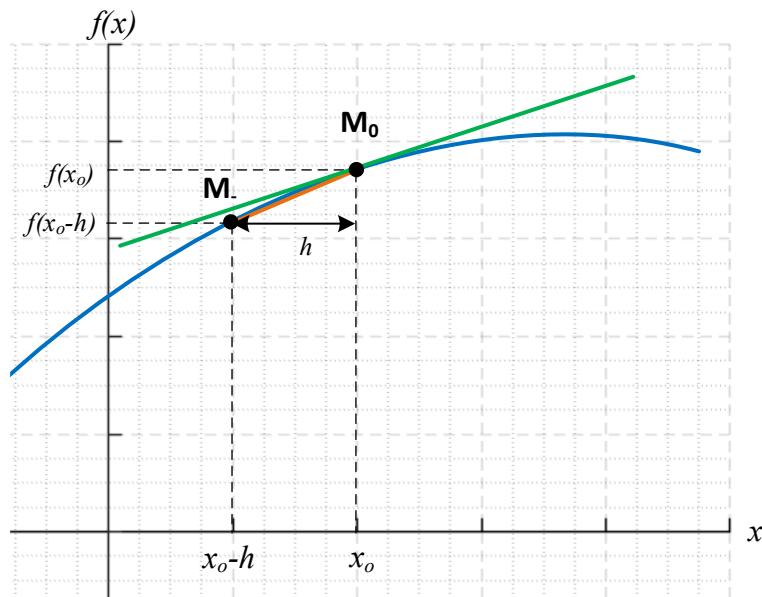


Figure 640 : formule de différence finie rétrograde

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie rétrograde :

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h} \quad (\text{h tend vers } 0)$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x_0 .

3. Différence finie centrée

On choisit les points M_1 et M_2 (Figure 641) pour déterminer le coefficient directeur.

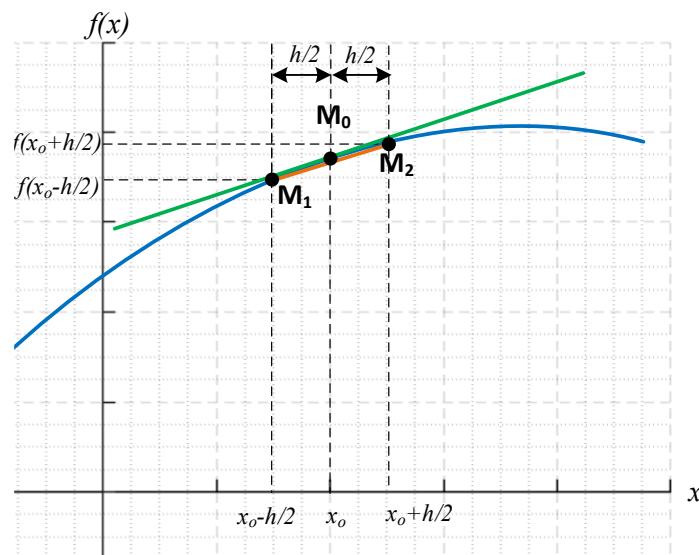


Figure 641 : formule de différence finie centrée

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie centrée :

$$f'(x_0) \approx \frac{f(x_0 + \frac{h}{2}) - f(x_0 - \frac{h}{2})}{h} \quad (\text{h tend vers } 0)$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x_0 .

B. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique avec MATLAB/ Dérivation numérique**.

L'expression mathématique de la fonction à dériver sera définie dans la fonction **f1.m** (Figure 642). Cette expression peut être modifiée pour tester toutes les fonctions souhaitées. Il faudra juste vérifier que lors de l'appel de la fonction l'intervalle d'étude [a ; b] appartienne bien au domaine de définition de la fonction.

```
%% f1.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f1.m contient l'expression mathématique de la fonction à dériver
%%
function y = f1(x)
y = sin(x);
end
```

Figure 642 : codage de la fonction f1 qui contient l'expression de la fonction à dériver

1. Différence finie progressive

Pour calculer la dérivée numérique en utilisant la méthode de la différence fine progressive, nous allons créer la fonction **fderiv_forward (a,b,nb_points,f)** (Figure 643).

Cette fonction prendra en arguments :

- a** : borne inférieure de l'intervalle d'étude
- b** : borne supérieure de l'intervalle d'étude
- nb_points** : nombre de points à calculer
- f** : fonction à dériver

La fonction aura comme sorties :

- t** : vecteur contenant les instants correspondant aux points où la dérivée est calculée
- df** : vecteur contenant les valeurs de la dérivée numérique au différents instants

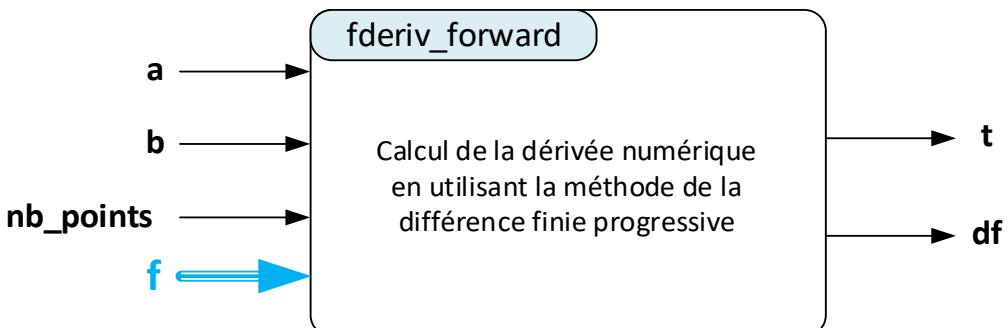


Figure 643 : fonction fderiv_forward (a,b,nb_point,f)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**.

L'objectif sera de comparer les erreurs effectuées lors de l'utilisation des différentes méthodes de dérivation numérique. Afin de pouvoir effectuer cette comparaison, nous devons prendre quelques précautions quant à la génération des sorties des différentes fonctions.

Pour la méthode de différence finie progressive : il n'est pas possible de calculer \mathbf{df} pour le dernier point de l'intervalle $[a ; b]$ correspondant à $\mathbf{t} = \mathbf{b}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{b} + \mathbf{h}$ (en dehors de l'intervalle d'étude)

Pour la méthode de différence finie rétrograde : il n'est pas possible de calculer \mathbf{df} pour le premier point de l'intervalle $[a ; b]$ correspondant à $\mathbf{t} = \mathbf{a}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{a} - \mathbf{h}$ (en dehors de l'intervalle d'étude)

Pour la méthode de différence finie centrée : il n'est pas possible de calculer \mathbf{df} pour le premier point de l'intervalle $[a ; b]$ correspondant à $\mathbf{t} = \mathbf{a}$ et pour le dernier point de l'intervalle $[a ; b]$ correspondant à $\mathbf{t} = \mathbf{b}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{a} - \mathbf{h}/2$ et en $\mathbf{t} = \mathbf{b} + \mathbf{h}/2$ (en dehors de l'intervalle d'étude)

Afin de pouvoir comparer les trois méthodes de dérivation, nous choisirons de travailler avec le même vecteur t pour les trois types de dérivation soit $t \in [a+h ; b-h]$. Par construction de nos algorithmes, le dernier point calculé correspondra à $\mathbf{t} = \mathbf{b} - \mathbf{h}$. Le premier point correspondant à $\mathbf{t} = \mathbf{a}$ devra être systématiquement supprimé afin de rendre la comparaison possible.

La Figure 644 montre le codage de la fonction **fderiv_forward** (a,b,nb_points,f).

```
%% [t,df] = fderiv_forward(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul la dérivée numérique de la fonction f sur l'intervalle [a;b] en
% prenant nb_points pour le calcul
% La méthode est celle de la différence finie progressive

%%
function [t,df] = fderiv_forward(a,b,nb_points,f)
% calcul du pas h
h = (b-a)/nb_points;

% définition du vecteur t
t = [a:h:b];

% calcul de la dérivée en chaque point
for k = 1:1:nb_points
    df(k) = (f(t(k+1))-f(t(k)))/h;
end

% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
% deux vecteurs
t(end)= [];

% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];
end
```

Figure 644 : codage de la fonction **fderiv_forward** (a,b,nb_point,f)

2. Différence finie rétrograde

Pour calculer la dérivée numérique en utilisant la méthode de la différence fine rétrograde, nous allons créer la fonction **fderiv_backward** (a,b,nb_points,f) (Figure 645).

Cette fonction prendra en arguments :

a : borne inférieure de l'intervalle d'étude
b : borne supérieure de l'intervalle d'étude
nb_points : nombre de points à calculer
f : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points où la dérivée est calculée

df : vecteur contenant les valeurs de la dérivée numérique au différents instants

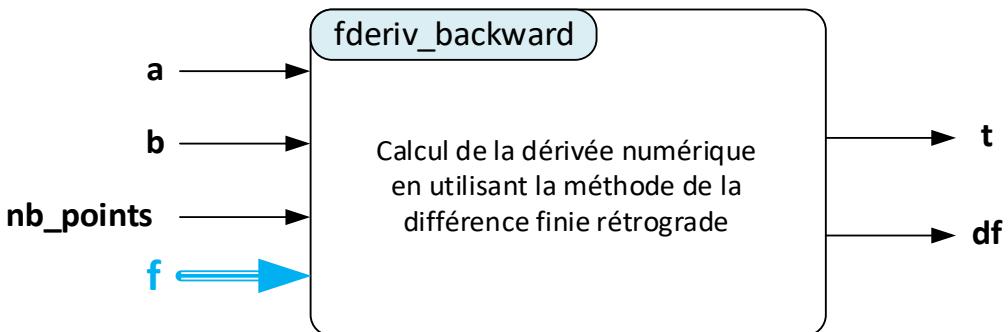


Figure 645 : fonction **fderiv_backward** (**a,b,nb_point,f**)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**.

Comme expliqué précédemment nous exploiterons **t** et **df** sur l'intervalle $t \in [a+h ; b-h]$. Le premier point correspondant à $t = a$ sera supprimé afin de rendre la comparaison entre les différentes méthodes de dérivation significative.

La Figure 646 montre le codage de la fonction **fderiv_backward** (**a,b,nb_points,f**).

```

%% [t,df] = fderiv_backward(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul la dérivée numérique de la fonction f sur l'intervalle [a;b] en
% prenant nb_points pour le calcul
% La méthode est celle de la différence finie rétrograde
%%
function [t,df] = fderiv_backward(a,b,nb_points,f)

% calcul du pas h
h = (b-a)/nb_points;

% définition du vecteur t
t = [a:h:b];

% calcul de la dérivée en chaque point
for k = 2:1:nb_points;
    df(k) = (f(t(k))-f(t(k-1)))/h;
end

% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
% deux vecteurs
t(end)= [];

% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];

```

end

Figure 646 : codage de la fonction fderiv_backward (a,b,nb_point,f)

3. Différence finie centrée

Pour calculer la dérivée numérique en utilisant la méthode de la différence finie centrée, nous allons créer la fonction **fderiv_center** (a,b,nb_points,f) (Figure 647).

Cette fonction prendra en arguments :

- **a** : borne inférieure de l'intervalle d'étude
- **b** : borne supérieure de l'intervalle d'étude
- **nb_points** : nombre de points à calculer
- **f** : fonction à dériver

La fonction aura comme sorties :

- **t** : vecteur contenant les instants correspondant aux points où la dérivée est calculée
- **df** : vecteur contenant les valeurs de la dérivée numérique au différents instants

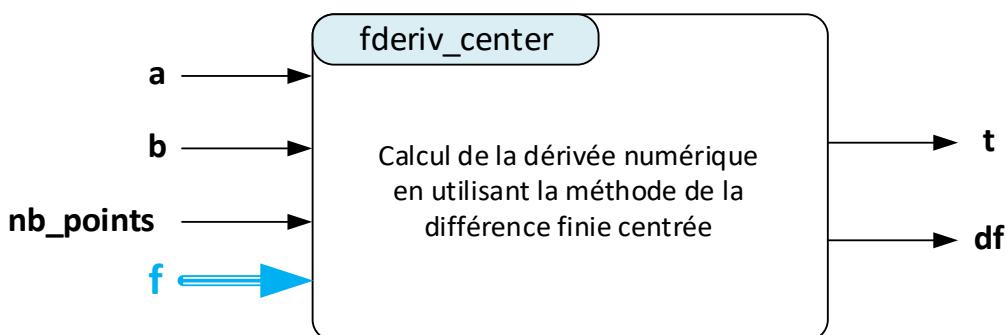


Figure 647 : fonction fderiv_center (a,b,nb_point,f)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**.

Comme expliqué précédemment nous exploiterons **t** et **df** sur l'intervalle $t \in [a+h ; b-h]$. Le premier point correspondant à $t = a$ sera supprimé afin de rendre la comparaison entre les différentes méthodes de dérivation significative.

La Figure 648 montre le codage de la fonction **fderiv_center** (a,b,nb_points,f).

```
%% [t,df] = fderiv_center(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul la dérivée numérique de la fonction f sur l'intervalle [a;b] en
% prenant nb_points pour le calcul
% La méthode est celle de la différence finie centrée

%%
function [t,df] = fderiv_center(a,b,nb_points,f)

% calcul du pas h
h = (b-a)/nb_points;

% définition du vecteur t
t = [a:h:b];
```

```

% calcul de la dérivée en chaque point
for k = 2:1:nb_points
    df(k) = (f(t(k)+h/2)-f(t(k)-h/2))/h;
end
% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
deux vecteurs
t(end)= [];

% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];
end

```

Figure 648 : codage de la fonction fderiv_center (a,b,nb_point,f)

4. Calcul exact de la fonction dérivée en utilisant le calcul symbolique

Afin de pouvoir évaluer l'erreur induite par les différentes méthodes de dérivation numérique, nous allons créer une fonction **fderiv_ana_symbolique** (a,b,nb_points,f) qui calcule la dérivée analytique exacte d'une fonction en calcul symbolique (Figure 649). Cette dérivée constituera la référence pour calculer l'erreur.

Cette fonction prendra en arguments :

- a** : borne inférieure de l'intervalle d'étude
- b** : borne supérieure de l'intervalle d'étude
- nb_points** : nombre de points à calculer
- f** : fonction à dériver

La fonction aura comme sorties :

- t** : vecteur contenant les instants correspondant aux points où la dérivée est calculée
- df** : vecteur contenant les valeurs de la dérivée numérique au différents instants

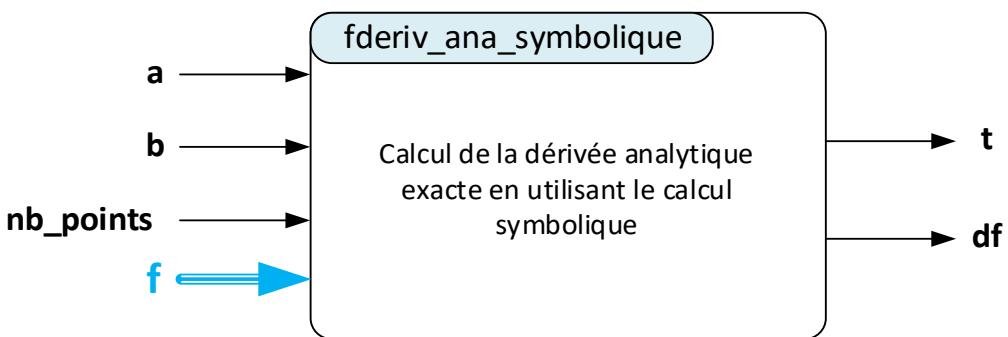


Figure 649 : fonction fderiv_ana_symbolique (a,b,nb_point,f)

Comme expliqué précédemment nous exploiterons **t** et **df** sur l'intervalle $t \in [a+h ; b-h]$. Il faudra donc supprimer le premier et le dernier point pour pouvoir effectuer la comparaison avec les autres méthodes de dérivation.

La Figure 650 montre le codage de la fonction **fderiv_ana_symbolique** (a,b,nb_points,f).

```

%% [t,df] = fderiv_anasymbolique(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la dérivée analytique exacte d'une fonction en utilisant le calcul
% symbolique
function [t,df]=fderiv_ana_symbolique(a,b,nb_points,f)
h = (b-a)/nb_points;

```

```

t = [a:h:b];
% x est déclarée comme variable symbolique
syms x;

% création de la fonction symbolique f_symbolique à partir de f(x)
f_symbolique(x) = f(x);

% dérivation de la fonction f_symbolique pour créer la fonction dérivée
% symbolique
df_symbolique(x) = diff(f_symbolique,x);

% calcul des valeurs du vecteur df qui contient les valeurs de la dérivée
% calculée analytiquement
df = df_symbolique(t);

% conversion du vecteur df en double précision
df = double(df);

df(1) = [];
t(1) = [];
df(end) = [];
t(end) = [];
end

```

Figure 650 : codage de la fonction fderiv_ana_symbolique (a,b,nb_point,f)

C. Exploitation de l'algorithme

1. Comparaison des différentes méthodes de dérivation numérique

Afin d'évaluer les écarts entre les différentes méthodes de dérivation numériques et la dérivée analytique calculée en calcul symbolique, nous allons créer un premier script qui va tracer ces 4 courbes dans la même fenêtre graphique, puis quantifier et tracer l'amplitude comparée de ces écarts.

Ouvrir le script **num_deriv_plot_comparaison.m**.

Le script est structuré en trois sections (Figure 651) :

- Définition de l'intervalle d'étude de la fonction $[a ; b]$ et du nombre de points **nb_points** à calculer pour tracer les courbes
- Appel des différentes fonctions et tracé des différentes courbes sur la même fenêtre graphique
- Evaluation des écarts entre la dérivée analytique et les trois méthodes de dérivation numérique et tracé des écarts sur la même fenêtre graphique

```

%% num_deriv_plot_comparison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%% Calcul de la dérivée numérique d'une fonction en utilisant différentes
% méthodes de calcul:
% - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
% Toutes les courbes sont placées sur le même graphiques afin de pouvoir
% évaluer visuellement les erreurs dues au processus de dérivation
% numérique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% et le nombre de points nb_points que l'on souhaite pour effectuer le
% calcul
% La fonction à dériver est stockée dans la fonction f1.m
%% Script

```

```

% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% définition du nombre de point à utiliser pour le calcul
nb_points = 40;

%% Calcul et tracé de la dérivée numérique avec les différentes méthodes

% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[2 8 18 12]);

grid on; grid minor; hold on;

% calcul et tracé de la dérivée exacte en calcul symbolique
[t_ana_symbolique,df_num_ana_symbolique] = fderiv_ana_symbolique(a,b,nb_points,@f1);
plot(t_ana_symbolique,df_num_ana_symbolique,'red','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie centrée
[t_center,df_num_center] = fderiv_center(a,b,nb_points,@f1);
plot(t_center,df_num_center,'green','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie progressive
[t_forward,df_num_forward] = fderiv_forward(a,b,nb_points,@f1);
plot(t_forward,df_num_forward,'blue','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie rétrograde
[t_backward,df_num_backward] = fderiv_backward(a,b,nb_points,@f1);
plot(t_backward,df_num_backward,'cyan','LineWidth',1);

% titre et légende
title({'Comparaison des méthodes de dérivation numérique',...
    'de la fonction f1'});

legend('dérivée analytique','différence finie centrée',...
    'différence finie progressive','différence fine rétrograde')
xlim([a b]);

%% Calcul des différents écarts en prenant comme référence la dérivée
% exacte déterminée en calcul symbolique

error_center = df_num_ana_symbolique - df_num_center;
error_forward = df_num_ana_symbolique - df_num_forward;
error_backward = df_num_ana_symbolique - df_num_backward;

% création du vecteur des abscisses
t = [a:(b-a)/nb_points:b];
t(1) = [];
t(end) = [];

% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[22 8 18 12]);
hold on; grid on; grid minor;

% tracé des différents écarts

plot(t,error_center,'green','LineWidth',2)
plot(t,error_forward,'blue','LineWidth',2)
plot(t,error_backward,'cyan','LineWidth',2)

% titre et légende

title({'Erreurs comparées des différentes méthodes de dérivation numérique';...
    'par rapport à la dérivée analytique'});

legend('différence finie centrée','différence finie progressive',...
    'différence fine rétrograde')

```

```
xlim([a b]);
```

Figure 651 : comparaison des différentes méthodes de dérivation numérique

Dans un premier temps le script sera exécuté sur l'intervalle $[0 ; 10]$ avec 40 points calculés. La fonction définie dans **f1.m** est la fonction $f(x) = \sin(x)$. Cette fonction peut-être modifiée par l'utilisateur.

L'exécution du script permet de visualiser les résultats du calcul numérique de la dérivée avec les différentes méthodes de calcul (Figure 652, Figure 653 et Figure 654).

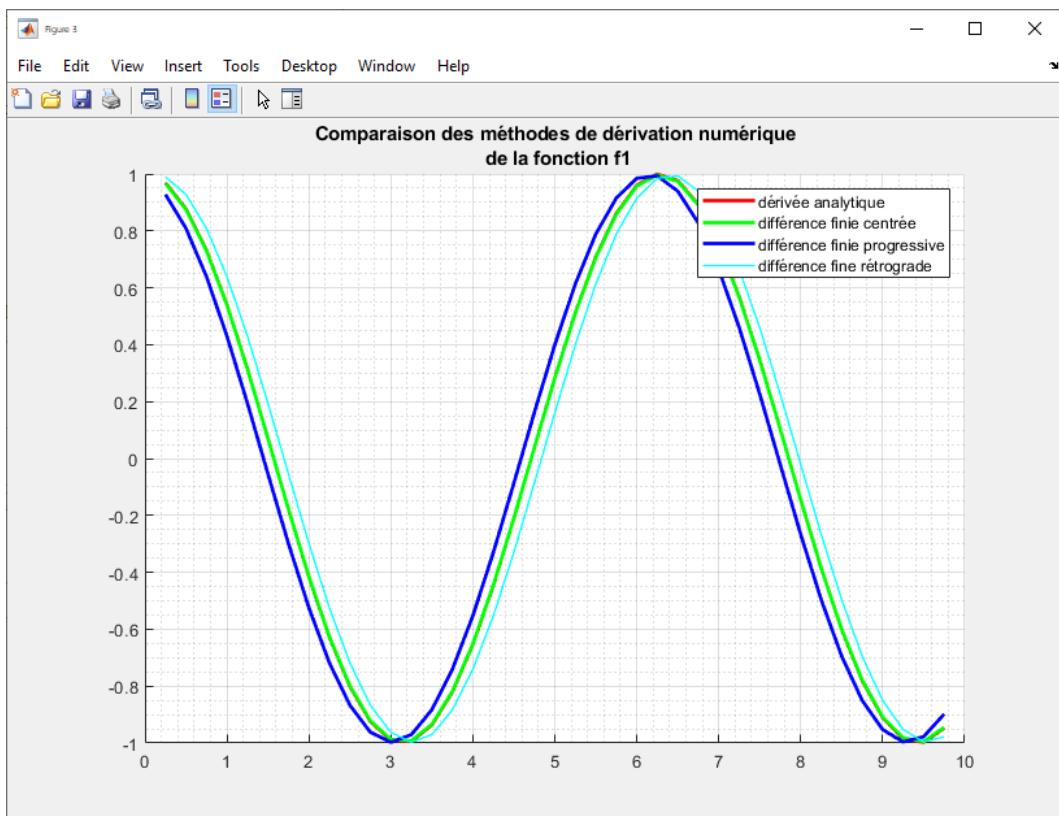


Figure 652 : comparaison des différentes méthodes de dérivation numérique

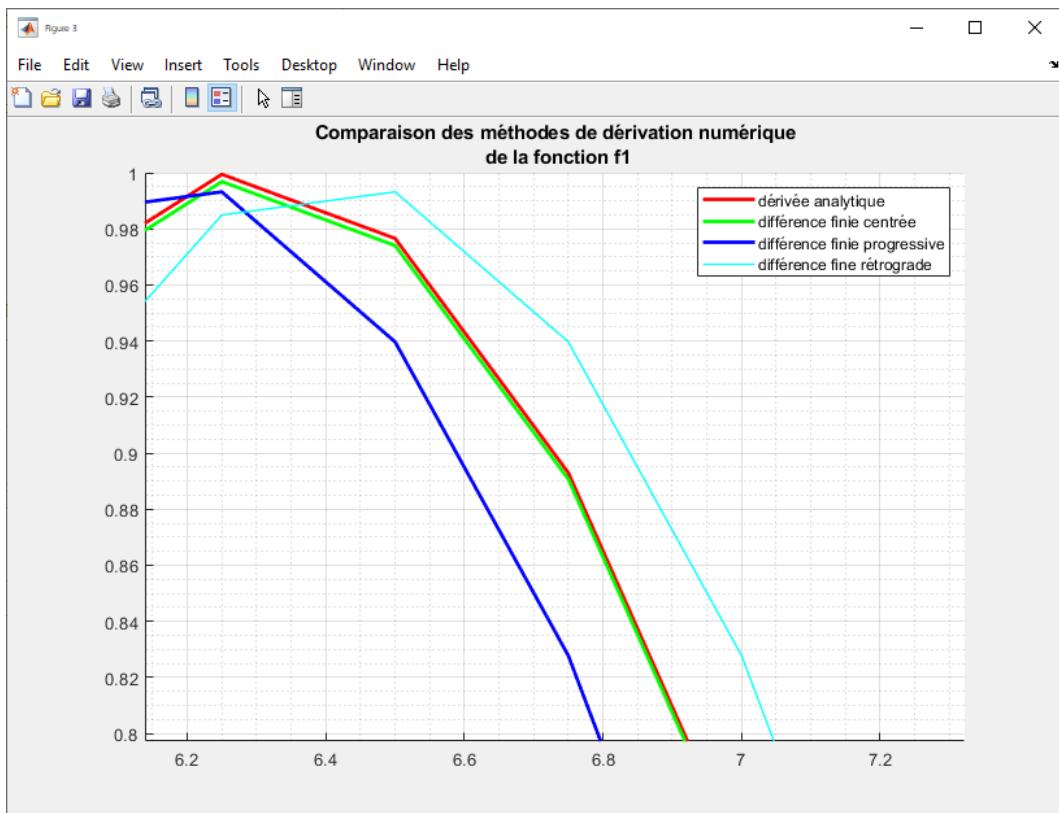


Figure 653 : zoom sur la courbe

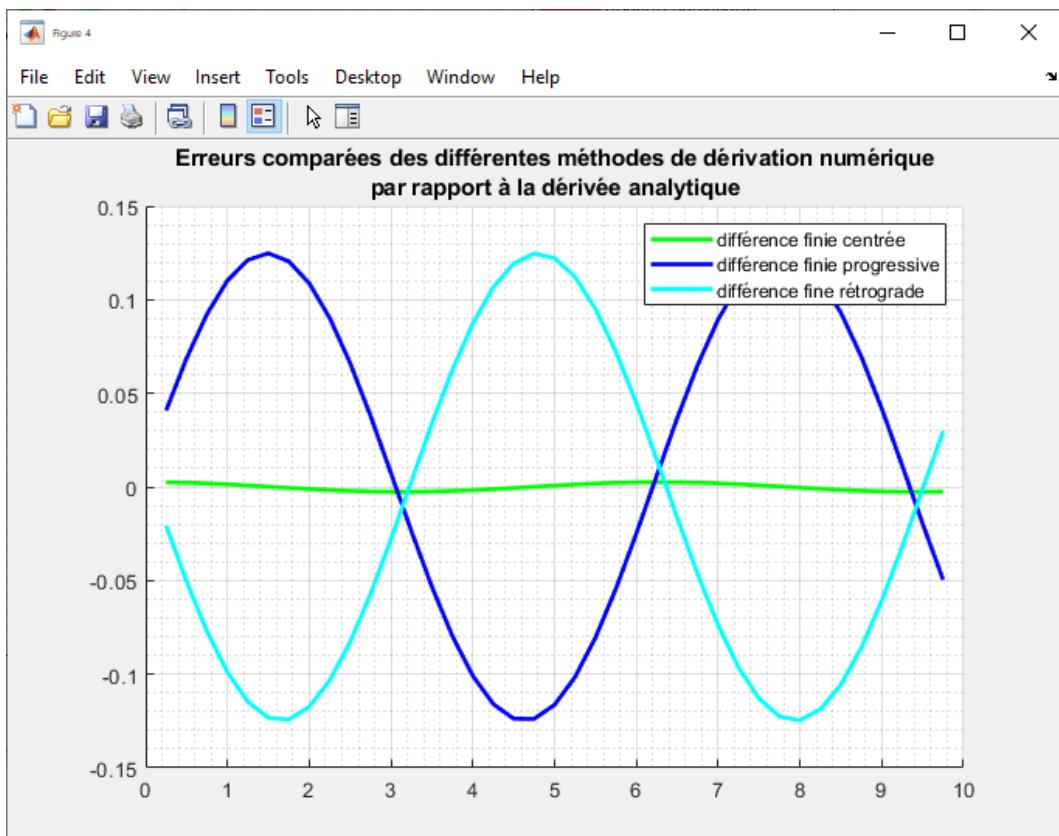


Figure 654 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique

L'analyse des différents courbes nous permet de dégager les tendances suivantes :

- La différence finie progressive est en avance sur la dérivée théorique exacte
- La différence finie rétrograde est en retard sur la dérivée théorique exacte
- Les erreurs induites par les méthodes de différences finies progressive et rétrograde semblent être du même ordre de grandeur
- L'erreur induite par la méthode de différence finie centrée semble significativement plus faible

2. Quantification et visualisation de l'erreur induite par les différentes méthodes

Il serait utile d'avoir une idée de l'évolution de cette erreur en fonction du nombre de point calculés et donc en fonction du pas $h = \frac{b-a}{nb_points}$.

Pour cela ouvrir le script **num_deriv_plot_comparaison_subplot.m**.

Ce script permet de tracer 4 courbes avec un nombre de points calculés différents pour chaque courbe et de voir ainsi l'évolution de l'erreur en fonction du nombre de points et du pas h choisi pour le calcul. Les 4 courbes correspondent à 4 valeurs du nombre de points calculés choisies équitablement réparties entre les variables **nb_points_min** et **nb_points_max** définies en début de script.

Le script est structuré en deux sections (Figure 655):

- Définition de l'intervalle d'étude de la fonction [a ;b] et du nombre de points **nb_points_min** et **nb_points_max**
- Appel des différentes fonctions et tracé des différentes courbes sur 4 fenêtres graphiques différentes

```
%% num_deriv_plot_comparison_subplot
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%
% Calcul de la dérivée numérique d'une fonction en utilisant différentes
% méthodes de calcul:
% - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% Représentation de 4 courbes correspondant à 4 valeurs du nombres de
% points calculés choisies équitablement réparties entre nb_points_min et
% nb_points_max
% La fonction à dériver est stockée dans la fonction f1.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% spécification des nombres de points mini et maxi
nb_points_min = 10;
nb_points_max = 50;
pas = round((nb_points_max - nb_points_min) / 3);
%%
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig,'Color','white');
shtitle({'Comparaison des différentes méthodes de dérivation numérique',...
    'de la fonction f1 en fonction du nombre de points calculés'});
%
% initialisation du compteur de boucle
```

```

k = 1;

for nb_points = nb_points_min : pas : nb_points_max;
subplot(2,2,k); hold on; grid on; grid minor;

% calcul et tracé de la dérivée exacte en calcul symbolique
[t_ana_symbolique,df_num_ana_symbolique] = fderiv_ana_symbolique(a,b,nb_points,@f1);
plot(t_ana_symbolique,df_num_ana_symbolique,'red','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie centrée
[t_center,df_num_center] = fderiv_center(a,b,nb_points,@f1);
plot(t_center,df_num_center,'green','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie progressive
[t_forward,df_num_forward] = fderiv_forward(a,b,nb_points,@f1);
plot(t_forward,df_num_forward,'blue','LineWidth',2);

% calcul et tracé de la dérivée avec différence finie rétrograde
[t_backward,df_num_backward] = fderiv_backward(a,b,nb_points,@f1);
plot(t_backward,df_num_backward,'cyan','LineWidth',1);

% légend('dérivée analytique','différence finie centrée',...
%       'différence finie progressive','différence fine rétrograde');

title('Nombre de points calculés ' + string(nb_points))
k = k + 1;
end

% spécification de la légende générale du graphique
leg = legend('dérivée analytique','différence finie centrée',...
    'différence finie progressive','différence fine rétrograde');

% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg,'Position', newPosition, 'Units', newUnits);

```

Figure 655 : script permettant de visualiser l'influence du pas

L'exécution du script permet de visualiser l'influence du pas sur l'allure des courbes (Figure 656).

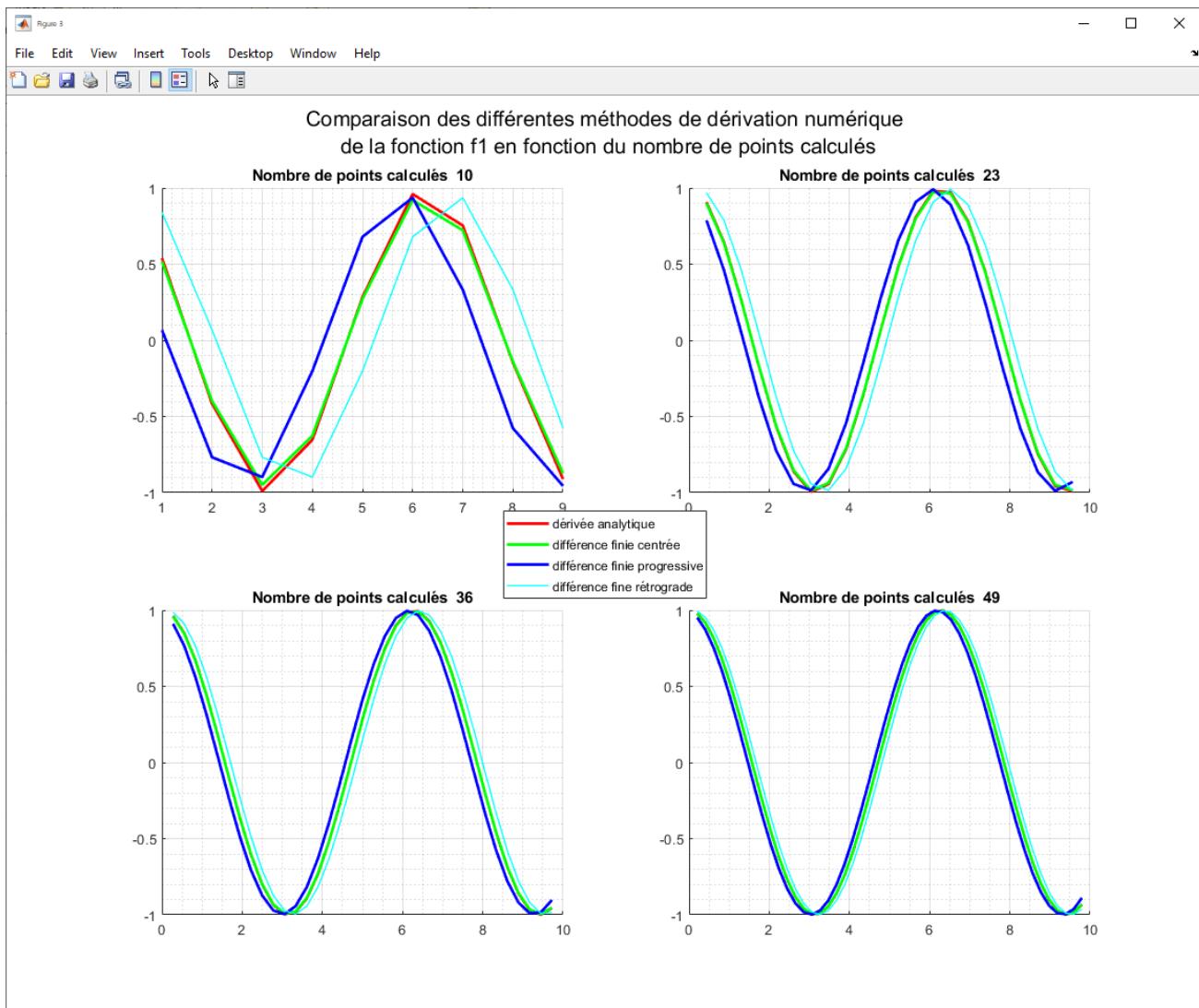


Figure 656 : influence du pas sur l'allure des courbes

L'analyse de ces courbes nous confirme que l'erreur va diminuer avec le pas mais ne nous permet pas de connaître la loi d'évolution de cette erreur en fonction du pas h . Lors de l'utilisation d'un processus de dérivation numérique, il est important de déterminer cette loi d'évolution.

3. Détermination de la loi d'évolution de l'erreur en fonction du pas

Pour connaître la loi d'évolution de cette erreur, il suffit de se placer en un point x_0 arbitraire de l'intervalle et de calculer l'erreur pour différentes valeurs du pas h .

L'erreur en fonction du pas h se définit de la manière suivante :

- Pour la différence finie progressive :

$$f_{err_forward}(h) = \left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right|$$

- Pour la différence finie rétrograde :

$$f_{err_backward}(h) = \left| f'(x_0) - \frac{f(x_0) - f(x_0 - h)}{h} \right|$$

- Pour la différence finie centrée :

$$f_{err_center}(h) = \left| f'(x_0) - \frac{f(x_0 + \frac{h}{2}) - f(x_0 - \frac{h}{2})}{h} \right|$$

L'utilisation de la formule de Taylor pourrait nous permettre de trouver la forme mathématique de ces fonctions erreurs. L'objectif ici est plutôt d'exploiter notre algorithme et les capacités de calcul de MATLAB afin de déterminer cette forme.

Pour cela nous allons construire un script qui se placera arbitrairement au centre de notre intervalle d'étude et qui va tracer automatiquement l'erreur en fonction du pas h. Le script contient une boucle qui calculera les différentes fonctions erreurs pour un nombre de points allant de 100 à 1000 avec un incrément de 100. Le pas h est défini par $h = \frac{b-a}{nb_points}$.

La boucle permettra donc de calculer 10 valeurs de l'erreur au point x_0 pour chacune des trois méthodes de dérivation numérique. Le script permet ensuite d'interpoler les points calculés avec une fonction polynomiale adaptée pour trouver l'ordre de variation des fonctions erreurs.

Ouvrir le script **num_deriv_plot_error.m** (Figure 657).

```
%% num_deriv_plot_error
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%
% Calcul de la forme mathématique de l'erreur effectuée lors des
% opérations de dérivation numériques pour différentes méthodes de calcul:
% - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
%
% On se place arbitrairement au centre de l'intervalle d'étude de la
% fonction en un point x0 afin d'évaluer l'erreur en ce point pour
% différentes valeurs de h
% et du nombre de points de calcul
% on utilise ensuite une interpolation polynomiale pour déterminer la forme
% mathématique de l'erreur en fonction de h
% La fonction à dériver est stockée dans la fonction f1.m
%
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% initialisation du compteur qui servira d'indice
k = 1;

%% On définit ici une boucle qui permettra de calculer l'erreur pour un
% échantillon de points allant de 100 points jusqu'à 1000 points par pas
% de 100 pour toutes les méthodes de dérivation

for nb_points = 100:100:1000

    % calcul de la dérivée exacte en utilisant le calcul symbolique
    [t_ana_symbolique,df_num_ana_symbolique] = fderiv_ana_symbolique(a,b,nb_points,@f1);

    % calcul de la dérivée avec différence finie centrée
    [t_center,df_num_center] = fderiv_center(a,b,nb_points,@f1);

    % calcul et tracé de la dérivée avec différence finie rétrograde
```

```

[t_backward,df_num_backward] = fderiv_backward(a,b,nb_points,@f1);

% calcul et tracé de la dérivée avec différence finie progressive
[t_forward,df_num_forward] = fderiv_forward(a,b,nb_points,@f1);

% calcul des erreurs au point x0, milieu de l'intervalle d'étude en prenant
% comme référence la dérivée exacte déterminée en calcul symbolique
x0 = round(nb_points/2);
error_f_center_h(k) = df_num_ana_symbolique(x0) - df_num_center(x0);
error_f_forward_h(k) = df_num_ana_symbolique(x0) - df_num_forward(x0);
error_f_backward_h(k) = df_num_ana_symbolique(x0) - df_num_backward(x0);

% détermination de la valeur du pas h(k) pour l'itération en cours
h(k) = (b-a)/nb_points;

% incrémentation de k
k = k + 1;
end

%% Tracé et interpolation de l'erreur pour la différence finie centrée
% Création d'une nouvelle figure pour les graphiques
figure('Units','centimeters','Position',[2 8 18 12]);
hold all;grid on; grid minor;
plot(h,error_f_center_h,'sm','MarkerSize',10,'LineWidth',2);

% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_center = polyfit (h,error_f_center_h,2);

% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_center = polyval(coeff_poly_inter_center,t);
plot(t,poly_inter_center,'LineWidth',2);

% Ecriture de la légende
legend('différence finie centrée',...
    string(coeff_poly_inter_center(3)) + ' h^2 + ' ...
    + string(coeff_poly_inter_center(2)) + ' h + ' ...
    + string(coeff_poly_inter_center(1)),...
    'Location','northwest','FontSize',12);

% Ecriture du titre et des étiquettes des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour la différence finie centrée'}); 

xlabel('pas h');
ylabel('erreur');

%% Tracé et interpolation de l'erreur pour la différence finie progressive
figure('Units','centimeters','Position',[22 8 18 12]);
hold all;grid on; grid minor;
plot(h,error_f_forward_h,'sb','MarkerSize',10,'LineWidth',2)

% Détermination du polynôme d'interpolation pour la différence finie
% progressive

% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_forward = polyfit (h,error_f_forward_h,1);

% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_forward = polyval(coeff_poly_inter_forward,t);
plot(t,poly_inter_forward,'LineWidth',2);

%% Tracé et interpolation de l'erreur pour la différence finie rétrograde
plot(h,error_f_backward_h,'og','MarkerSize',10,'LineWidth',2)

```

```
% Détermination du polynôme d'interpolation pour la différence finie
% rétrograde

% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_backward = polyfit (h,error_f_backward_h,1);

% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_backward = polyval(coeff_poly_inter_backward,t);
plot(t,poly_inter_backward,'LineWidth',2);

% écriture de la légende
legend('différence finie progressive',...
    string(coeff_poly_inter_forward(1)) + ' h + ' ...
    + string(coeff_poly_inter_forward(2)),...
    'différence finie rétrograde',...
    string(coeff_poly_inter_backward(1)) + ' h + ' ...
    + string(coeff_poly_inter_backward(2)),...
    'Location','northwest','FontSize',12);

% titre et étiquette des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour les différences finies progressive et retrograde'}); 

xlabel('pas h');
ylabel('erreur');
```

Figure 657 : script permettant de connaître l'ordre de variation des fonctions erreurs

Exécutez le script et analyser les courbes obtenues sur la Figure 658 et sur la Figure 659.

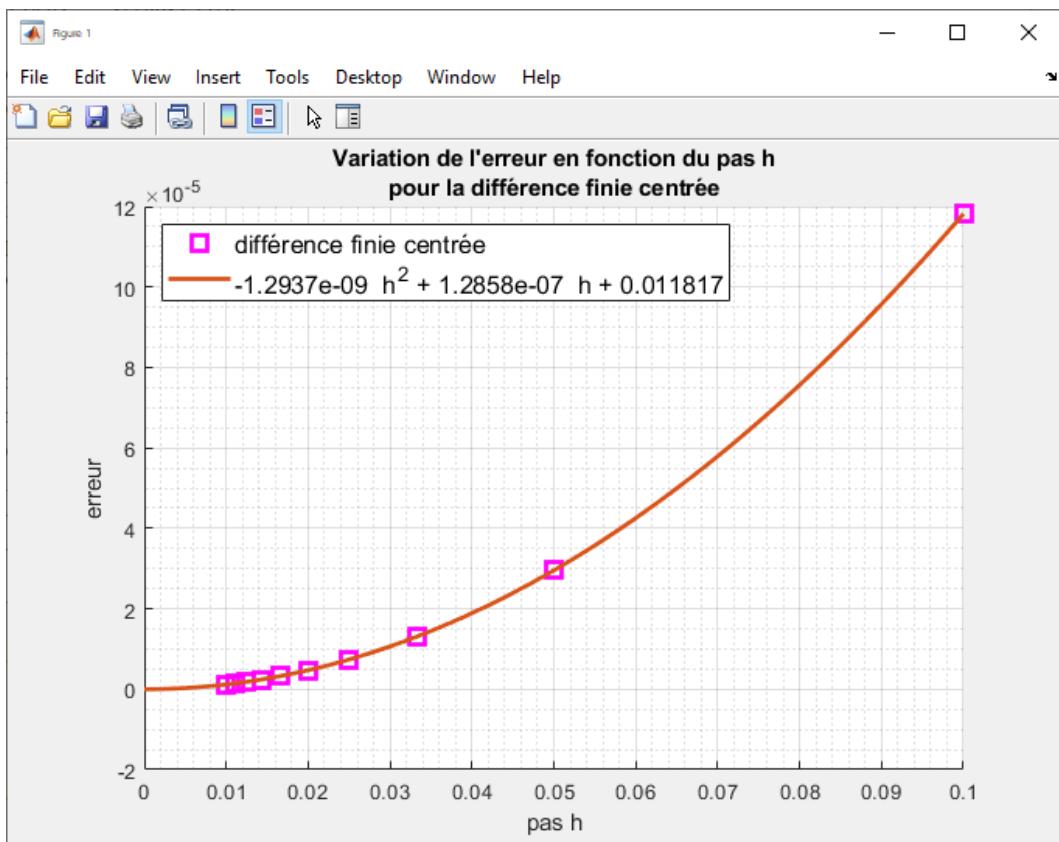


Figure 658 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée

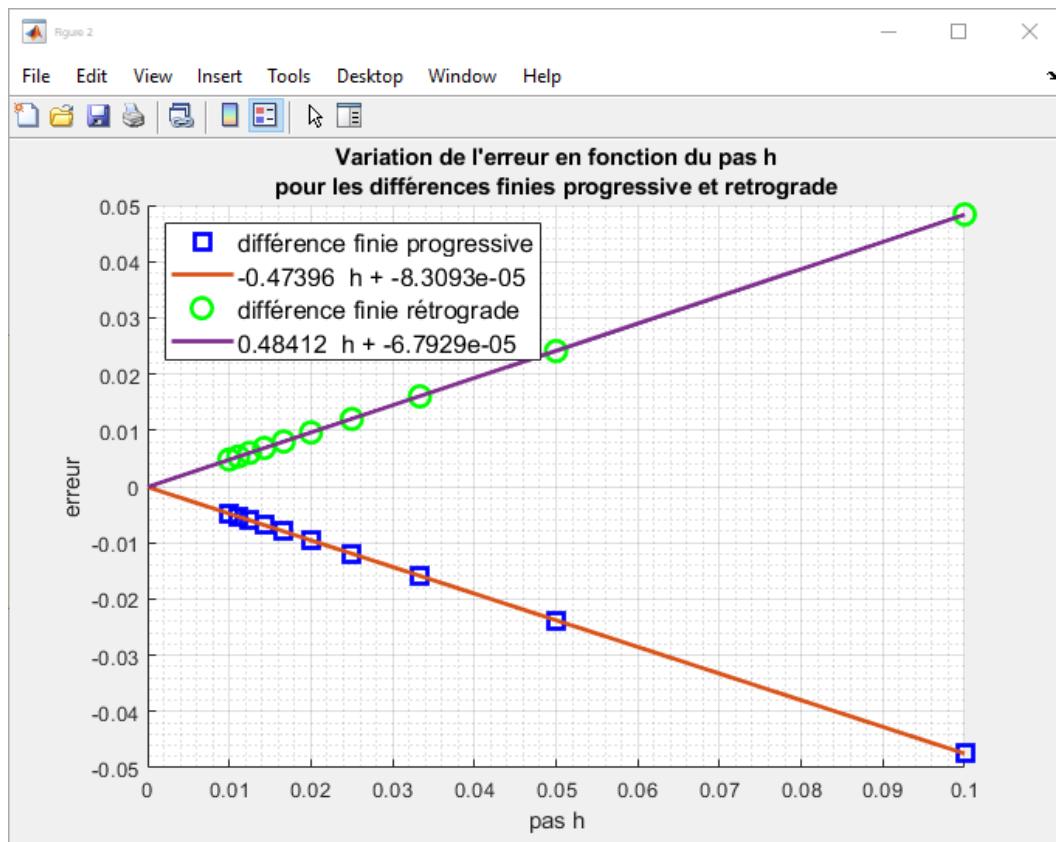


Figure 659 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde

En observant les points qui représentent l'erreur en fonction du pas h , nous choisissons une régression linéaire pour la différence finie progressive et rétrograde et une régression polynomiale d'ordre 2 pour la différence finie centrée. Il apparaît que ces choix permettent d'interpoler la série de données avec une excellente précision. Les équations des polynômes d'interpolation apparaissent dans les légendes des différentes fenêtres graphiques.

En conclusion :

- l'erreur pour la différence finie progressive et pour la différence finie rétrograde est d'ordre 1 en h
- l'erreur pour la différence finie centrée est d'ordre 2 en h

L'exploitation d'un algorithme peut donc nous amener à connaître la tendance de variation d'un paramètre en exploitant la puissance de calcul et en utilisant les outils adaptés. Notre seule intuition est parfois insuffisante et nous devons nous appuyer sur le potentiel de notre algorithme. Ces tendances peuvent évidemment être démontrées mathématiquement en utilisant la formule de Taylor qui confirmerait ces observations.

Les concepts que nous avons implantés dans notre algorithme sont très simples. Une exploitation pertinente de ces concepts nous permet de dégager des tendances que seuls des concepts mathématiques avancés peuvent prévoir.

4. Robustesse du processus

Afin de tester la robustesse de notre algorithme, nous pouvons tester plusieurs fonctions en modifiant la fonction **f1.m** et en reproduisant le processus.

Modifier la fonction **f1.m** pour reproduire l'analyse sur une fonction polynomiale $y(x) = 5x^3 - 3x^2 + 5x + 1$ (Figure 660).

```
function y = f1(x)
y = 5*x^3-3*x^2+5*x+1;
end
```

Figure 660 : modification de la fonction f1.m

Relancer les différents scripts pour visualiser les résultats dans les différentes figures.

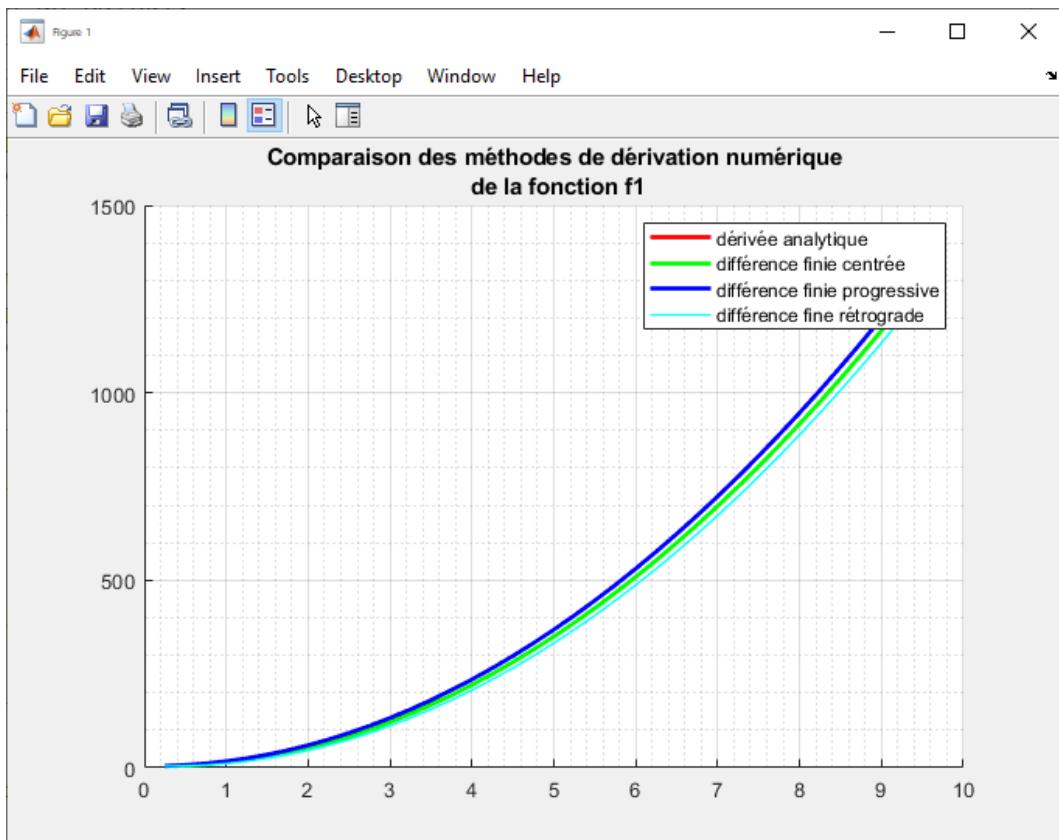


Figure 661 : comparaison des différentes méthodes de dérivation numérique

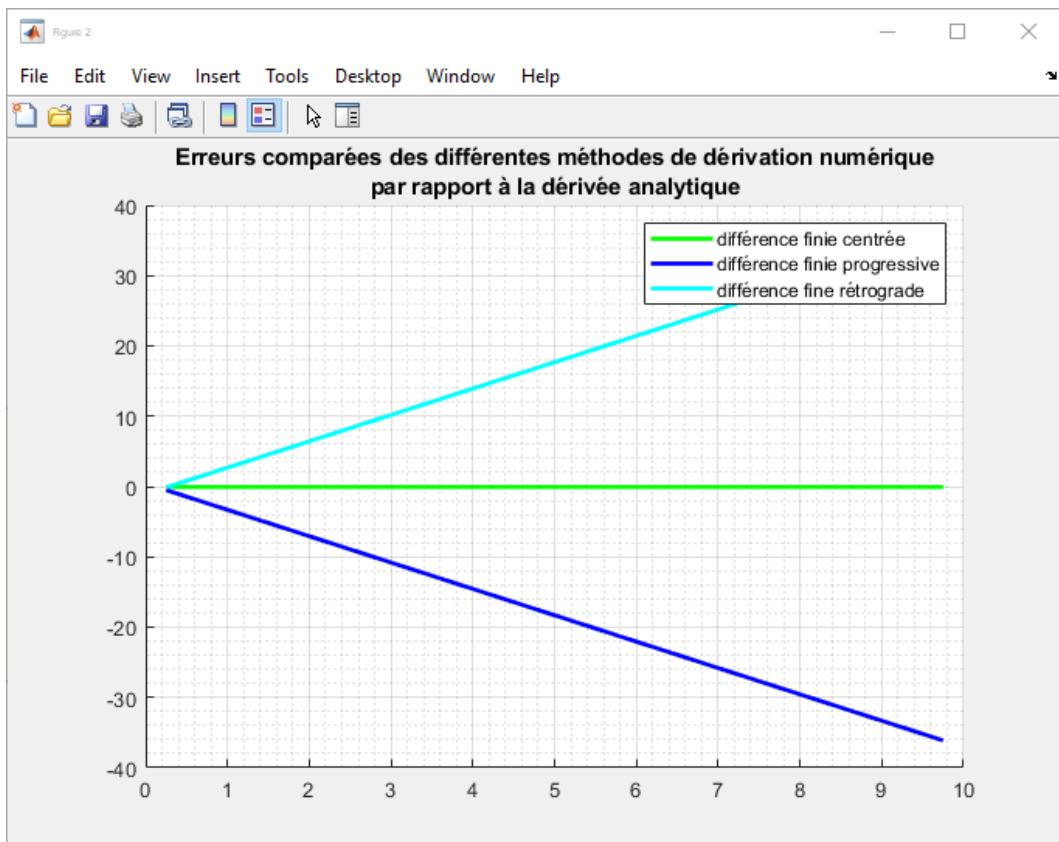


Figure 662 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique

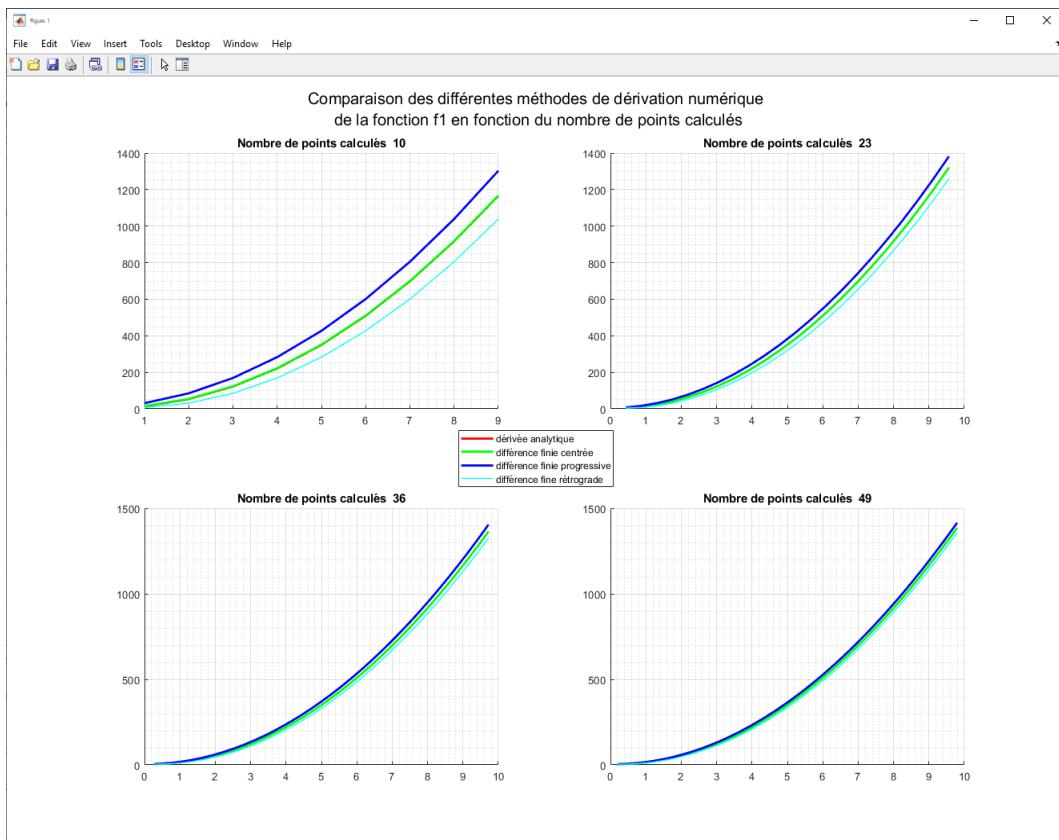


Figure 663 : influence du pas sur l'allure des courbes

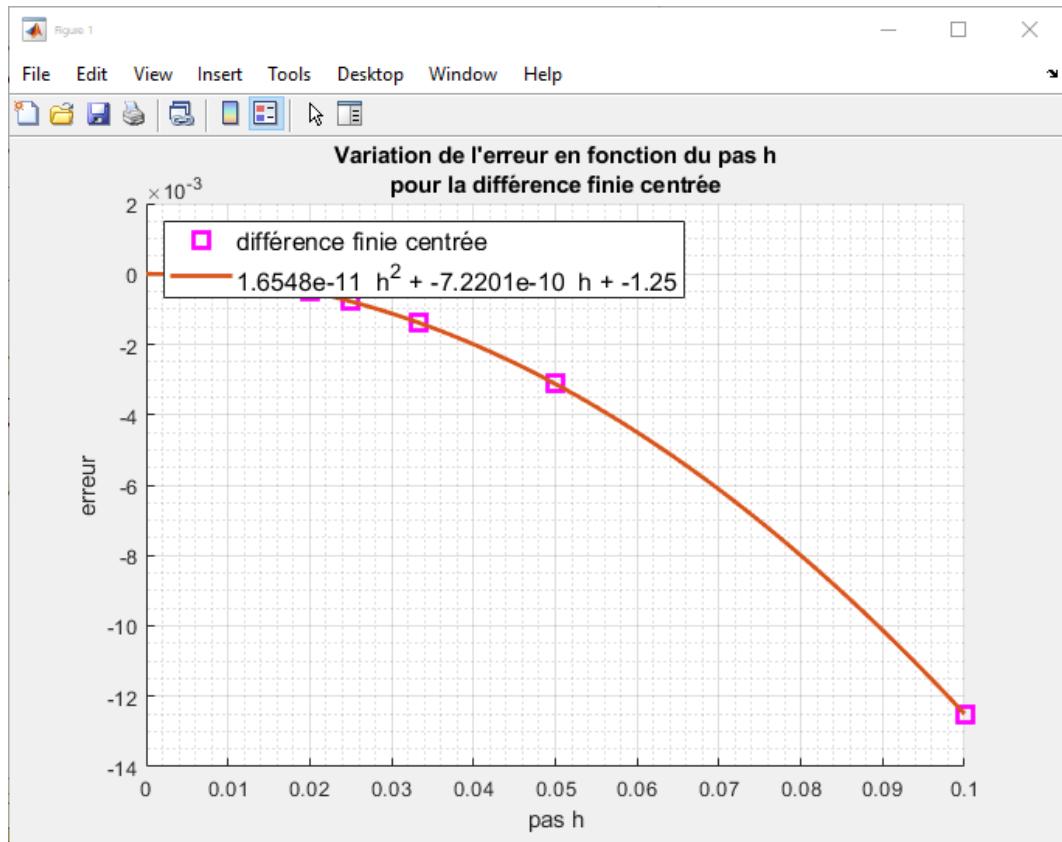


Figure 664 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée

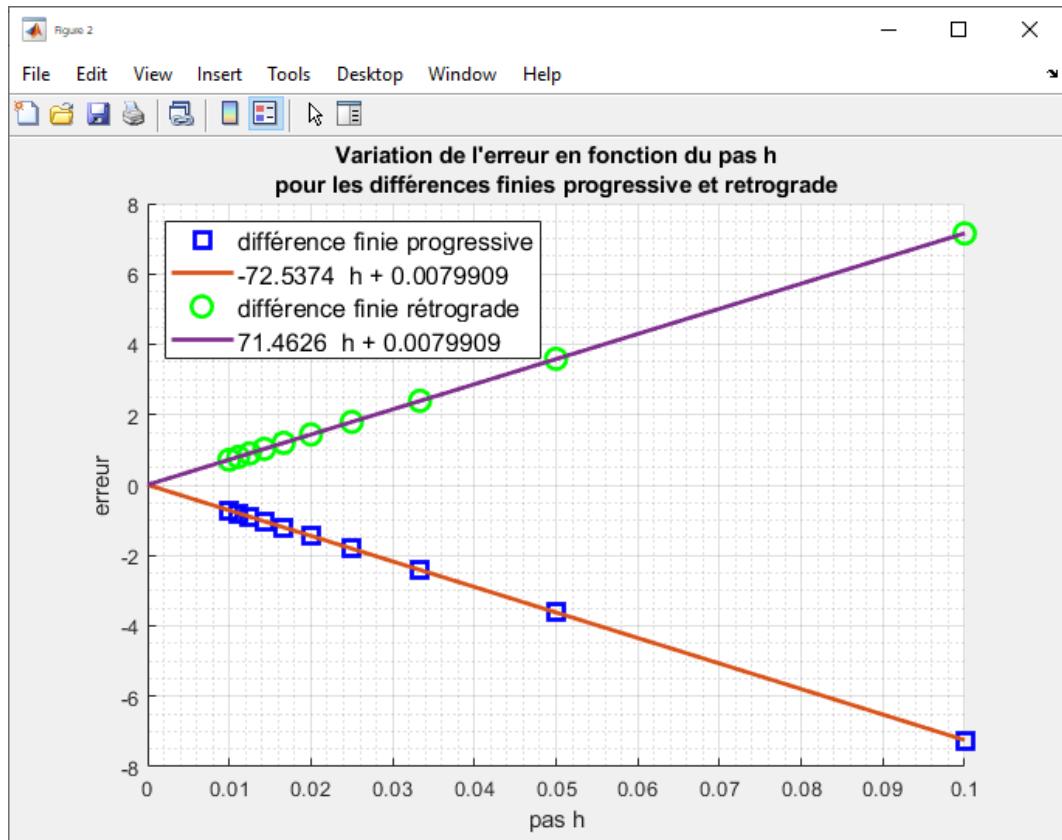


Figure 665 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde

D. Application de la dérivation numérique

1. Dérivation du signal issu d'un codeur incrémental

Le signal relevé sur la Figure 666 (zoom sur la Figure 667) représente la position mesurée à l'aide d'un codeur incrémental placé sur l'arbre d'un moteur. Le signal une fois traité présente de nombreuses discontinuités qui seront très préjudiciables lors d'un processus de dérivation numérique. Si nous voulons obtenir la vitesse de rotation de l'arbre du moteur en fonction du temps, il faudra dériver numériquement le signal mais il ne sera pas possible de le faire sur le signal brut.

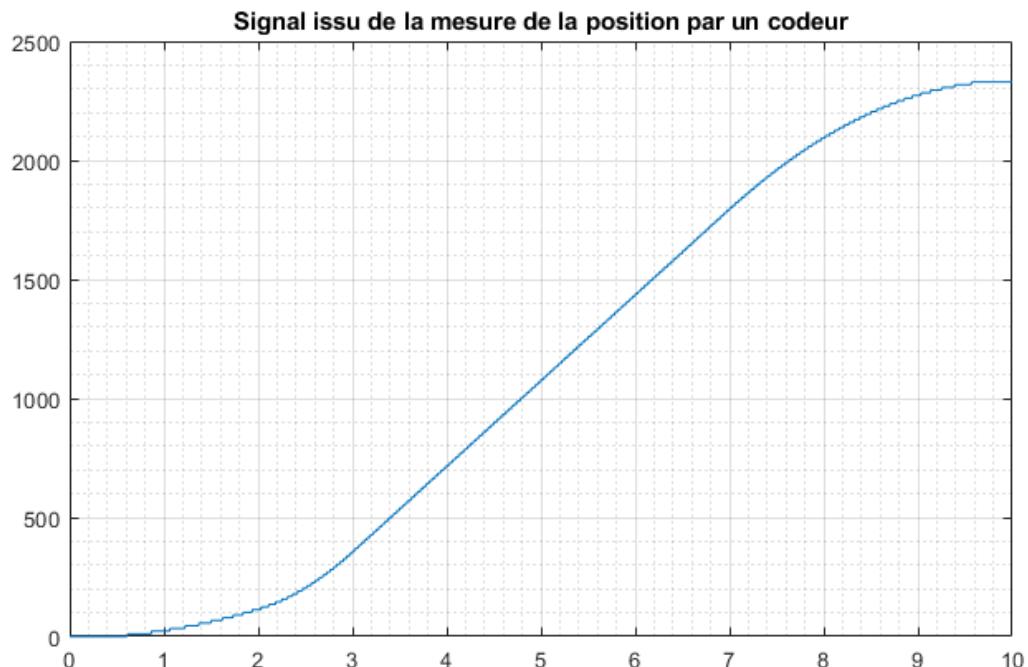


Figure 666 : mesure de la position à partir du signal d'un codeur incrémental

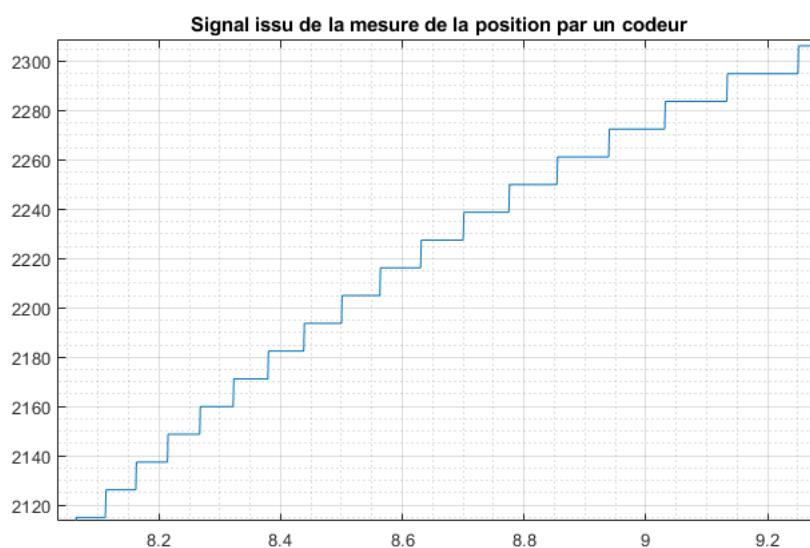


Figure 667 : zoom sur la courbe

2. Mise en évidence des problèmes rencontrés

Lorsqu'un signal présente des discontinuités importantes ou lorsqu'il est fortement bruité, le processus de dérivation numérique simple ne pourra pas donner un résultat satisfaisant. En effet la Figure 668 montre la variation très forte des coefficients directeurs calculés par la méthode des différences finies sur les intervalles ①, ②, ③ et ④. Sur les intervalles ① et ② la dérivée numérique est nulle alors qu'elle est très élevée sur l'intervalle ③.

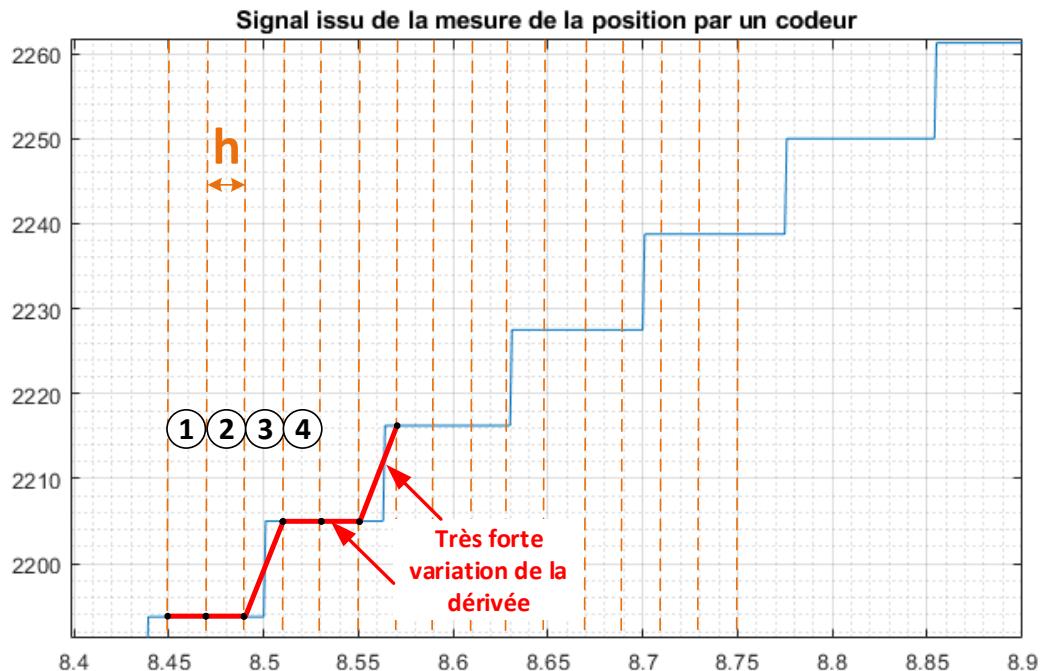


Figure 668 : influence du pas d'échantillonnage sur le calcul de la dérivée numérique

L'application directe d'un algorithme de dérivation numérique sur le signal brut nous donne le résultat visible sur la Figure 669. Le signal obtenu montre les variations intempestives du coefficient directeur calculé sur une distance correspondant au pas h . Cette méthode n'est donc pas satisfaisante.

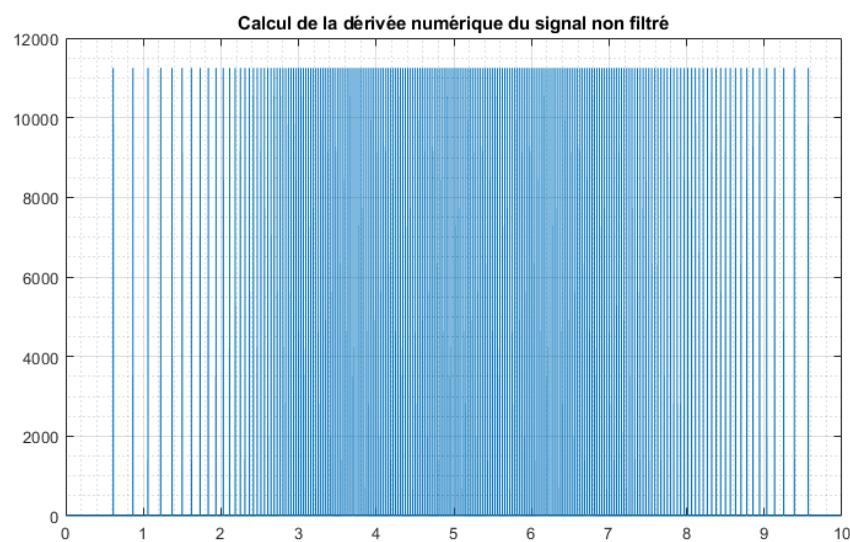


Figure 669 : dérivation numérique du signal brut

3. Dérivation numérique à pas variable

Afin d'obtenir une allure correcte pour notre signal obtenu par dérivation numérique, nous allons allonger le pas de dérivation afin de limiter au maximum les variations intempestives de la dérivé.

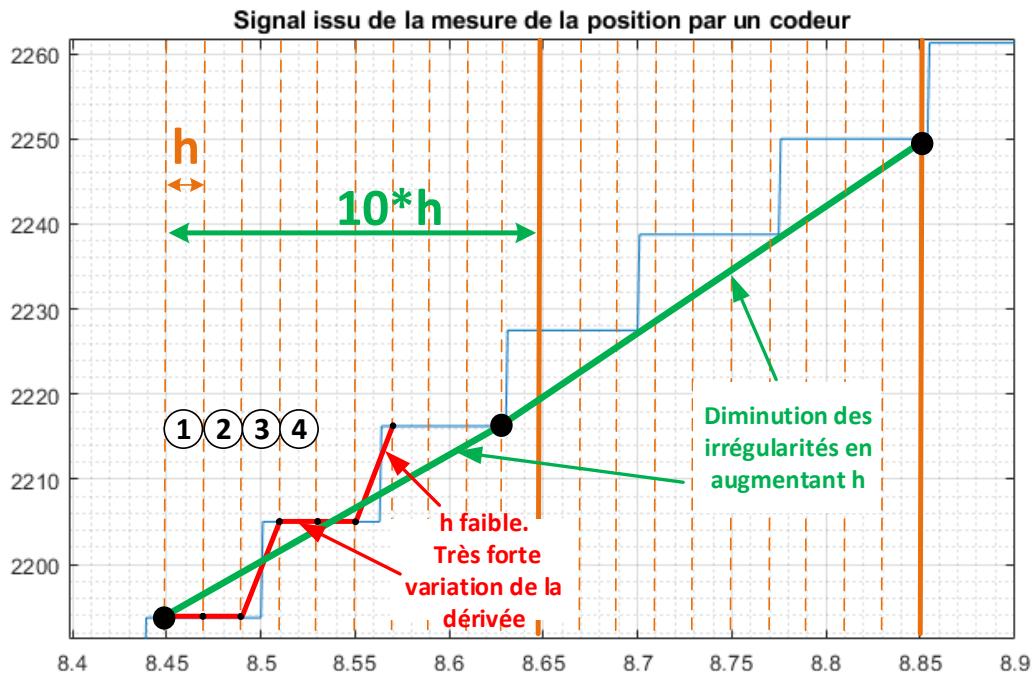


Figure 670 : influence de l'augmentation du pas de dérivation

Pour dériver un signal avec pas variable nous allons utiliser la fonction **fderiv_num_pas_variable(t,Y,step_size)** (Figure 671).

Cette fonction prendra en arguments :

- **t** : vecteur contenant les abscisses du signal à dériver
- **Y** : vecteur contenant les valeurs du signal à dériver
- **step_size_points** : nombre de points à considérer pour construire le pas de dérivation

La fonction aura comme sorties :

- **t_step** : vecteur contenant les instants correspondant aux points où la dérivée est calculée
- **dY_step** : vecteur contenant les valeurs de la dérivée numérique au différents instants

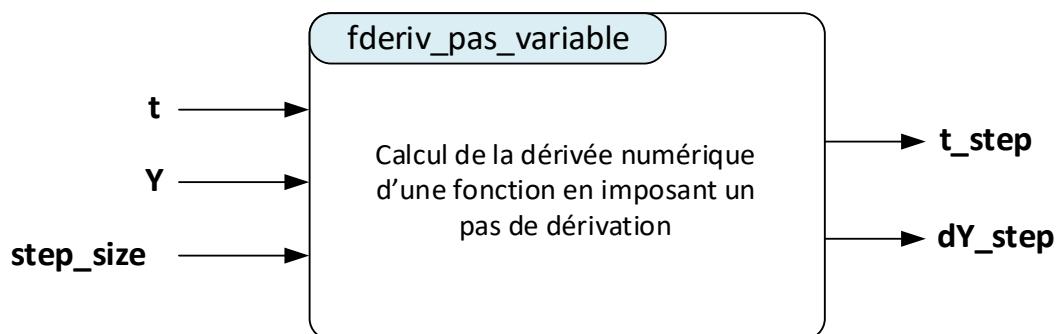


Figure 671 : fonction **fderiv_num_pas_variable(t,Y,step_size)**

Le codage de la fonction **fderiv_num_pas_variable(t,Y,step_size)**

```

%% fderiv_pas_variable(t,Y,step_size)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de réaliser une dérivation numérique en choisissant
% la longueur du pas de dérivation
% step_size représente la longueur du pas de dérivation

function [t_step,dY_step] = fderiv_num_pas_variable(t,Y,step_size)

% Evaluation du nombre de points des vecteurs
nb_points = length(t);

% calcul du pas h
h = (t(end)-t(1))/(nb_points - 1);

% calcul du pas à utiliser pour l'opération de dérivation numérique
pas = step_size * h;

% calcul de la dérivée en chaque point
i = 1;
% construction des vecteurs t_step et dY_step
for k = 1:step_size:(nb_points-1-step_size)
    dY_step(i) = (Y(k+step_size)-Y(k))/pas;
    t_step(i) = t(k+step_size);
    i = i + 1;
end
end

```

Figure 672 : codage de la fonction fderiv_num_pas_variable(t,Y,step_size)

Afin de tester cette fonction et de voir l'influence du pas de dérivation sur la forme du signal, ouvrir le script **deriv_num_pas_variable_comparaison.m**. Ce script permet de tracer dans une même fenêtre graphique 4 courbes représentant la dérivée numérique calculée avec 4 valeurs du pas de dérivation. Il est possible de modifier les variables **nb_points_min** et **nb_points_max**.

```

%% deriv_num_pas_variable_comparaison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Représentation de 4 courbes correspondant à 4 dérivations numériques du
% signal en considérant différents pas de dérivation choisis équitablement
% répartis entre nb_points_min et nb_points_max
%% Effacement des variables du workspace
clear all;

% chargement dans le workspacedes données correspondant au signal du codeur et au
% temps
load num_deriv_data.mat;

% tracé du signal du codeur
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig,'Color','white');
shtitle({'Position donnée par le codeur en fonction du temps',...
    ''});
plot(t,position_codeur);
grid on; grid minor;

% spécification des nombres de points mini et maxi
nb_points_min = 50;

```

```

nb_points_max = 450;
pas = round((nb_points_max - nb_points_min) / 3);
%%
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig,'Color','white');
sgtitle({'Analyse de l''influence du pas de dérivation';...
    ' sur la forme du signal à dériver'});

% initialisation du compteur de boucle
k = 1;

for step_size = nb_points_min : pas : nb_points_max;
    subplot(2,2,k); hold on; grid on; grid minor;

    % calcul et tracé de la dérivée exacte en calcul symbolique
    [t_step,dY_step] = fderiv_num_pas_variable(t,position_codeur,step_size);
    plot(t_step,dY_step,'blue','LineWidth',2);

    title('La longueur du pas couvre ' + string(step_size) + ' points')
    k = k + 1;
end

% spécification de la légende générale du graphique
leg = legend('signal dérivé');

% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg,'Position', newPosition, 'Units', newUnits);

```

Figure 673 : script permettant de faire varier le pas de dérivation

Exécuter le script et observer le résultat sur la Figure 674.

Analyse de l'influence du pas de dérivation
sur la forme du signal à dériver

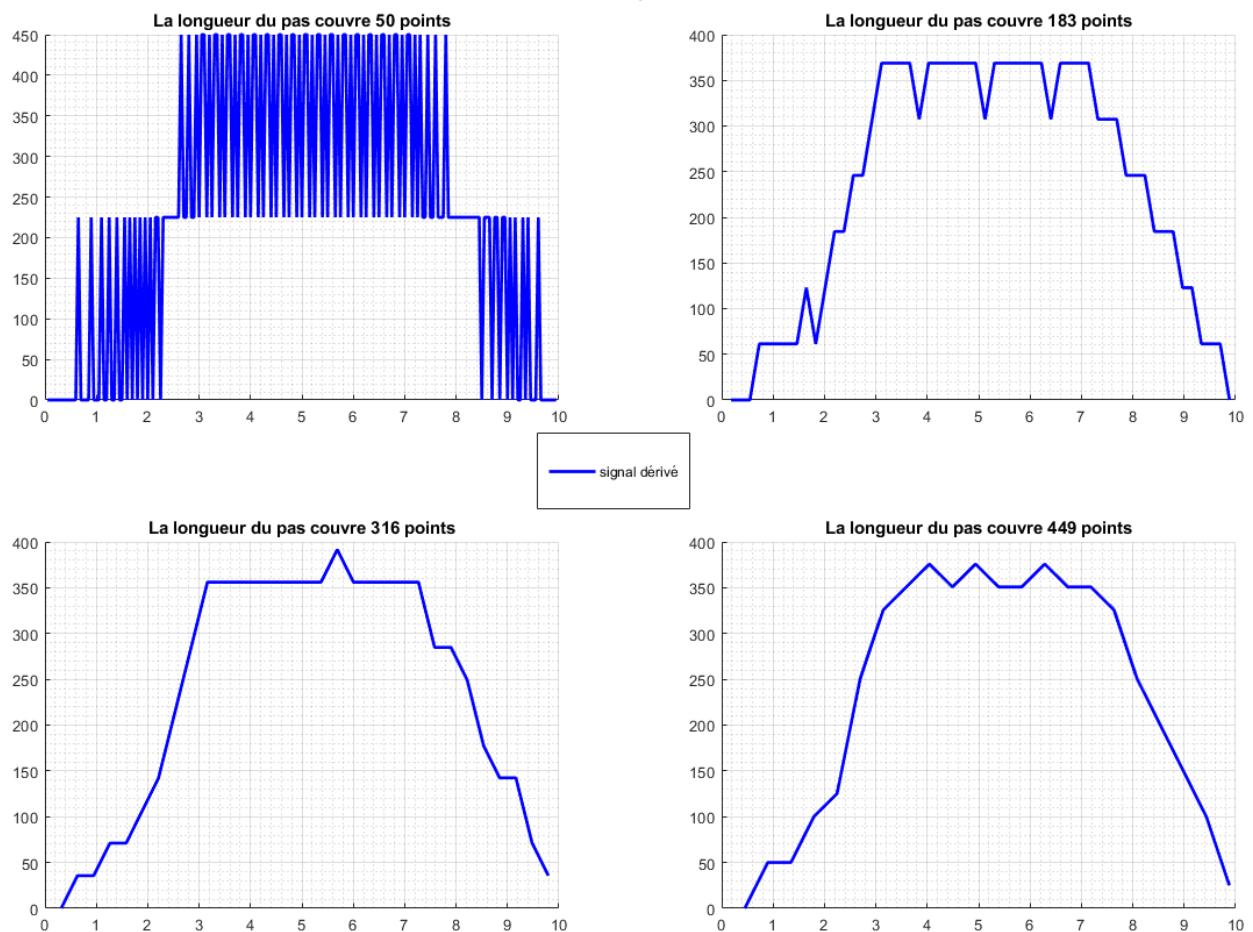


Figure 674 : influence du pas de dérivation numérique

Nous pouvons constater que l'augmentation du pas de dérivation permet d'obtenir un signal dérivé exploitable. A partir de 300 points le résultat est satisfaisant.

Cette application montre que la dérivation numérique est une opération délicate dans le cas de signaux provenant de capteurs, qui peuvent être fortement bruité et présenter des discontinuités.

Il est également possible de filtrer le signal avant de le dériver pour éliminer le bruit et les discontinuités.

IV. Résolution de l'équation $f(x)=0$

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentés dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Résolution $f(x)=0$**

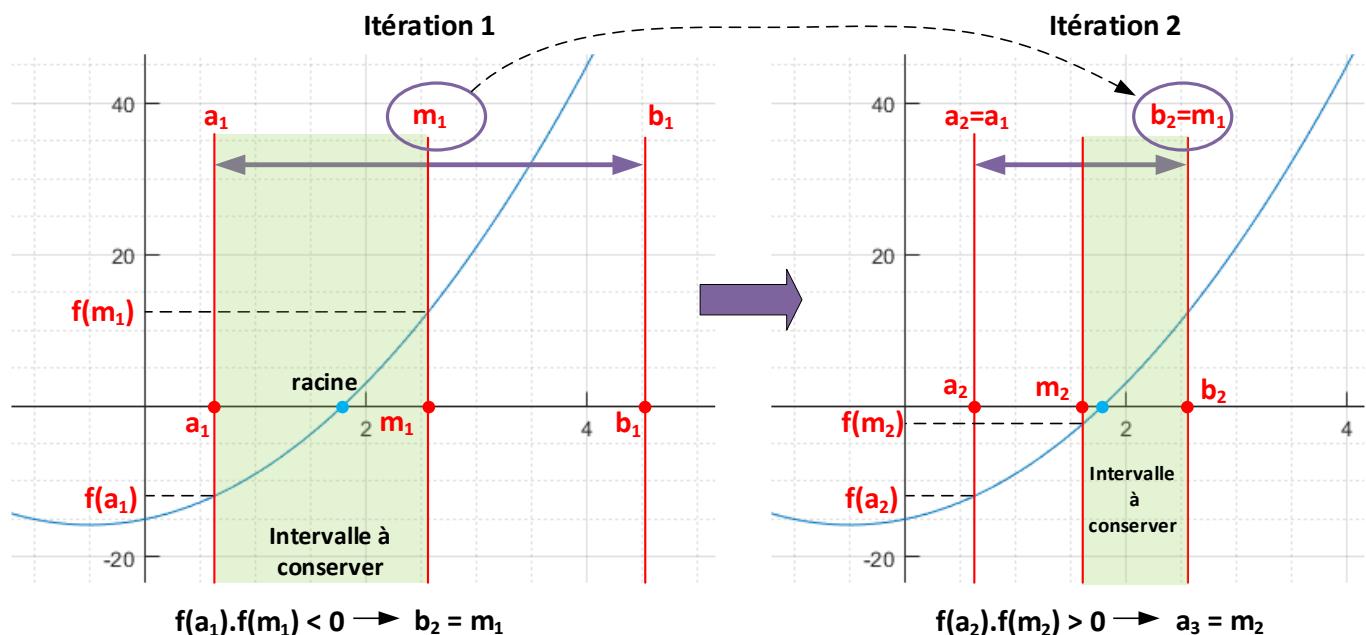
A. Dichotomie

Principe de la méthode : On part de deux valeurs a et b encadrant une solution unique de l'équation $f(x)=0$. A chaque itération, l'intervalle est divisé en deux parties égales en conservant l'intervalle qui contient la racine.

- Déterminer le point milieu de l'intervalle : $m = \frac{a+b}{2}$
- Evaluer $f(a) \cdot f(m)$
- Si $f(a) \cdot f(m) > 0$, l'intervalle $[a ; m]$ ne contient pas la racine. La racine se trouve dans l'intervalle $[m ; b]$ et a prend la valeur de m pour la prochaine itération.
- Si $f(a) \cdot f(m) < 0$, l'intervalle $[a ; m]$ contient la racine et b prend la valeur de m pour la prochaine itération.

Le processus s'arrête quand l'intervalle ($b-a$) devient inférieur à epsilon qui représente la précision que l'on souhaite obtenir dans l'évaluation de la solution. Afin de tester la condition de fin de notre algorithme, nous utiliserons une boucle « tant que ».

La Figure 675 illustre les 2 premières itérations de la méthode de recherche des solutions de l'équation $f(x)=0$ par la méthode de dichotomie. La solution de l'équation $f(x)=0$ est également appelée « racine » .



Afin de mettre en place le processus de recherche de solution par dichotomie, nous allons créer la fonction **fdichotomie(f,a,b,epsilon)** (Figure 676) qui calcule la solution de l'équation $f(x)=0$ dans l'intervalle $[a ; b]$ avec une précision **epsilon**.

Cette fonction prendra en arguments :

f: fonction dont on recherche la solution de l'équation $f(x)=0$

a : borne inférieure de l'intervalle d'étude

b : borne supérieure de l'intervalle d'étude

epsilon : précision souhaitée pour le processus de recherche de la solution

La fonction aura comme sortie :

sol: solution de l'équation $f(x)=0$ dans l'intervalle $[a ; b]$ avec une précision epsilon

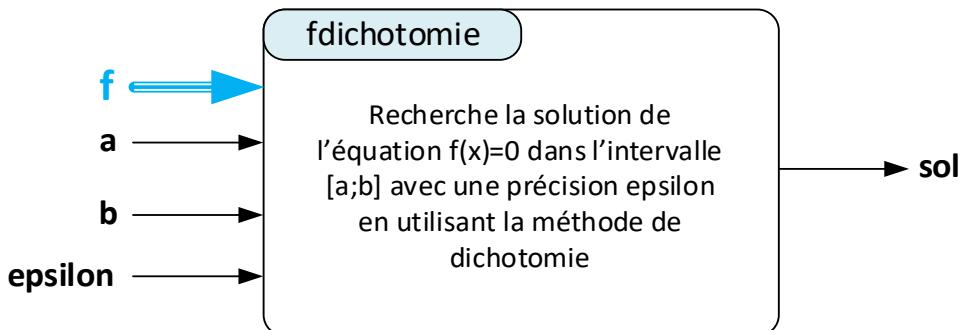


Figure 676 : fonction **fdichotomie(f,a,b,epsilon)**

La Figure 677 montre le codage de la fonction **fdichotomie(f,a,b,epsilon)**.

```
%% fdichotomie.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction renvoie la solution de l'équation f(x)=0 recherchée dans
% l'intervalle [a;b] avec une précision epsilon en utilisant la méthode de
% dichotomie

function sol = fdichotomie(f,a,b,epsilon)
while (b-a) > epsilon
    m = (a+b)/2;
    if f(a) * f(m) > 0
        a = m;
    else
        b = m;
    end
end
sol = (a+b)/2;
end
```

Figure 677 : codage de la fonction **fdichotomie(f,a,b,epsilon)**

Nous souhaitons résoudre l'équation $f(x)=0$ avec $f(x)=3x^2 + 3x - 15$. Cette fonction est stockée dans **f2.m** dont le codage est décrit sur la Figure 678.

```
%% f2.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f2.m contient l'expression mathématique de la fonction dont on recherche
% la solution de type f(x)=0

function y=f2(x)
y = 3.*x.^2+3.*x-15;
```

```
end
```

Figure 678: fonction f2.m

Afin de tester la fonction **fdichotomie**, il faut créer un script qui va appeler la fonction **fdichotomie** pour résoudre l'équation stockée dans la fonction **f2.m**.

Ouvrir le script **dichotomie.m**.

Ce script va dans un premier temps tracer la courbe représentative de la fonction et demander à l'utilisateur de lui donner un intervalle **[a ; b]** et une précision de recherche **epsilon** afin de démarrer la méthode de résolution. Le script donnera également le temps d'exécution de la fonction.

```
%% dichotomie.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Recherche par dichotomie de la solution de l'équation f(x)=0 dans
% l'intervalle [a;b] avec une précision epsilon
% La courbe y = f(x) est tracée
% L'utilisateur peut alors spécifier l'intervalle [a;b] de recherche de la
% solution et la précision epsilon souhaitée
% la fonction f(x) est stockée dans la fonction f2.m

%% Tracé de la fonction dont on cherche les racines par dichotomie
figure;
ezplot(@f2);
grid on; grid minor;

% affichage des axes sur le graphique
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

%% Spécification de l'intervalle [a;b] de recherche de la solution f(x)=0 et de
la précision souhaitée epsilon

disp('La solution de l''équation f(x)=0 est recherchée par dichotomie dans
l''intervalle [a;b]');
a = input('Saisir la valeur de a: ');
b = input('Saisir la valeur de b: ');
disp('La recherche de la solution se fera avec une précision epsilon: ');
eps = input('Spécifier la valeur de epsilon: ');

%% Appel de la fonction fdichotomie et affichage du résultat

% démarrage du compteur pour évaluer le temps d'exécution de la fonction
tic;

% appel de la fonction fdichotomie
sol = fdichotomie(@f2,a,b,eps);

% création de la variable t_dicho contenant le temps d'exécution de la
% fonction
t_dicho = toc;

% affichage de la solution et du temps nécessaire à l'exécution de la
% fonction
```

```

reponse = ['La solution de l''équation f(x)=0 dans l''intervalle
[',num2str(a),';',num2str(b),'] est: ', num2str(sol)];
disp(reponse);
temps = ['Le temps nécessaire au calcul est de ',num2str(t_dicho),' secondes'];
disp(temps);

```

Figure 679 : recherche de solution par dichotomie

Exécutez le script. La courbe représentative de la fonction apparaît (Figure 680).

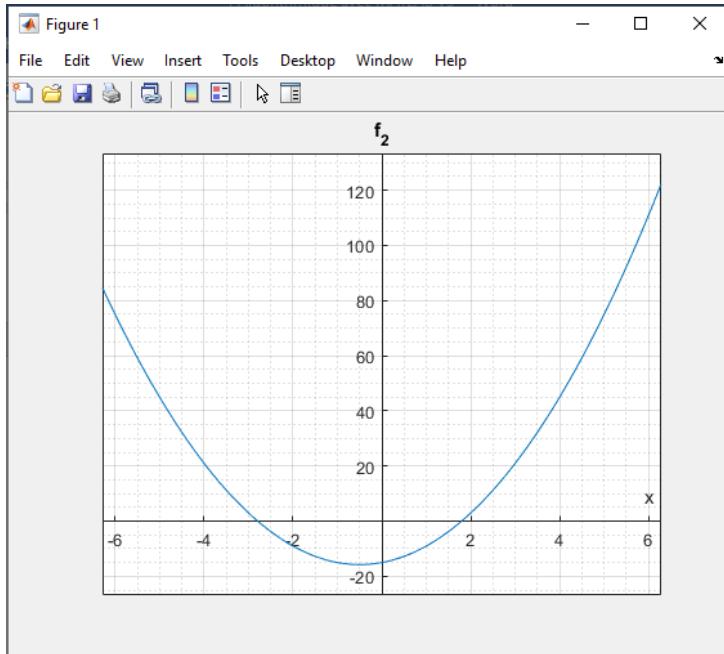


Figure 680 : courbe représentative de la fonction $f(x)$

Observez la courbe représentative de la fonction et indiquez l'intervalle **[a ; b]** dans lequel vous souhaitez rechercher les racines et la précision **epsilon** souhaitée (Figure 681).

La solution de l'équation $f(x)=0$ est recherchée par dichotomie dans l'intervalle **[a;b]**

Saisir la valeur de a: -4

Saisir la valeur de b: -2

La recherche de la solution se fera avec une précision epsilon:

Spécifier la valeur de epsilon: 0.001

La solution de l'équation $f(x)=0$ dans l'intervalle [-4;-2] est: -2.7915

Le temps nécessaire au calcul est de 0.0040617 secondes

Figure 681 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle[-4 ; 2]

Vous pouvez relancer le script pour chercher une solution dans un autre intervalle (Figure 682).

La solution de l'équation $f(x)=0$ est recherchée par dichotomie dans l'intervalle **[a;b]**

Saisir la valeur de a: 0

Saisir la valeur de b: 2

La recherche de la solution se fera avec une précision epsilon:

Spécifier la valeur de epsilon: 0.001

La solution de l'équation $f(x)=0$ dans l'intervalle [0,2] est: 1.7915

Le temps nécessaire au calcul est de 0.0012746 secondes

Figure 682 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle[0 ; 2]

B. Méthode de Newton

Principe de la méthode : Cette méthode consiste à approximer la courbe représentative de $f(x)$ par sa tangente en un point x_i . Le processus démarre avec le choix d'un point x_0 choisi proche de la solution recherchée. On cherche ensuite l'intersection de cette tangente avec l'axe des abscisses pour trouver un nouveau point x_1 . On cherche alors la tangente à la courbe représentative de $f(x)$ au point x_1 et on recherche de la même manière le point x_2 . Les x_i définissent alors une suite qui converge vers la solution de l'équation $f(x)=0$. La solution est alors trouvée par itérations successives avec une précision epsilon spécifiée.

La Figure 683 illustre les deux premières itérations de la méthode de Newton.

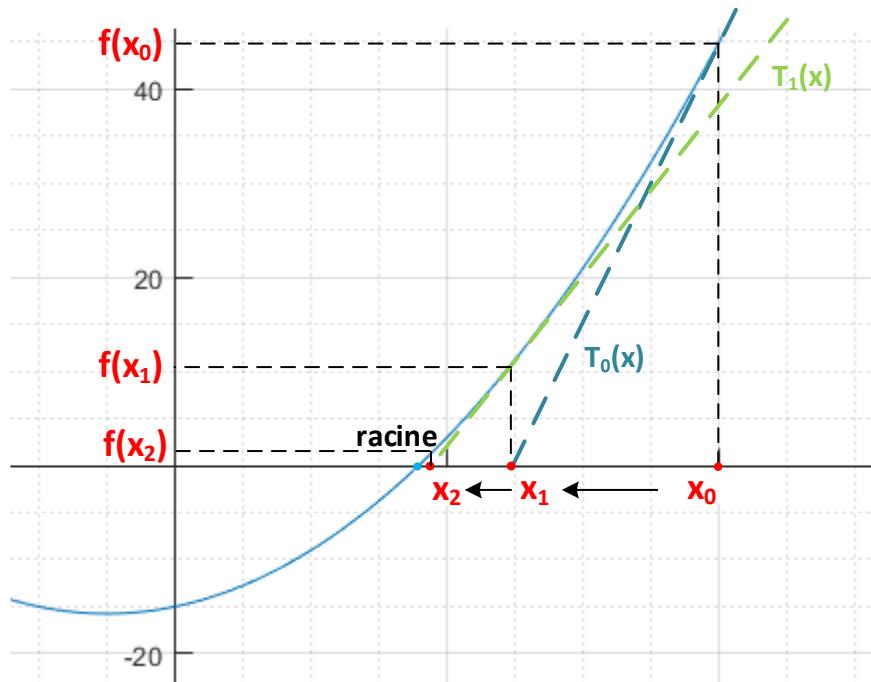


Figure 683 : illustration de la méthode de Newton

L'équation de la tangente au point x_0 à la courbe représentative de $f(x)$ est donnée par :

$$T_0(x) = f(x_0) + f'(x_0)(x - x_0)$$

- Première itération : On cherche l'intersection de la tangente T_0 avec l'axe des abscisses en cherchant la solution x_1 tel que $T_0(x_1) = 0$:

$$\begin{aligned} T_0(x_1) &= f(x_0) + f'(x_0)(x_1 - x_0) = 0 \\ \Rightarrow x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$

-
- $i^{\text{ème}}$ itérations : On cherche l'intersection de la tangente T_i avec l'axe des abscisses en cherchant x_{i+1} tel que $T_i(x_{i+1}) = 0$:

$$T_i(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) = 0$$

$$\Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (\text{relation de récurrence à programmer})$$

A chaque itération, il faudra évaluer les valeurs en x_i de la fonction f et de sa dérivée. Il faudra donc prévoir dans la fonction **fNewton.m** que nous allons définir un processus de dérivation numérique. Le processus de recherche de la solution s'arrêtera lorsque le critère de précision sera atteint, nous utiliserons une boucle « tant que ».

Afin de mettre en place le processus de recherche de solution par la méthode de Newton, nous allons créer la fonction **fNewton(f,x0,epsilon)** (Figure 684) qui calcule la solution de l'équation $f(x)=0$ autour de x_0 avec une précision **epsilon**.

Cette fonction prendra en arguments :

f: fonction dont on recherche la solution de l'équation $f(x)=0$

x0 : valeur autour de laquelle on recherche la solution

epsilon : précision souhaitée pour la recherche de la solution

La fonction aura comme sortie :

sol: solution de l'équation $f(x)=0$ dans l'intervalle $[a ; b]$ avec une précision epsilon

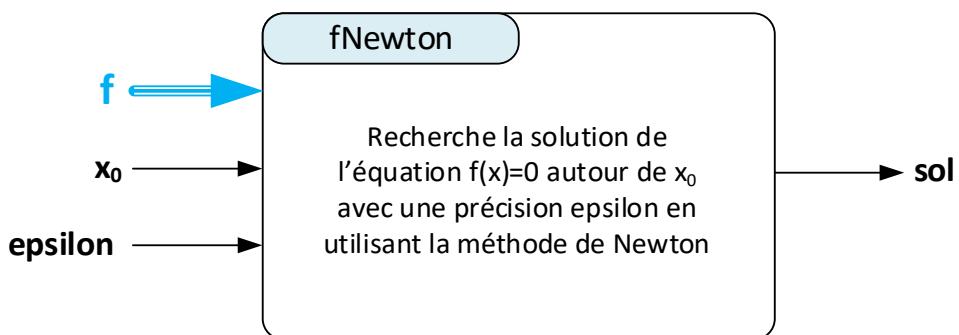


Figure 684 : fonction **fNewton(f,x0,epsilon)**

La Figure 685 montre le codage de la fonction **fNewton(f,x0,epsilon)**.

```

%% fNewton.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction renvoie la solution de l'équation f(x)=0 recherchée
% autour de la valeur x0 avec une précision epsilon en utilisant la
% méthode de Newton
% La fonction derive=df(f,x) permet d'obtenir la dérivée numérique de
% la fonction f avec un pas h

function [sol,N] = Newton(f,x0,epsilon)
sol = x0 - f(x0)/df(f,x0);

% tant que la précision n'est pas atteinte, on continue la méthode
while abs(f(sol)) > epsilon
    sol = sol - f(sol)/df(f,sol);
end
end

```

```
% fonction permettant de calculer la dérivée numérique de la fonction
% (différence finie progressive)
function derive = df(f,x)
h = 1e-5;
derive = (f(x+h/2)-f(x-h/2))/h;
end
```

Figure 685 : codage de la fonction fNewton(f,x0,epsilon)

Nous souhaitons résoudre l'équation $f(x)=0$ avec $f(x)=3x^2+3x-15$. Cette fonction est stockée dans **f2.m** dont le codage est décrit sur la Figure 686.

```
%% f2.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f2.m contient l'expression mathématique de la fonction dont on recherche
% la solution de type f(x)=0

function y=f2(x)
y = 3.*x.^2+3.*x-15;
end
```

Figure 686: fonction f2.m

Afin de tester la fonction **fNewton**, il faut créer un script qui va appeler la fonction **fNewton** pour résoudre l'équation stockée dans la fonction **f2.m**.

Ouvrir le script **Newton.m**.

Ce script va dans un premier temps tracer la courbe représentative de la fonction et demander à l'utilisateur de lui donner une valeur de **x0** et une précision de recherche **epsilon** afin de démarrer la méthode de résolution. Le script donnera également le temps d'exécution de la fonction.

```
%% Newton.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Recherche par la méthode de Newton de la solution de l'équation f(x)=0
% autour de la valeur x0 avec une précision epsilon
% La courbe y = f(x) est tracée
% L'utilisateur peut alors spécifier la valeur de x0 permettant de démarrer
% le processus de recherche de solution ainsi que la précision epsilon
% souhaitée
%% Tracé de la fonction dont on cherche les racines par la méthode de Newton
figure;
ezplot(@f2);
grid on; grid minor;

% affichage des axes sur le graphique
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

%% Spécification de la valeur de x0 permettant de démarrer le processus de
% recherche de la solution f(x)=0 et de la précision souhaitée epsilon

disp('La solution de l''équation f(x)=0 est recherchée par la méthode de Newton
autour de la valeur x0');
```

```

x0 = input('Saisir la valeur de x0: ');
disp('La recherche de la solution se fera avec une précision epsilon: ');
eps = input('Spécifier la valeur de epsilon: ');

%% Appel de la fonction fNewton et affichage du résultat
tic;
sol = fNewton(@f2,x0,eps);
t_dicho = toc;
reponse = ['La solution de l''équation f(x)=0  autour de x=',num2str(x0), ' est: ', num2str(sol)];
disp(reponse);
temps = ['Le temps nécessaire au calcul est de ',num2str(t_dicho), ' secondes'];
disp(temps);

```

Figure 687 : recherche de solution par la méthode de Newton

Exécutez le script. La courbe représentative de la fonction apparaît (Figure 688).

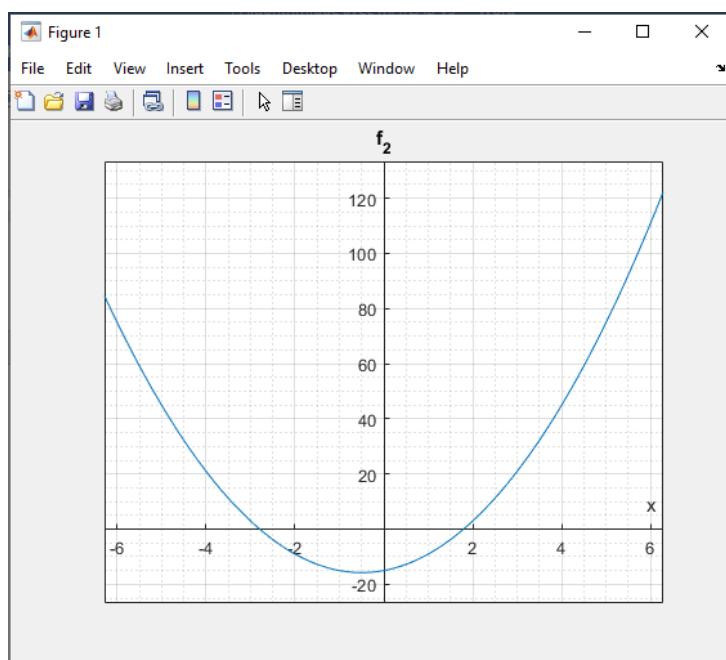


Figure 688 : courbe représentative de la fonction $f(x)$

Observez la courbe représentative de la fonction et indiquez la valeur de x_0 et la précision **epsilon** souhaitée (Figure 689).

La solution de l'équation $f(x)=0$ est recherchée par la méthode de Newton autour de la valeur x_0
Saisir la valeur de x_0 : 3

La recherche de la solution se fera avec une précision epsilon:

Spécifier la valeur de epsilon: 0.001

La solution de l'équation $f(x)=0$ autour de $x=3$ est: 1.7913

Le temps nécessaire au calcul est de 0.00063 secondes

Figure 689 : exécution du script Newton.m pour rechercher la solution autour de $x=3$

Vous pouvez relancer le script pour chercher une solution autour d'une autre valeur de x_0 (Figure 690).

La solution de l'équation $f(x)=0$ est recherchée par la méthode de Newton autour de la valeur x_0
Saisir la valeur de x_0 : -2

La recherche de la solution se fera avec une précision epsilon:

Spécifier la valeur de epsilon: 0.001

La solution de l'équation $f(x)=0$ autour de $x=-2$ est: -2.7913
Le temps nécessaire au calcul est de 0.0014403 secondes

Figure 690 : exécution du script Newton.m pour rechercher la solution autour de $x=-2$

V. Intégration numérique

A. Principe

Intégrer numériquement une fonction sur un intervalle $[a ; b]$ consiste à évaluer numériquement l'aire sous la courbe représentative de la fonction.

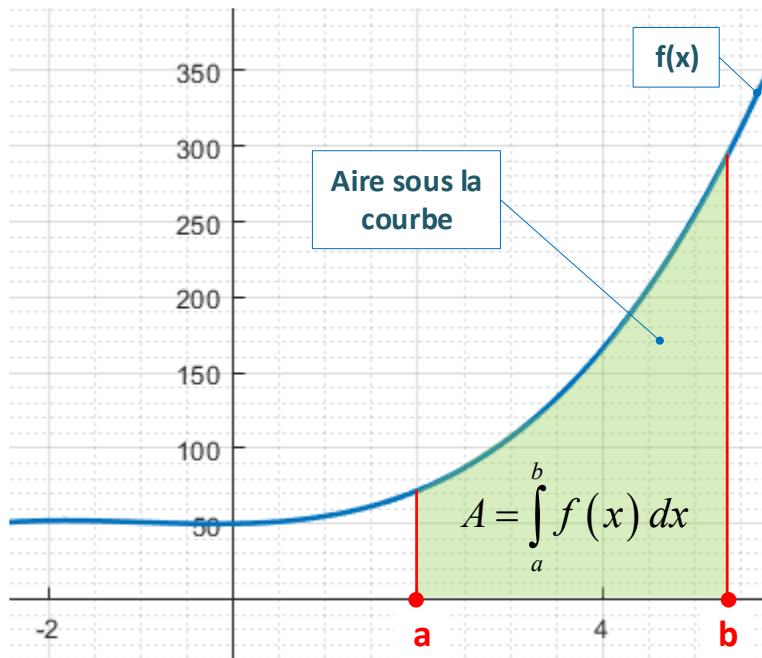


Figure 691 : calcul de l'intégrale d'une fonction sur un intervalle $[a ; b]$

Pour réaliser une intégration numérique et calculer cette aire A , il faut dans un premier temps discréteriser l'intervalle $[a ; b]$ en N intervalles de longueur identique en choisissant un pas d'intégration \mathbf{h} . Plus ce pas sera petit et plus le calcul numérique sera précis.

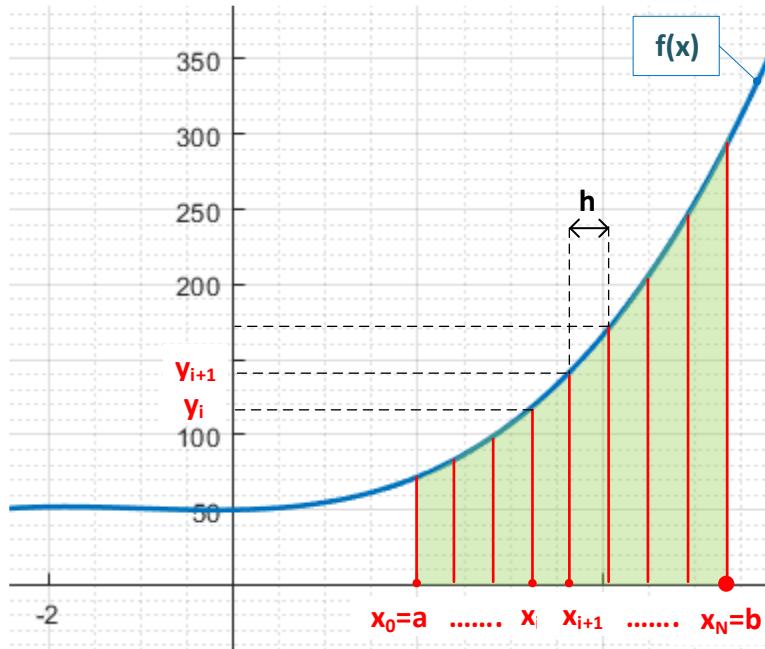


Figure 692 : discréétisation du domaine d'intégration

La fonction à intégrer sera alors remplacée par son évaluation numérique y_i en chaque point x_i de l'intervalle $[a ; b]$.

Les points x_i sont définis tels que :

$$x_i = a + i.h$$

Les valeurs de y_i sont calculées à partir des x_i :

$$y_i = f(x_i)$$

Il faut ensuite choisir une méthode d'interpolation de la fonction $f(x)$ entre deux valeurs de y_i . Nous aborderons deux cas d'interpolations :

- Interpolation de degré 0 : La fonction gardera une valeur constante à l'intérieur d'un intervalle discréétisé. On parlera de méthode des rectangles (Figure 693). Le calcul numérique de l'intégrale se ramène à additionner l'aire de chaque rectangle élémentaire.

$$I_{rec} = \sum_{i=0}^{i=N-1} h.y_i \simeq \int_a^b f(x)dx$$

- Interpolation de degré 1 : La fonction sera remplacée par une interpolation linéaire à l'intérieur d'un intervalle discréétisé. On parlera de méthode des trapèzes (Figure 694). Le calcul numérique de l'intégrale se ramène à additionner l'aire de chaque trapèze élémentaire.

$$I_{trap} = \sum_{i=0}^{i=N-1} h \cdot \frac{y_i + y_{i+1}}{2} \simeq \int_a^b f(x)dx$$

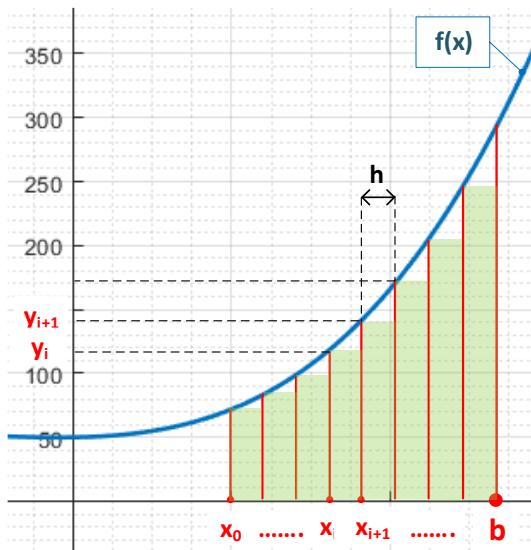


Figure 693 : méthode des rectangles

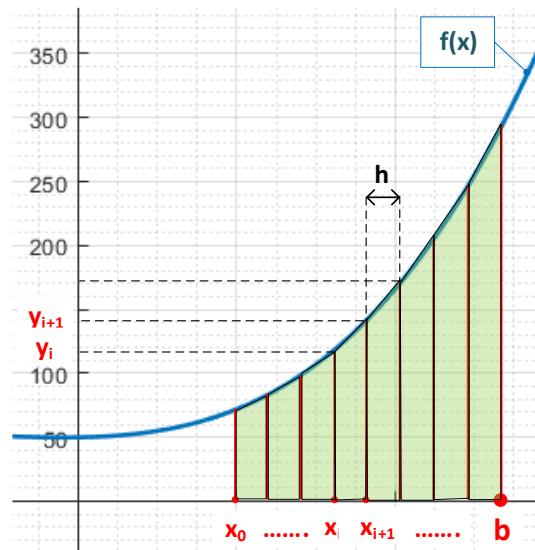


Figure 694 : méthode des trapèzes

Il apparaît déjà à ce stade que plus le polynôme d'interpolation est de degré élevé et plus la méthode d'intégration numérique sera précise. La méthode des trapèzes sera plus précise que la méthode des rectangles.

B. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Intégration_numérique**.

L'expression mathématique de la fonction à intégrer sera définie dans la fonction **f3.m** (Figure 695). Cette expression peut être modifiée pour tester toutes les fonctions souhaitées. Il faudra juste vérifier que lors de l'appel de la fonction l'intervalle d'étude $[a ; b]$ appartienne bien au domaine de définition de la fonction. On choisira $f_3(m) = \cos(x)$.

```
%% f3.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f3.m contient l'expression mathématique de la fonction à intégrer

%%
function y = f3(x)
y = cos(x);
end
```

Figure 695 : codage de la fonction f3.m qui contient l'expression de la fonction à intégrer

1. Méthode des rectangles

Pour calculer l'intégrale d'une fonction par la méthode des rectangles, nous allons créer la fonction **integration_rec(a,b,nb_points,f)** (Figure 696).

Cette fonction prendra en arguments :

a : borne inférieure de l'intervalle d'étude

b : borne supérieur de l'intervalle d'étude
nb_points : nombre de points à calculer
f : fonction à intégrer

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points où la valeur de l'intégrale est calculée
df : vecteur contenant les valeurs de la fonction intégrée numériquement

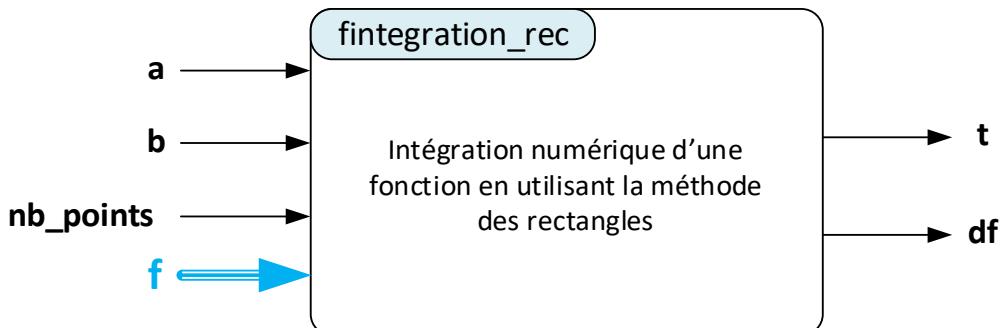


Figure 696 : fonction fintégration_rec (a,b,nb_point,f)

La Figure 697 montre le codage de la fonction **fintegration_rec** (a,b,nb_points,f).

```

%% [t,df] = fintegration_rec(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des rectangles
%%
function [t,int_f] = fintegration_rec(a,b,nb_points,f)
% calcul du pas h
h = (b-a)/nb_points;

% définition des conditions initiale de l'intégration (par défaut CI=0)
int_f(1) = 0;
t(1) = a;
% La boucle va ajouter l'aire des rectangles pour chaque intervalle et
% créer le vecteur contenant les temps et le vecteur contenant les valeurs
% de l'intégrale de la fonction prise en argument

for k = [2:1:nb_points]
    % construction du vecteur des abscisses
    t(k) = t(k-1) + h;
    % construction du vecteur contenant les valeurs de l'intégrale
    % de la fonction
    int_f(k) = int_f(k-1) + f(t(k)) * h;
end

```

Figure 697 : codage de la fonction fintegration_rec (a,b,nb_point,f)

2. Méthode des trapèzes

Pour calculer l'intégrale d'une fonction par la méthode des trapèzes, nous allons créer la fonction **fintegration_trap** (a,b,nb_points,f) (Figure 698).

Cette fonction prendra en arguments :

a : borne inférieure de l'intervalle d'étude
b : borne supérieure de l'intervalle d'étude

nb_points : nombre de points à calculer
f : fonction à intégrer

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points où la valeur de l'intégrale est calculée
df : vecteur contenant les valeurs de la fonction intégrée numériquement

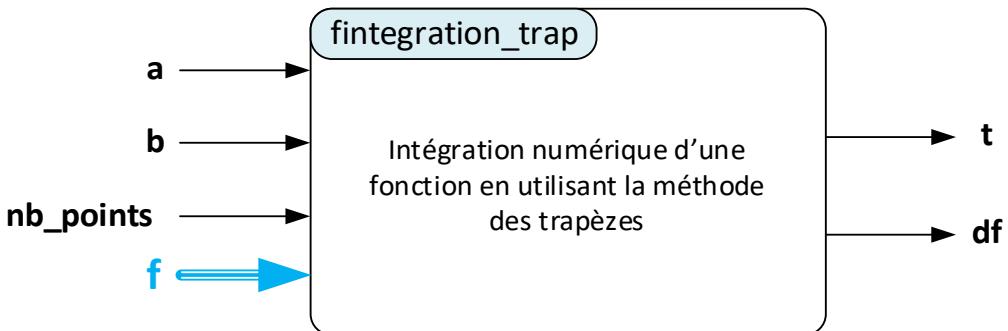


Figure 698 : fonction fintegration_trap (a,b,nb_point,f)

La Figure 699 montre le codage de la fonction **fintegration_trap** (a,b,nb_points,f).

```

%% [t,df] = fintegration_trap(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des trapèzes
%%
function [t,int_f] = fintegration_trap(a,b,nb_points,f)
% calcul du pas h
h = (b-a)/nb_points;

% définition des conditions initiales de l'intégration (par défaut CI=0)
int_f(1) = 0;
t(1) = a;

% La boucle va ajouter l'aire des trapèzes pour chaque intervalle et
% créer le vecteur contenant les temps et le vecteur contenant les valeurs
% de l'intégrale de la fonction prise en argument
for k = [2:1:nb_points]
    % construction du vecteur des abscisses
    t(k) = t(k-1) + h;
    % construction du vecteur contenant les valeurs de l'intégrale
    % de la fonction
    int_f(k) = int_f(k-1) + (f3(t(k))+f3(t(k-1))) * h/2;
end
  
```

Figure 699 : codage de la fonction fintegration_trap (a,b,nb_point,f)

3. Calcul exact de l'intégrale d'une fonction en utilisant le calcul symbolique

Afin de pouvoir évaluer l'erreur induite par les différentes méthodes d'intégration numérique, nous allons créer une fonction **fintegration_ana_symbolique** (a,b,nb_points,f) qui calcule l'intégrale analytique exacte d'une fonction en calcul symbolique (Figure 700). Cette intégration constituera la référence pour calculer l'erreur.

Cette fonction prendra en arguments :

a : borne inférieure de l'intervalle d'étude

b : borne supérieur de l'intervalle d'étude
nb_points : nombre de points à calculer
f : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points où l'intégrale est calculée
df : vecteur contenant les valeurs de l'intégrale de la fonction calculée analytiquement

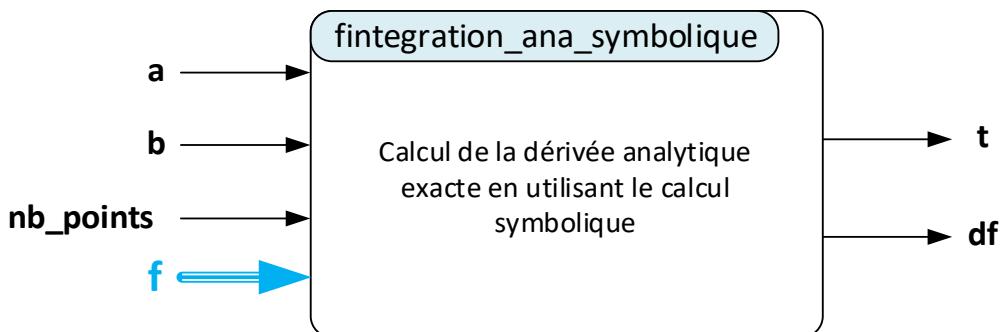


Figure 700 : fonction fintegration_ana_symbolique (a,b,nb_point,f)

La Figure 701 montre le codage de la fonction **fintegration_ana_symbolique** (a,b,nb_points,f).

```

%% [t,df] = fintegration_anasymbolique(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul analytique exacte de l'intégrale d'une fonction en utilisant
% le calcul symbolique

function [t,int_f]=fintegration_ana_symbolique(a,b,nb_points,f)
h = (b-a)/nb_points;
t = [a:h:b];
% x est déclarée comme variable symbolique
syms x;

% création de la fonction symbolique f_symbolique à partir de f(x)
f_symbolique(x) = f(x);

% intégration de la fonction f_symbolique pour créer la fonction dérivée
% symbolique
int_f_symbolique(x) = int(f_symbolique,x);

% calcul des valeurs du vecteur int_f qui contient les valeurs de
% l'intégrale de la fonction calculée analytiquement
int_f = int_f_symbolique(t);

% conversion de la fonction symbolique en double précision
int_f = double(int_f);

int_f(end) = [];
t(end) = [];
end

```

Figure 701 : codage de la fonction fintegration_ana_symbolique (a,b,nb_point,f)

C. Exploitation de l'algorithme

1. Comparaison des différentes méthodes d'intégration numérique

Afin d'évaluer les écarts entre les différentes méthodes d'intégration numérique et l'intégrale de la fonction calculée analytiquement, nous allons créer un script qui va tracer les intégrales calculées avec les 3 méthodes dans la même fenêtre graphique, puis quantifier et tracer l'amplitude comparée de ces écarts.

Ouvrir le script **num_integration_plot_comparaison.m**.

Le script est structuré en trois sections (Figure 702) :

- Définition de l'intervalle d'étude de la fonction **[a ;b]** et du nombre de points **nb_points** à calculer pour tracer les courbes
- Appel des différentes fonctions et tracé des différentes courbes sur la même fenêtre graphique
- Evaluation des écarts entre l'intégrale analytique et les deux méthodes d'intégration numérique et tracé des écarts sur la même fenêtre graphique

```
%% num_integration_comparaison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%Calcul numérique de l'intégrale d'une fonction en utilisant différentes
% méthodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% et le nombre de points nb_points pris en compte pour le calcul
% La fonction à intégrer est stockée dans la fonction f3.m

%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% définition du nombre de point à utiliser pour le calcul
nb_points = 20;
close all;
%% Calcul et tracé de l'intégrale de la fonction en utilisant différente méthodes

% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[2 8 18 12]);

grid on; grid minor; hold on;

% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t_ana_symbolique,int_f_ana_symbolique] =
fintegration_ana_symbolique(a,b,nb_points,@f3);
plot(t_ana_symbolique,int_f_ana_symbolique,'red','LineWidth',2);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
[t_rec,int_f_rec] = fintegration_rec(a,b,nb_points,@f3);
```

```

plot(t_rec,int_f_rec,'cyan','LineWidth',2);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t_trap,int_f_trap] = fintegration_trap(a,b,nb_points,@f3);
plot(t_trap,int_f_trap,'blue','LineWidth',2);
% titre et légende

title({'Comparaison des méthodes d''intégration numérique';...
        'de la fonction f3'});

legend('Intégrale calculée analytiquement','Intégrale calculée avec la méthode
des rectangles',...
        'Intégrale calculée avec la méthode des trapèzes');

%% %% Calcul des différents écarts en prenant comme référence le calcul
% de l'intégrale exacte déterminée en calcul symbolique

error_rec = int_f_ana_symbolique - int_f_rec;
error_trap = int_f_ana_symbolique - int_f_trap;

% création du vecteur des abscisses

t = [a:(b-a)/nb_points:b];
t(end) = [];

% création et positionnement de la fenêtre graphique

figure('Units','centimeters','Position',[22 8 18 12]);
hold on; grid on; grid minor;

% tracé des différents écarts

plot(t,error_rec,'green','LineWidth',2)
plot(t,error_trap,'blue','LineWidth',2)

% titre et légende

title({'Erreurs comparées des différentes méthodes d''intégration numérique';...
        'par rapport à l''intégrale calculée analytiquement'});
legend('Méthode des rectangles','Méthode des trapèzes')

xlim([a b]);

```

Figure 702 : comparaison des différentes méthodes de dérivation numérique

Dans un premier temps le script sera exécuté sur l'intervalle $[0 ; 10]$ avec 20 points calculés. La fonction définie dans **f3.m** est la fonction $f(x) = \cos(x)$. Cette fonction peut être modifiée par l'utilisateur.

L'exécution du script permet de visualiser les résultats du calcul numérique de l'intégrale de la fonction avec les différentes méthodes de calcul (Figure 703, Figure 704 et Figure 705).

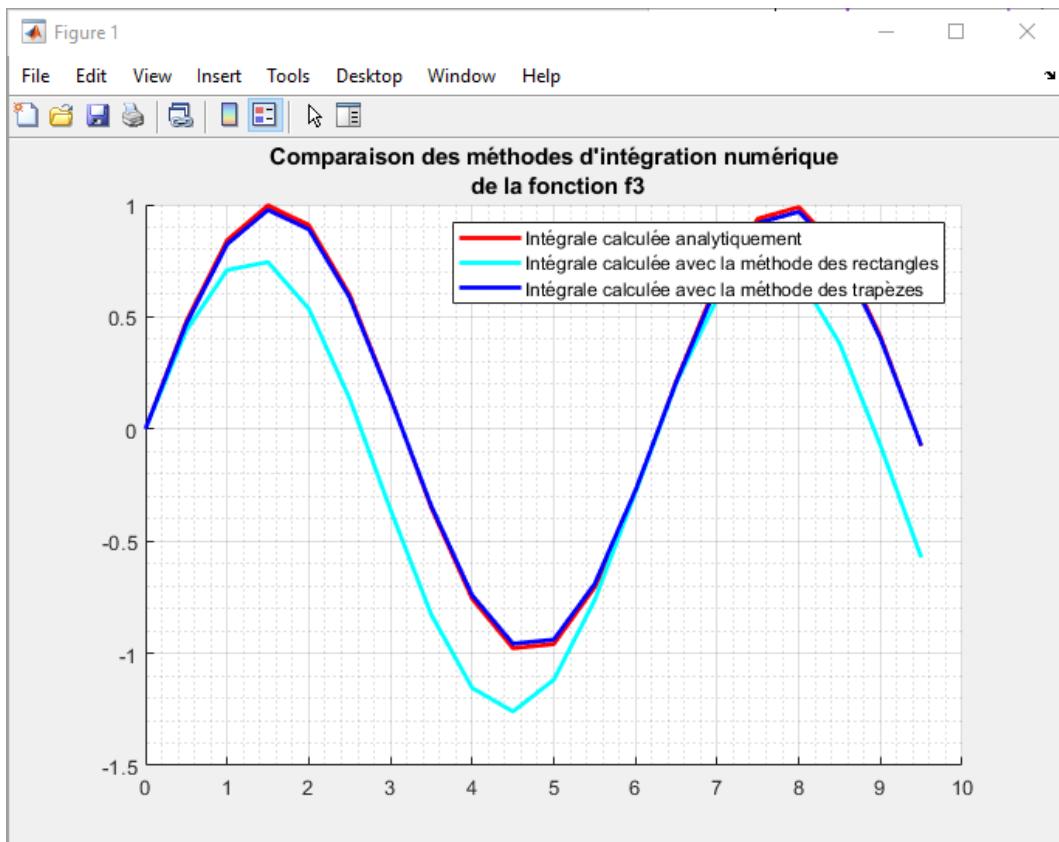


Figure 703 : comparaison des différentes méthodes d'intégration numérique

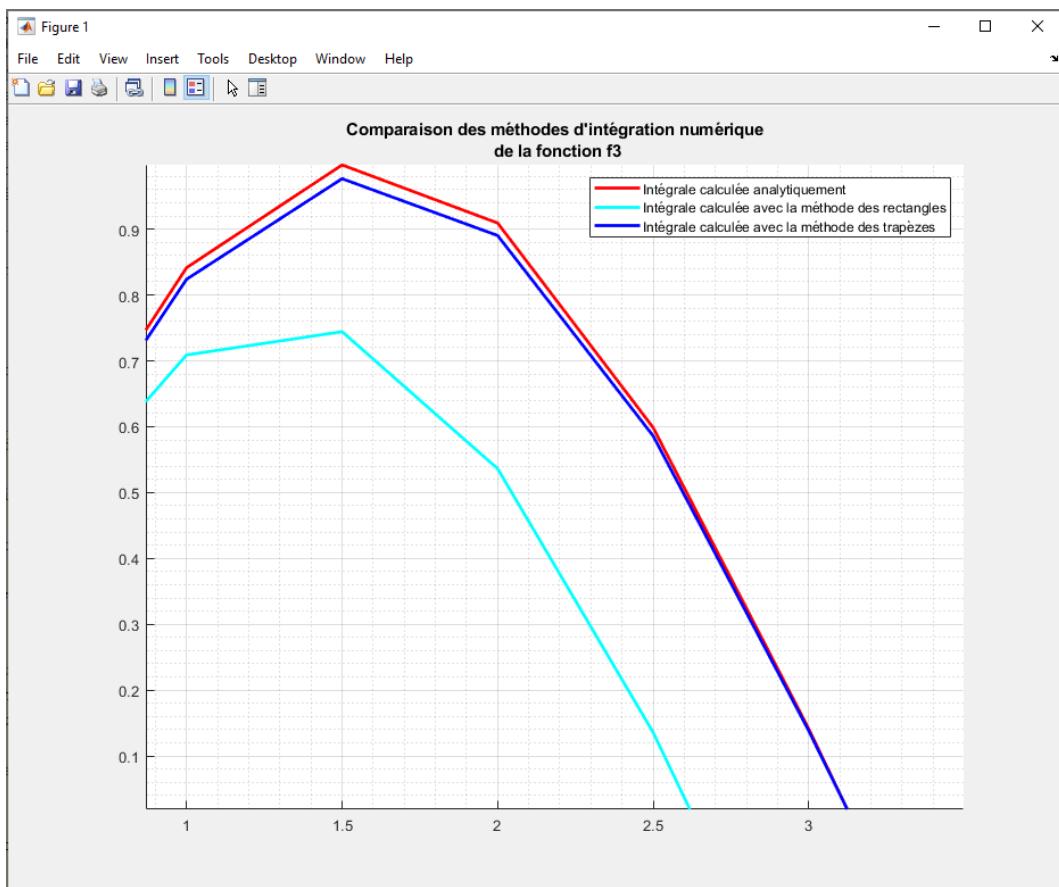


Figure 704 : zoom sur la courbe

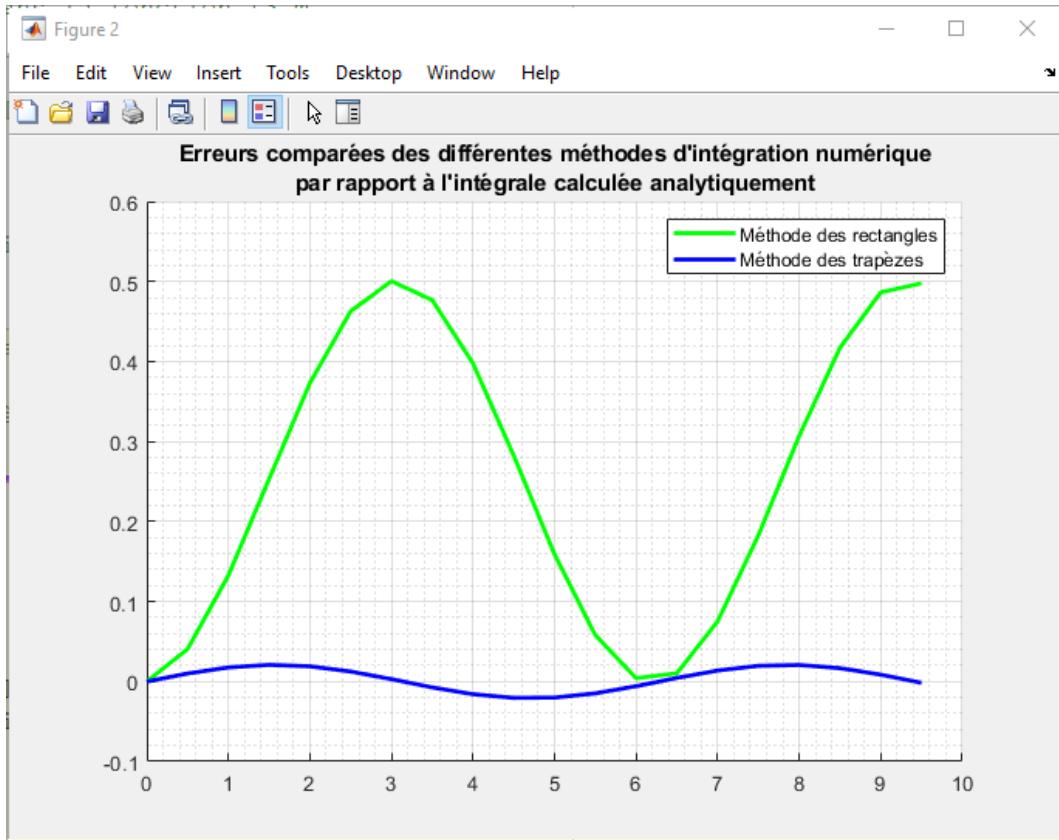


Figure 705 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique

L'analyse des différents courbes nous permet de voir de manière évidente que la méthode des trapèzes est plus précise que la méthode des rectangles.

2. Quantification et visualisation de l'erreur induite par les différentes méthodes

Il serait utile d'avoir une idée de l'évolution de cette erreur en fonction du nombre de point calculés et donc en fonction du pas $h = \frac{b-a}{nb_points}$.

Pour cela ouvrir le script **num_integration_plot_comparaison_subplot.m**.

Ce script permet de tracer 4 courbes avec un nombre de points calculés différents pour chaque courbe et de voir ainsi l'évolution de l'erreur en fonction du nombre de points et du pas h choisi pour le calcul. Les 4 courbes correspondent à 4 valeurs du nombre de points calculés choisies équitablement réparties entre les variables **nb_points_min** et **nb_points_max** définies en début de script.

Le script est structuré en deux sections (Figure 706).

- Définition de l'intervalle d'étude de la fonction **[a ; b]** et du nombre de points **nb_points_min** et **nb_points_max**
- Appel des différentes fonctions et tracé des différentes courbes sur 4 fenêtres graphiques différentes

```

%% num_intégration_comparaison_subplot
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%
% Calcul numérique de l'intégrale d'une fonction en utilisant différentes
% méthodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% Représentation de 4 courbes correspondant à 4 valeurs du nombres de
% points calculés choisies équitablement réparties entre nb_points_min et
% nb_points_max
% La fonction à intégrer est stockée dans la fonction f3.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% spécification des nombres de points mini et maxi
nb_points_min = 10;
nb_points_max = 50;
pas = round((nb_points_max - nb_points_min) / 3);
%%
fig = figure('Units','centimeters','Position',[1 1 35 25]);
set(fig,'Color','white');
sgtitle({'Comparaison des différentes méthodes d''intégration numérique',...
    'de la fonction f3 en fonction du nombre de points calculés'});

% initialisation du compteur de boucle
k = 1;

for nb_points = nb_points_min : pas : nb_points_max;

subplot(2,2,k); hold on; grid on; grid minor;

% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t_ana_symbolique,int_f_ana_symbolique] =
fintegration_ana_symbolique(a,b,nb_points,@f3);
plot(t_ana_symbolique,int_f_ana_symbolique,'red','LineWidth',2);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
[t_rec,int_f_rec] = fintegration_rec(a,b,nb_points,@f3);
plot(t_rec,int_f_rec,'cyan','LineWidth',2);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t_trap,int_f_trap] = fintegration_trap(a,b,nb_points,@f3);
plot(t_trap,int_f_trap,'blue','LineWidth',2);

% légende('dérivée analytique','différence finie centrée',...
%     'différence finie progressive','différence fine rétrograde');

title('Nombre de points calculés ' + string(nb_points))
k = k + 1;
end

```

```
% spécification de la légende générale du graphique
leg = legend('Intégration analytique', 'Méthode des rectangles', ...
    'Méthode des trapèzes');

% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg, 'Position', newPosition, 'Units', newUnits);
```

Figure 706 : script permettant de visualiser l'influence du pas

L'exécution du script permet de visualiser l'influence du pas sur l'allure des courbes (Figure 656).

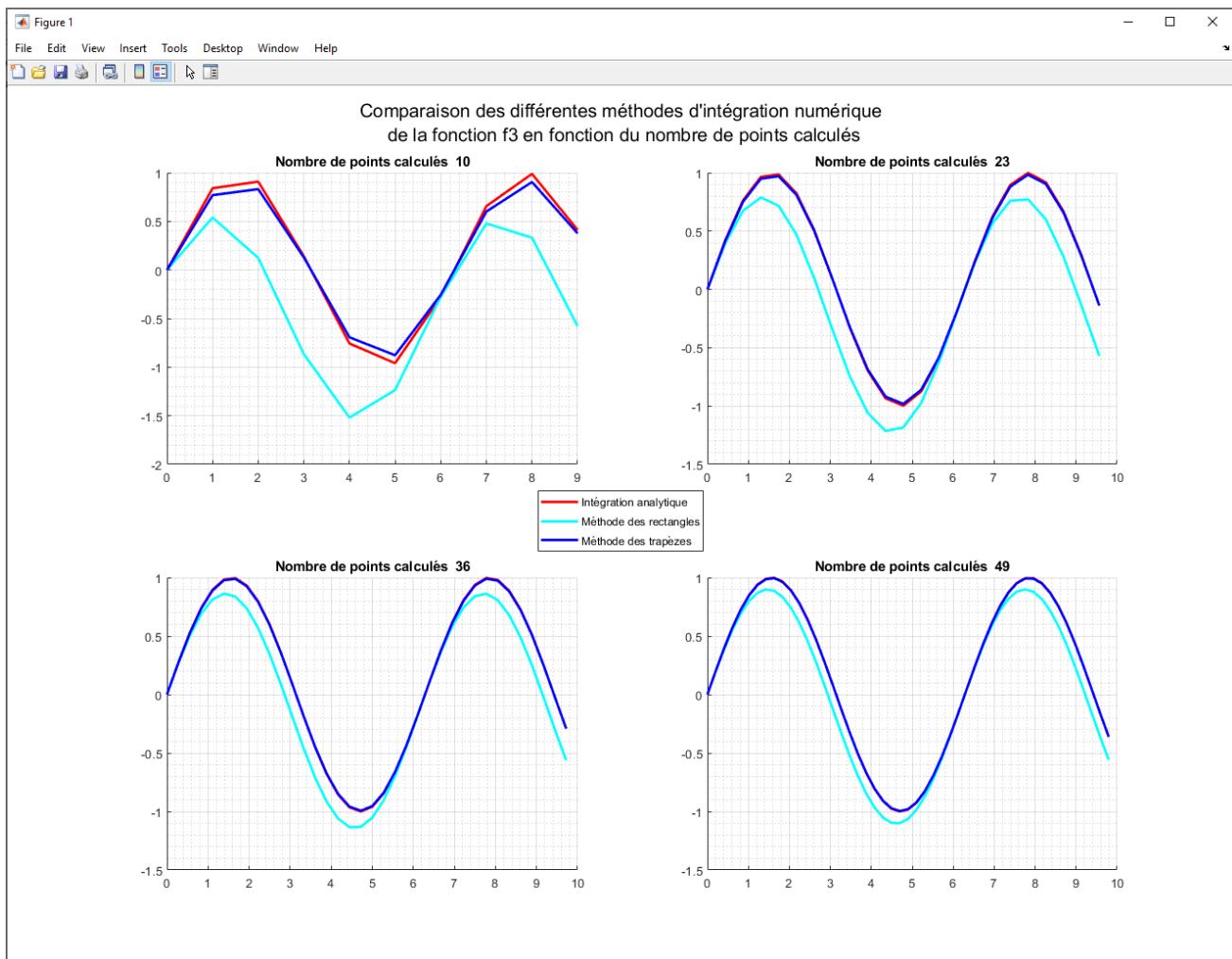


Figure 707 : influence du pas sur l'allure des courbes

L'analyse de ces courbes nous confirme que l'erreur va diminuer avec le pas mais ne nous permet pas de connaître la loi d'évolution de l'erreur en fonction du pas h . Lors de l'utilisation d'un processus d'intégration numérique, il est important de déterminer cette loi d'évolution.

3. Détermination de la loi d'évolution de l'erreur en fonction du pas

Pour connaître la loi d'évolution de cette erreur, il suffit de se placer en un point x_0 arbitraire de l'intervalle et de calculer l'erreur pour différentes valeurs du pas h .

L'erreur en fonction du pas h se définit de la manière suivante :

- Pour la méthode des rectangles

$$f_{err_rec}(h) = \int_a^b f(x) dx - \sum_{i=0}^{i=N-1} h.y_i$$

- Pour la méthode des trapèzes :

$$f_{err_trap}(h) = \int_a^b f(x) dx - \sum_{i=0}^{i=N-1} h \cdot \frac{y_i + y_{i+1}}{2}$$

Nous allons construire un script qui se placera arbitrairement au centre de notre intervalle d'étude et qui va tracer automatiquement l'erreur en fonction du pas h. Le script contient une boucle qui calculera les différentes fonctions erreurs pour un nombre de points allant de 100 à 1000 avec un incrément de 100.

Le pas h est défini par $h = \frac{b-a}{nb_points}$.

La boucle permettra donc de calculer 10 valeurs de l'erreur au point x_0 pour les deux méthodes d'intégration numérique. Le script permet ensuite d'interpoler les points calculés avec une fonction polynomiale adaptée pour trouver l'ordre de variation des fonctions erreurs.

Ouvrir le script **num_integration_error.m** (Figure 708).

```
%% num_integration_error
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la forme mathématique de l'erreur effectuée lors des
% opérations d'intégration numérique pour différentes méthodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% On se place arbitrairement au centre de l'intervalle d'étude de la
% fonction en un point  $x_0$  afin d'évaluer l'erreur en ce point pour
% différentes valeurs de h et du nombre de points de calcul
% on utilise ensuite une interpolation polynomiale pour déterminer la forme
% mathématique de l'erreur en fonction de h
% La fonction à intégrer est stockée dans la fonction f3.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;

% initialisation du compteur qui servira d'indice
k = 1;

%% On définit ici une boucle qui permettra de calculer l'erreur pour un
% échantillon de points allant de 100 points jusqu'à 1000 points par pas
% de 100 pour toutes les méthodes de dérivation

for nb_points = 100:100:1000

% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t_ana_symbolique,int_f_ana_symbolique] =
fintegration_ana_symbolique(a,b,nb_points,@f3);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
```

```

[t_rec,int_f_rec] = fintegration_rec(a,b,nb_points,@f3);

% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t_trap,int_f_trap] = fintegration_trap(a,b,nb_points,@f3);

% calcul des erreurs au point x0, milieu de l'intervalle d'étude en prenant
% comme référence la dérivée exacte déterminée en calcul symbolique
x0 = round(nb_points/2);
error_f_rec_h(k) = int_f_ana_symbolique(x0) - int_f_rec(x0);
error_f_trap_h(k) = int_f_ana_symbolique(x0) - int_f_trap(x0);
% détermination de la valeur du pas h(k) pour l'itération en cours
h(k) = (b-a)/nb_points;

% incrémentation de k
k = k + 1;
end

%% Tracé et interpolation de l'erreur pour la méthode des rectangles
% Création d'une nouvelle figure pour les graphiques
figure('Units','centimeters','Position',[2 8 18 12]);
hold all;grid on; grid minor;
plot(h,error_f_rec_h,'sm','MarkerSize',10,'LineWidth',2);

% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_rec = polyfit (h,error_f_rec_h,1);

% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_rec = polyval(coeff_poly_inter_rec,t);
plot(t,poly_inter_rec,'LineWidth',2);

% Ecriture de la légende
legend('Méthode des rectangles',...
    string(coeff_poly_inter_rec(2)) + ' h + ' ...
    + string(coeff_poly_inter_rec(1)), 'FontSize',12);

% Ecriture du titre et des étiquettes des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour la méthode des rectangles'});

xlabel('pas h');
ylabel('erreur');

%% Tracé et interpolation de l'erreur pour la méthode des trapèzes
figure('Units','centimeters','Position',[22 8 18 12]);
hold all;grid on; grid minor;
plot(h,error_f_trap_h,'sb','MarkerSize',10,'LineWidth',2)

% Détermination du polynôme d'interpolation pour la différence finie
% progressive

% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_trap = polyfit (h,error_f_trap_h,2);

% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_trap = polyval(coeff_poly_inter_trap,t);
plot(t,poly_inter_trap,'LineWidth',2);

% écriture de la légende

```

```

legend('Méthode des trapèzes',...
string(coeff_poly_inter_trap(3)) + ' h^2 + ' ...
+ string(coeff_poly_inter_trap(2)) + ' h + ' ...
+ string(coeff_poly_inter_trap(1)),...
'Location','southwest','FontSize',12);

% titre et étiquette des axes
title({'Variation de l''erreur en fonction du pas h' ;...
'pour la méthode des trapèzes'});

xlabel('pas h');
ylabel('erreur');

```

Figure 708 : script permettant de connaître l'ordre de variation des fonctions erreurs

Exécutez le script et analyser les courbes obtenues sur la Figure 709 et sur la Figure 710.

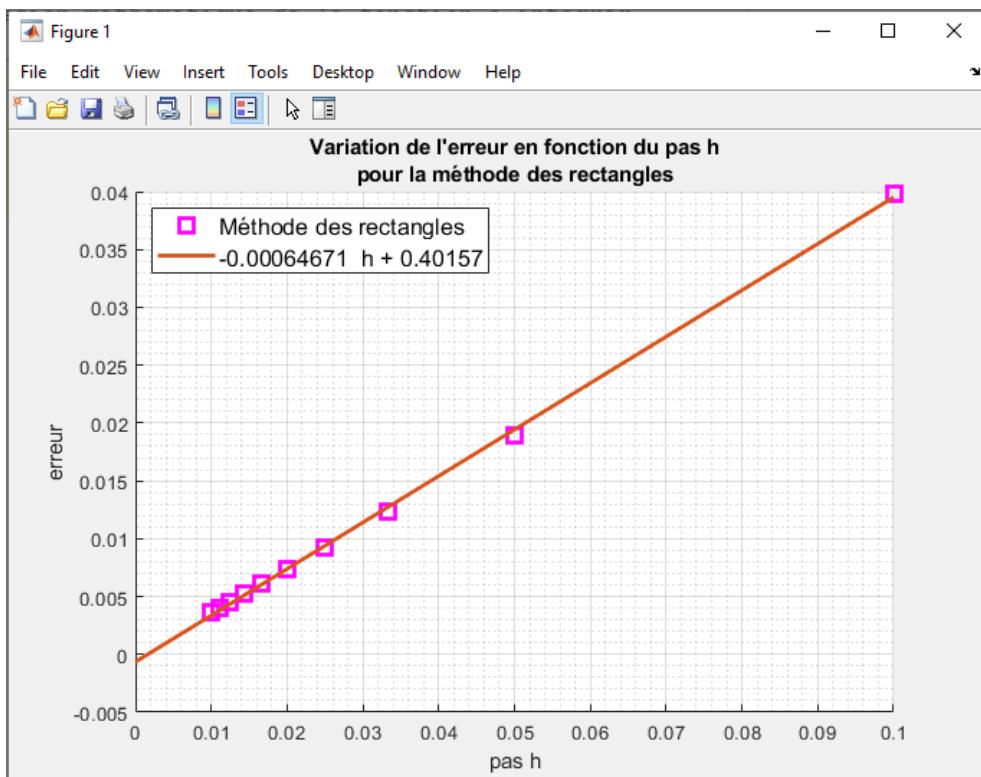


Figure 709 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles

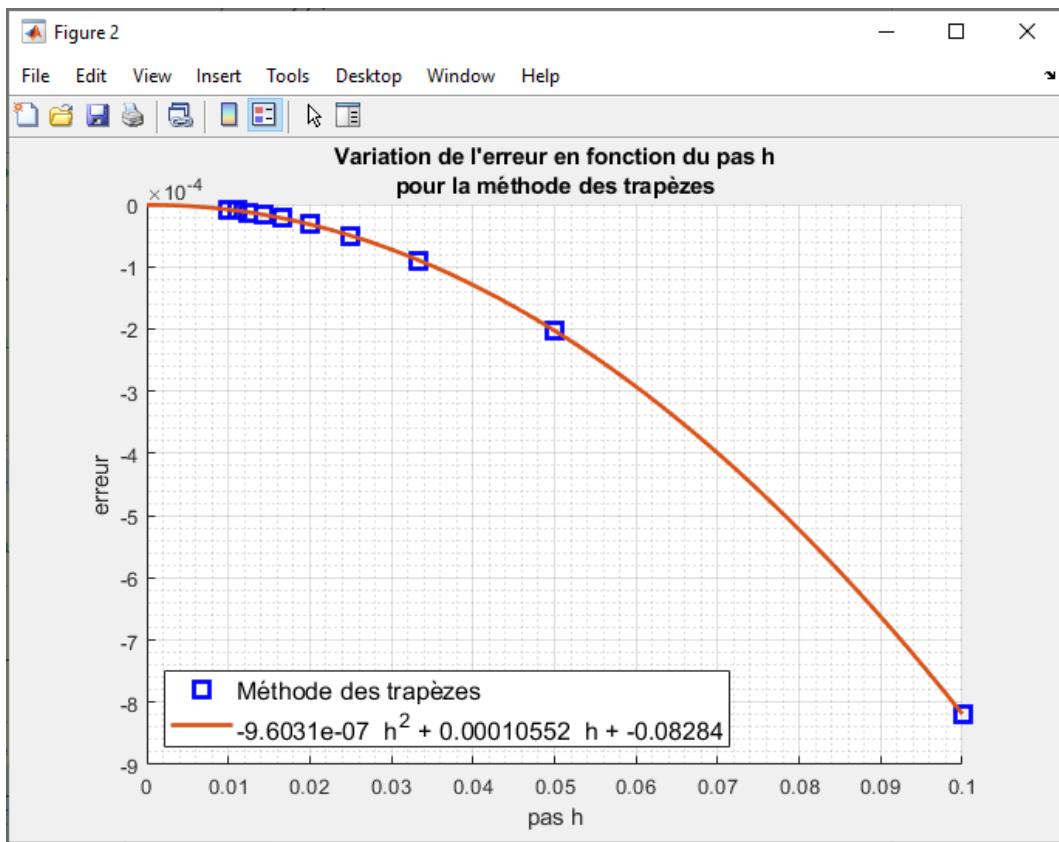


Figure 710 : variation de l'erreur de d'intégration en fonction de h pour la méthode des trapèzes

En observant les points qui représentent l'erreur en fonction du pas h , nous choisissons une régression linéaire pour interpoler l'erreur pour la méthode des rectangles et une interpolation polynomiale d'ordre 2 pour la méthode des trapèzes. Il apparaît que ces choix permettent d'interpoler la série de données avec une excellente précision. Les équations des polynômes d'interpolation apparaissent dans les légendes des différentes fenêtres graphiques.

En conclusion :

- l'erreur pour la méthode des rectangles est d'ordre 1 en h
- l'erreur pour la méthode des trapèzes est d'ordre 2 en h

L'exploitation d'un algorithme peut donc nous amener à connaître la tendance de variation d'un paramètre en exploitant la puissance de calcul et en utilisant les outils adaptés. Notre seule intuition est parfois insuffisante et nous devons nous appuyer sur le potentiel de notre algorithme pour conforter nos prévisions.

4. Robustesse du processus

Afin de tester la robustesse de notre algorithme, nous pouvons tester plusieurs fonctions en modifiant la fonction **f3.m** et en reproduisant le processus.

Modifier la fonction **f3.m** pour reproduire l'analyse sur une fonction polynomiale $y(x) = 5x^3 - 3x^2 + 5x + 1$ (Figure 711).

```
function y = f3(x)
y = 5*x^3-3*x^2+5*x+1;
end
```

Figure 711 : modification de la fonction f1.m

Relancer les différents scripts pour visualiser les résultats dans les différentes figures.

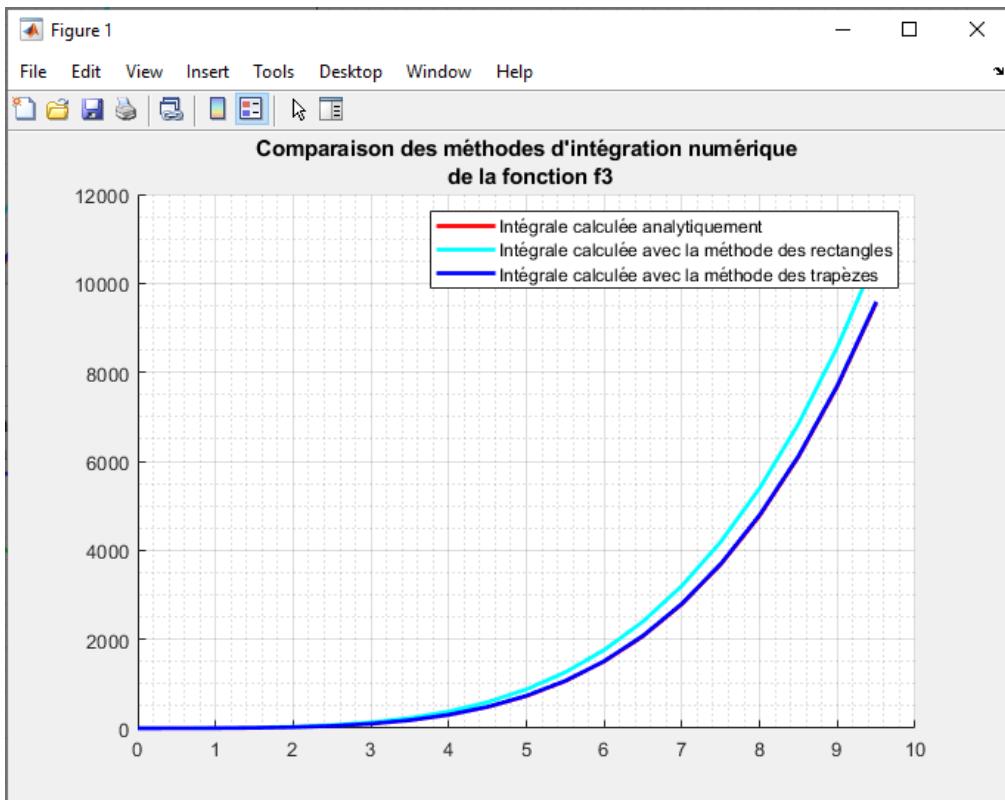


Figure 712 : comparaison des différentes méthodes d'intégration numérique

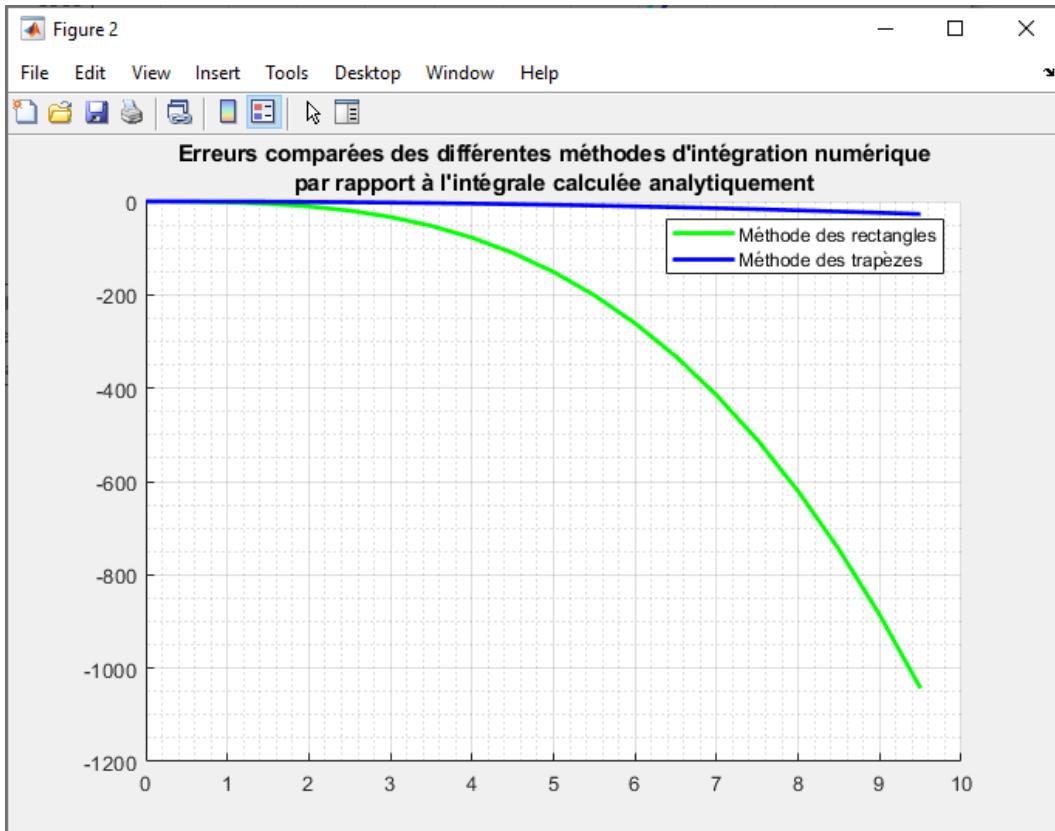


Figure 713 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique

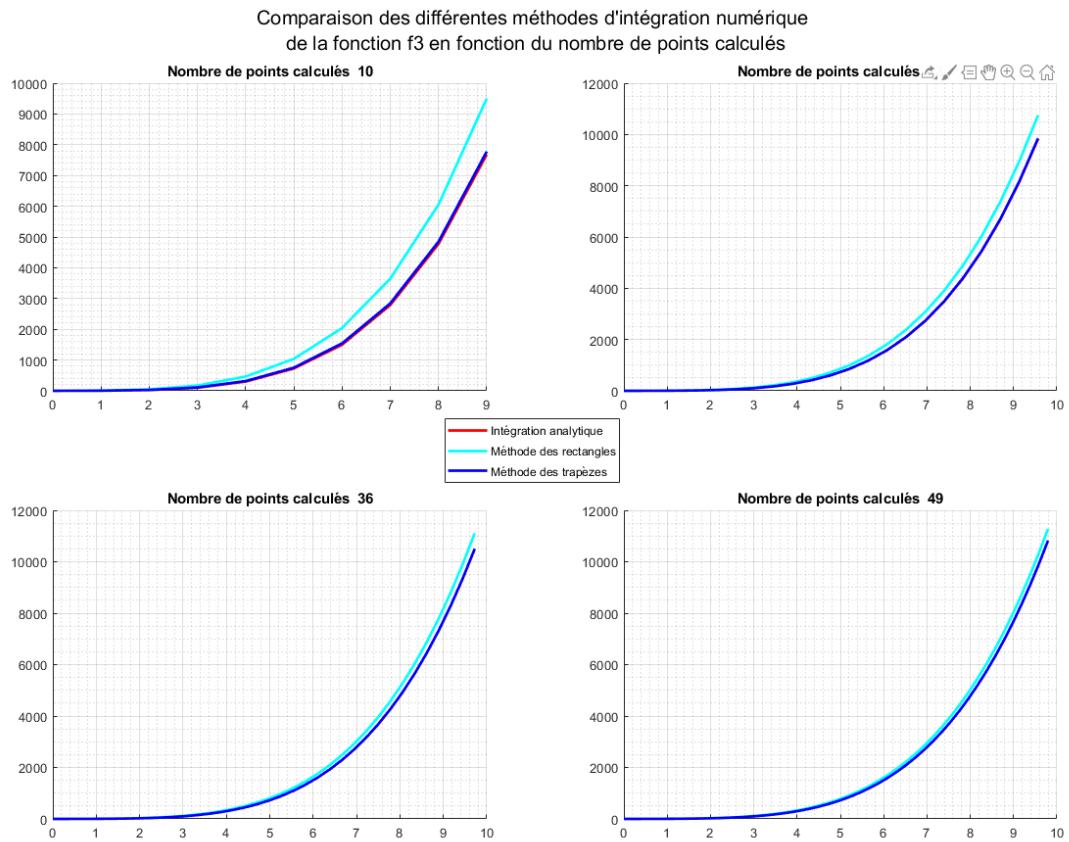


Figure 714 : influence du pas sur l'allure des courbes

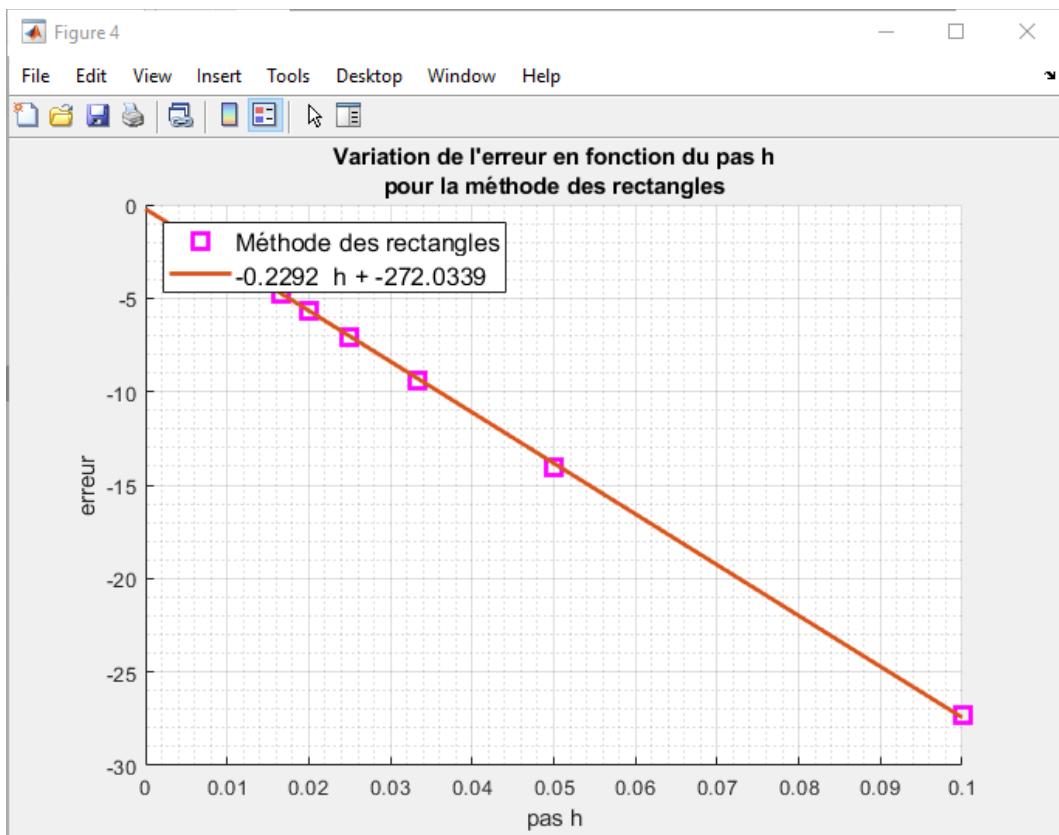


Figure 715 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles

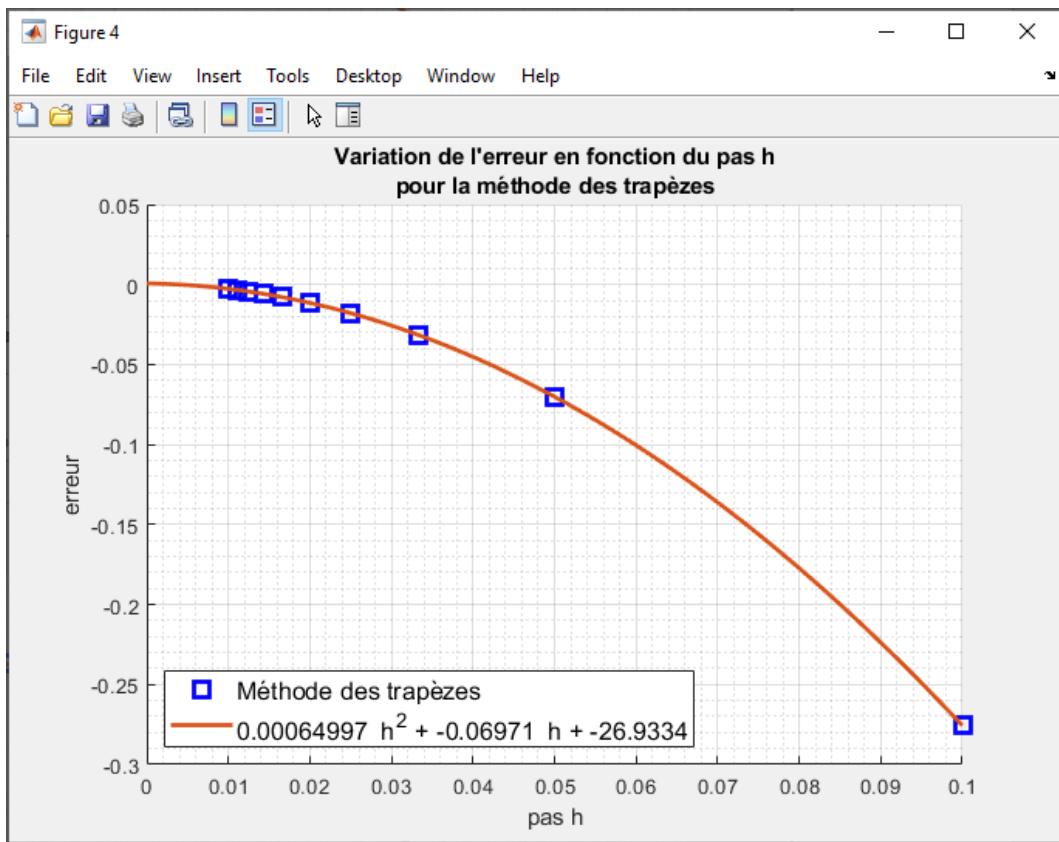


Figure 716 : variation de l'erreur d'intégration en fonction de h pour la méthode des trapèzes

D. Applications

1. Analyse d'une loi en trapèze de vitesse

Nous relevons à l'aide d'un accéléromètre monté sur le chariot d'un axe linéaire le signal $a(t)$ de la Figure 717. Nous souhaitons à partir de ce signal calculer numériquement la vitesse $v(t)$ et la position $x(t)$ du chariot à chaque instant.

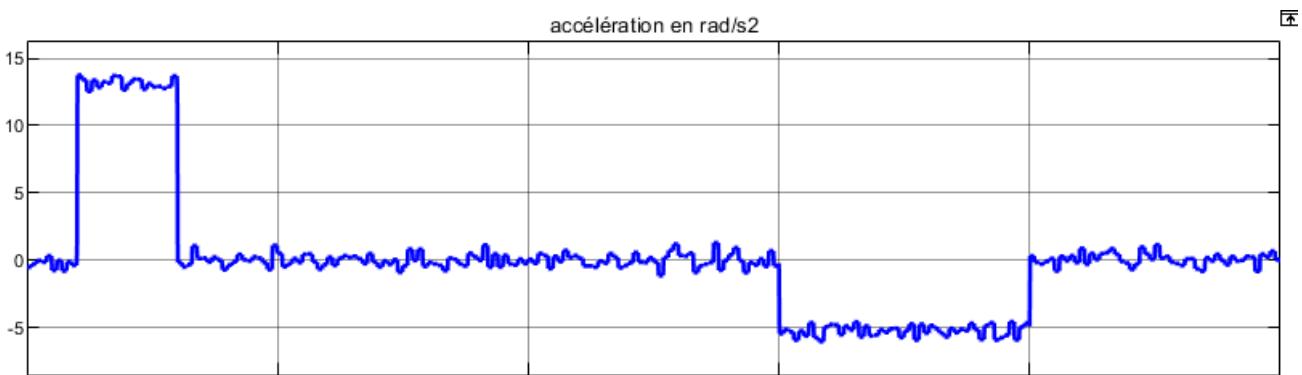


Figure 717 : relevé de l'accélération en fonction du temps

Pour avoir accès à la vitesse et à la position du chariot, nous allons mettre en place un processus d'intégration numérique. En intégrant l'accélération par rapport au temps, nous obtiendrons la vitesse $v(t)$. En intégrant la vitesse par rapport au temps, nous obtiendrons la position $x(t)$.

Nous considérons qu'à l'instant $t=0$, le chariot est à l'arrêt ($v(0)=0$) et que sa position initiale est prise comme origine ($x(0)=0$).

Pour effectuer le processus d'intégration numérique, nous utiliserons la méthode des trapèzes afin de minimiser l'erreur de calcul.

Ouvrir le script **Loi_en_trapeze_de_vitesse.m** (Figure 718).

Ce script permet de charger les données correspondant à la mesure de l'accélération. Ensuite un processus d'intégration numérique calcule la vitesse et la position. Les trois courbes sont tracées dans la même fenêtre graphique.

```
%% Loi_en_trapeze_de_vitesse.m
% Ivan LIEBGOTT @ décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des trapèzes
% Un relevé expérimental donne l'accélération en mm/s2 en fonction du
% temps. En utilisant l'intégration par la méthode des trapèzes, on calcule
% la vitesse et la position.
% les trois courbes sont tracées dans la même fenêtre graphique

% chargement des mesures de l'accélération
clear all;
load loi_en_trapeze_de_vitesse_data.mat

%% Calcul de la vitesse par intégration de l'accélération

% définition des conditions initiale de la vitesse
vitesse(1) = 0;

% La boucle va ajouter l'aire des trapèzes pour chaque intervalle
% d'amplitude h = t(k-1)-t(k)
for k = [2:1:length(t)]
    % calcul du pas d'intégration
    h = t(k) - t(k-1);

    % construction du vecteur contenant les valeur de la vitesse en
    % utilisant la méthode des trapèzes
    vitesse(k) = vitesse(k-1) + (acceleration(k)+acceleration(k-1)) * h/2;
end

%% Calcul de la position par intégration de la vitesse

% définition des conditions initiale de la vitesse
position(1) = 0;

% La boucle va ajouter l'aire des trapèzes pour chaque intervalle
% d'amplitude h = t(k-1)-t(k)
for k = [2:1:length(t)]
    % calcul du pas d'intégration
    h = t(k) - t(k-1);

    % construction du vecteur contenant les valeur de la vitesse en
    % utilisant la méthode des trapèzes
    position(k) = position(k-1) + (vitesse(k)+vitesse(k-1)) * h/2;
end

% Tracé des trois courbes dans la même fenêtre graphique
```

```

figure('Units','centimeters','Position',[2 8 25 20]);
subplot(3,1,1);
plot(t,acceleration);
grid on; grid minor;
xlabel('temps en s');
ylabel('accélération en mm/s^2')
title('Mesure de l''accélération du chariot en fonction du temps en mm/s^2')

subplot(3,1,2);
plot(t,vitesse);
grid on; grid minor;
xlabel('temps en s');
ylabel('vitesse en mm/s')
title({'Vitesse du chariot en fonction du temps en mm/s',...
    'obtenue par intégration numérique'})

subplot(3,1,3);
plot(t,position);
grid on; grid minor;
xlabel('temps en s');
ylabel('Position en mm')
title({'Position du chariot en fonction du temps en mm',...
    'obtenue par intégration numérique'});

xlabel('temps en s');
ylabel('Position en mm')
title({'Position du chariot en fonction du temps en mm',...
    'obtenue par intégration numérique'});

```

Figure 718 : intégration numérique de l'accélération

Exécuter le script et observer le résultat obtenu sur la Figure 719.

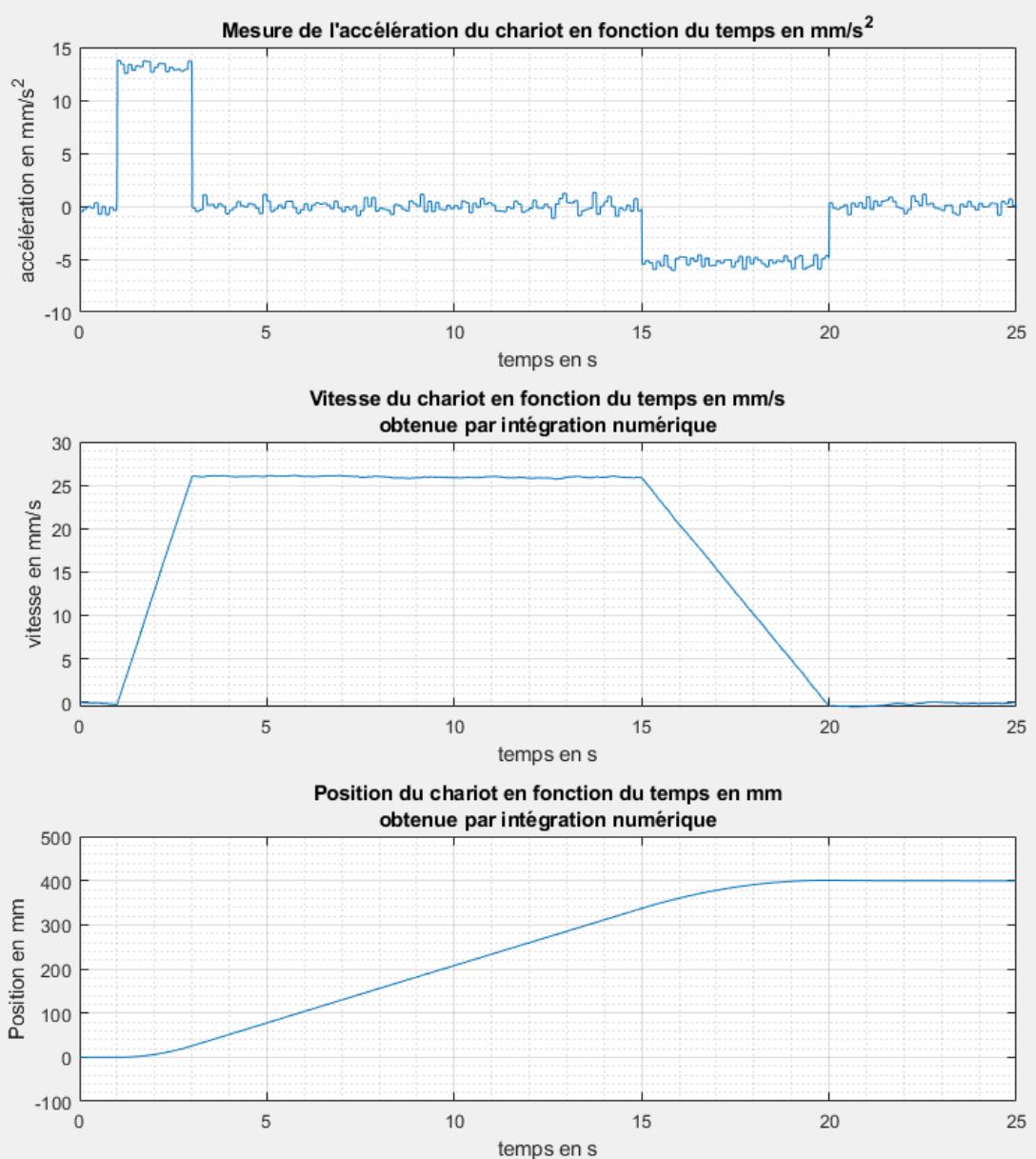


Figure 719 : loi en trapèze de vitesse

L'analyse de la courbe de vitesse nous donne la vitesse maximale atteinte par le chariot qui est de 26 mm/s. La courbe de position nous permet de déterminer la position finale qui est 400 mm.

VI. Filtrage d'un signal

A. Intérêt du filtrage d'un signal

Lors du processus d'acquisition d'une grandeur physique, les signaux provenant des capteurs sont très souvent perturbés par des signaux qui viennent s'ajouter au signal que l'on souhaite mesurer (Figure 720). Ce bruit prend souvent la forme d'un signal haute fréquence. Ces perturbations ou signaux parasites sont de différentes natures :

- Signal périodique 50 Hz, provenant du couplage électromagnétique entre le circuit utilisé et les conducteurs du réseau d'alimentation
- Perturbations électromagnétiques dues à la présence d'onde Hertzienne (téléphone portable, réseaux wifi...)
- Perturbations liées à la nature du circuit (bruit de fond)
- Discontinuités due à la nature des capteurs (codeur incrémental)

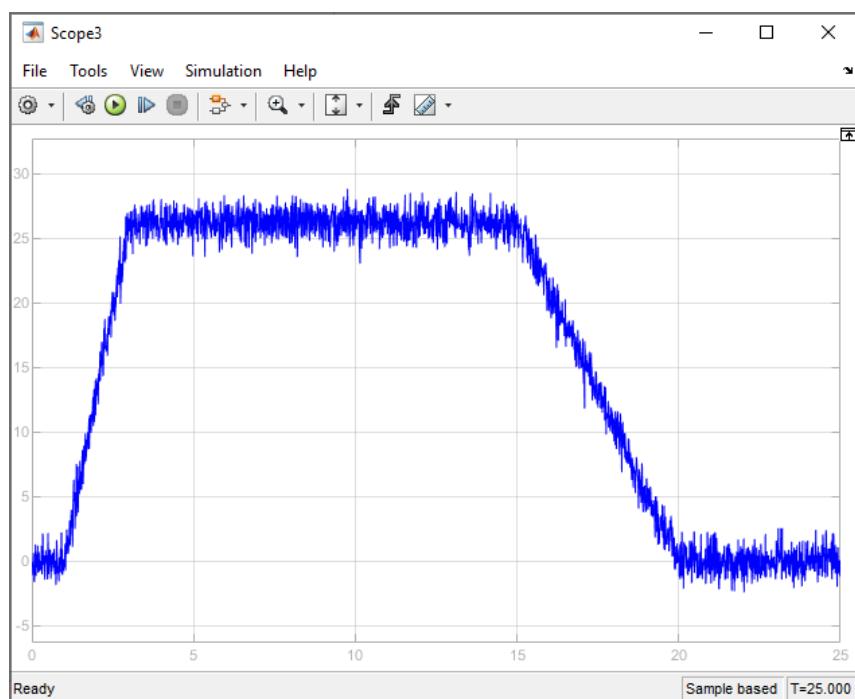


Figure 720 : exemple de signal bruité

Le filtrage consiste à extraire la composante que l'on souhaite mesurer en éliminant tous les bruits parasites ou à lisser un signal en vue d'un processus de traitement numérique (dérivation numérique en particulier).

La Figure 721 illustre le principe du filtrage numérique.



Figure 721 : filtre numérique

y_k : valeur numérique de l'échantillon en sortie de filtre

x_k : valeur numérique de l'échantillon en entrée de filtre

T_e : période d'échantillonnage

Nous verrons dans ce chapitre deux filtres élémentaires qui permettent de mettre en évidence l'intérêt du filtrage d'un signal.

- Le filtre à moyenne glissante
- Le filtre du premier ordre

Nous allons voir comment coder en langage MATLAB ces filtres élémentaires et les utiliser dans quelques cas pratiques.

Nous travaillerons sur le signal issu d'un codeur incrémental représenté sur la Figure 722 et sur la Figure 723. Les discontinuités dans le signal devront être atténouées par le processus de filtrage.

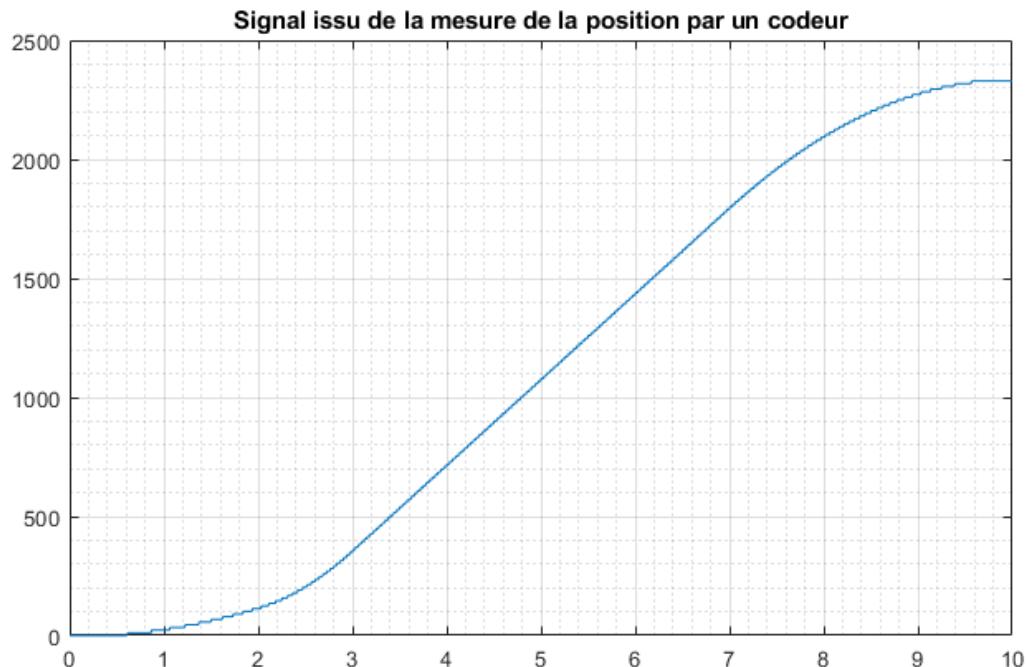


Figure 722 : mesure de la position à partir du signal d'un codeur incrémental

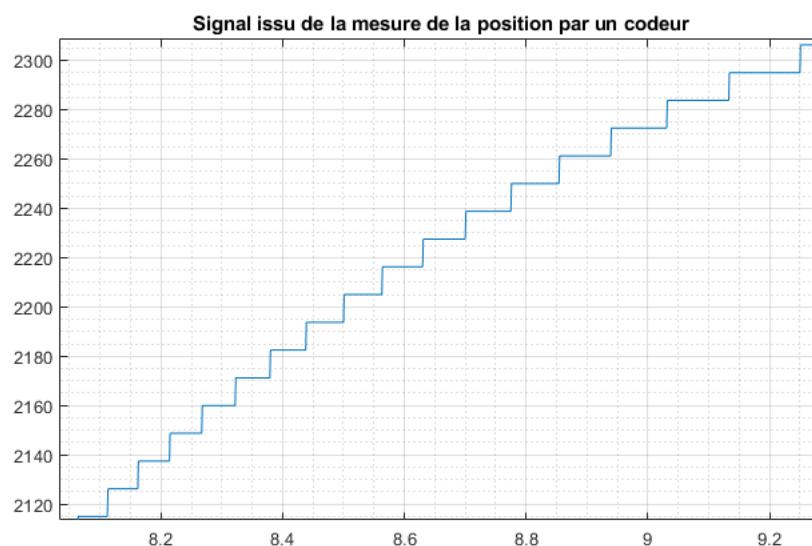


Figure 723 : zoom sur la courbe

B. Le filtre à moyenne glissante

Ce filtre très simple à coder est largement utilisé pour améliorer la qualité d'un signal bruité. Il permet de lisser le signal. Le principe est de calculer l'échantillon y_k comme étant la moyenne des n échantillons d'entrée précédents.

$$y_k = \frac{1}{n} \sum_{i=0}^{n-1} x_{k-i}$$

Par exemple si $n=3$, l'échantillon y_k sera calculé en faisant la moyenne des trois derniers échantillons :

$$y_k = \frac{1}{3}(x_k + x_{k-1} + x_{k-2})$$

La valeur de n sera choisie en fonction de la fréquence du bruit que l'on souhaite supprimer et de la fréquence d'échantillonage. Parfois plusieurs essais doivent être réalisés pour obtenir un lissage satisfaisant sans trop retarder le signal filtré par rapport au signal d'origine. Dans tous les cas il faudra trouver le meilleur compromis sachant que la composante continue subira de toute façon une légère déformation.

La fonction **filtre_mgl(Y,n)** (Figure 724) permet le filtrage par moyenne glissante d'un signal Y en prenant n échantillons pour faire la moyenne.

Cette fonction prendra en arguments :

Y : vecteur contenant les valeurs de la fonction à filtrer

n : nombre d'échantillons à prendre en compte pour la moyenne

La fonction aura comme sorties :

Y_mgl : vecteur contenant les valeurs du signal filtré par moyenne glissante

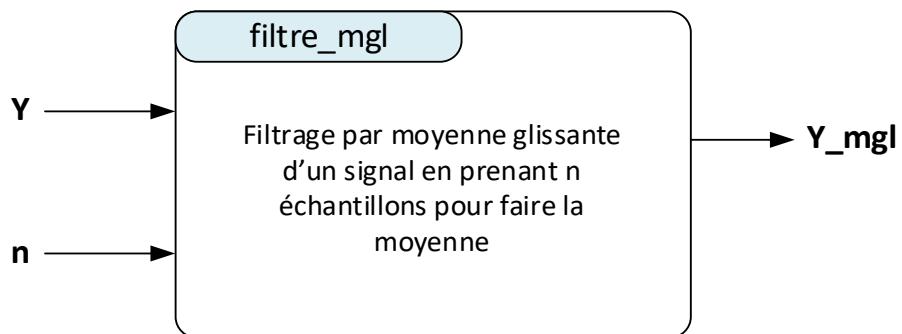


Figure 724 : fonction **filtre_mgl(Y,n)**

La Figure 725 montre le codage de la fonction **filtre_mgl(Y,n)**.

```

%% filtre_mgl(Y,n)
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de filtrer un signal en utilisant un filtre à
% moyenne glissante. Chaque point calculé représente la moyenne des n
% dernières valeurs du signal.
% Y : représente le vecteur contenant les valeurs du signal à filtrer
% n : nombre de points pris en compte pour faire la moyenne

function [Y_mgl] = filtre_mgl(Y,n)
% pour chaque point du signal Y à partir du point Y(n+1)
for p = n+1:1:length(Y)
    % calcul de la moyenne des n termes précédents
    moyenne_glissante = 0;
    for k = 1:1:n
        moyenne_glissante = moyenne_glissante + Y(p-k)/n;
    end
    % affectation de la valeur de l'échantillon pour la valeur de p
    Y_mgl(p) = moyenne_glissante;
% transposition pour obtenir une matrice colonne
Y_mgl = Y_mgl';
end

```

Figure 725 : codage de la fonction filtre_mgl(Y,n)

Afin de tester cette fonction sur un signal en prenant plusieurs valeurs pour **n**, ouvrir le script **filtrage_mgl_n_variable.m** (Figure 726).

```

%% filtrage_mgl_n_variable
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tester la fonction filtre_mgl(Y,n) qui permet de
% filtrer un signal en utilisant le principe de la moyenne glissante
% Le test est réalisé sur le signal issu d'un codeur
% Le script trace le signal non filtré et les signaux obtenus après
% filtrage par moyenne glissante en utilisant différentes valeurs du nombre
% n d'échantillons retenus pour faire la moyenne
%%
% Effacement des variables du workspace
clear all;

% chargement des données correspondant au signal du codeur et au temps dans
% le workspace
load filtrage_mgl.mat;

% spécification des valeurs mini et maxi de n

n_min = 10;
n_max = 110;
pas = round((n_max - n_min) / 5);

% création d'une nouvelle fenêtre graphique
fig = figure('Units','centimeters','Position',[2 5 33 23]);
set(fig,'Color','white');
hold all;

% tracé du signal du codeur non filtré
plot(t,position_codeur,'r','LineWidth',3);

```

```

%création d'une cellule pour stocker les lignes de la légende
Titre_Legende = cell(1,1);
Titre_Legende{1,1} = 'Signal brut';
ind_Legende = 2;

for n = n_min : pas : n_max;

% calcul du signal filtré à l'aide de la fonction filtre_mgl
pos_mgl = filtre_mgl(position_codeur,n);
Titre_Legende{1,ind_Legende} = 'n=' + string(n); % création de la ligne de la
légende

% Tracé du signal filtré correspondant à la valeur de n dans la même fenêtre
graphique
plot(t,pos_mgl);
ind_Legende = ind_Legende + 1;
end

grid on; grid minor;
% affichage de la légende
legend(Titre_Legende,'Location','northwest');

title('Filtrage d''un signal en utilisant un filtre à moyenne glissante avec n
compris entre'...
+ string(n_min) + ' et ' + string(n_max));

```

Figure 726 : script permettant de voir l'influence de n pour un filtre à moyenne glissante

Exécuter le script et observer le résultat sur la Figure 727 et sur la Figure 728.

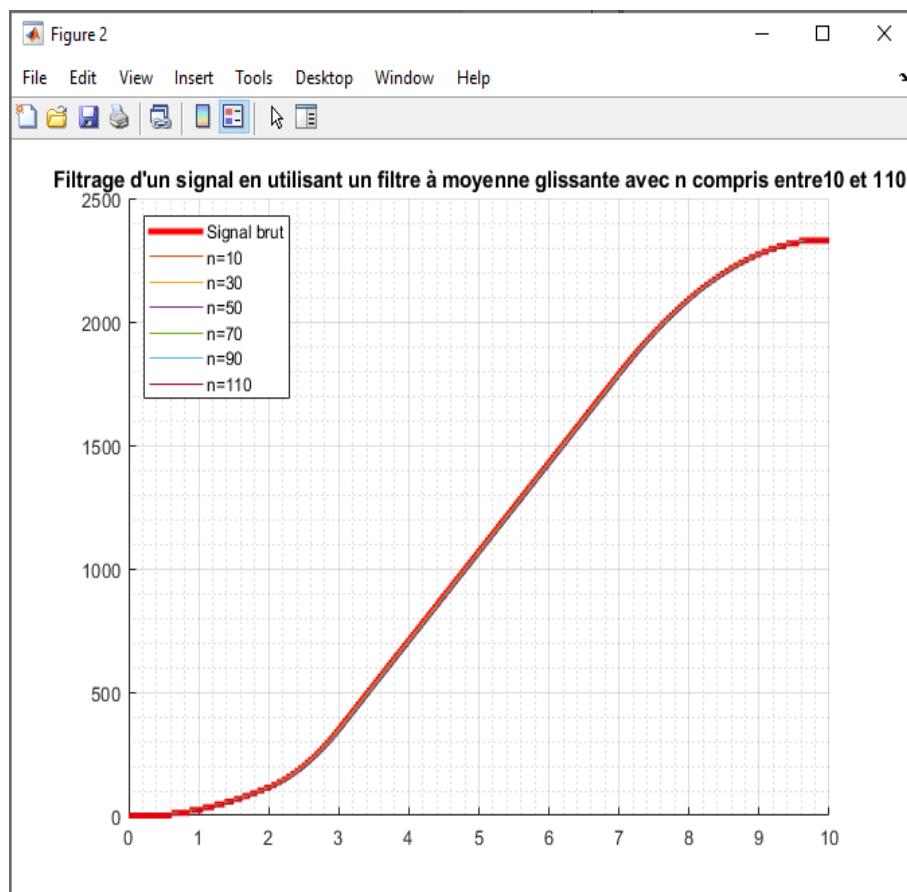


Figure 727 : filtrage par moyenne glissante d'un signal

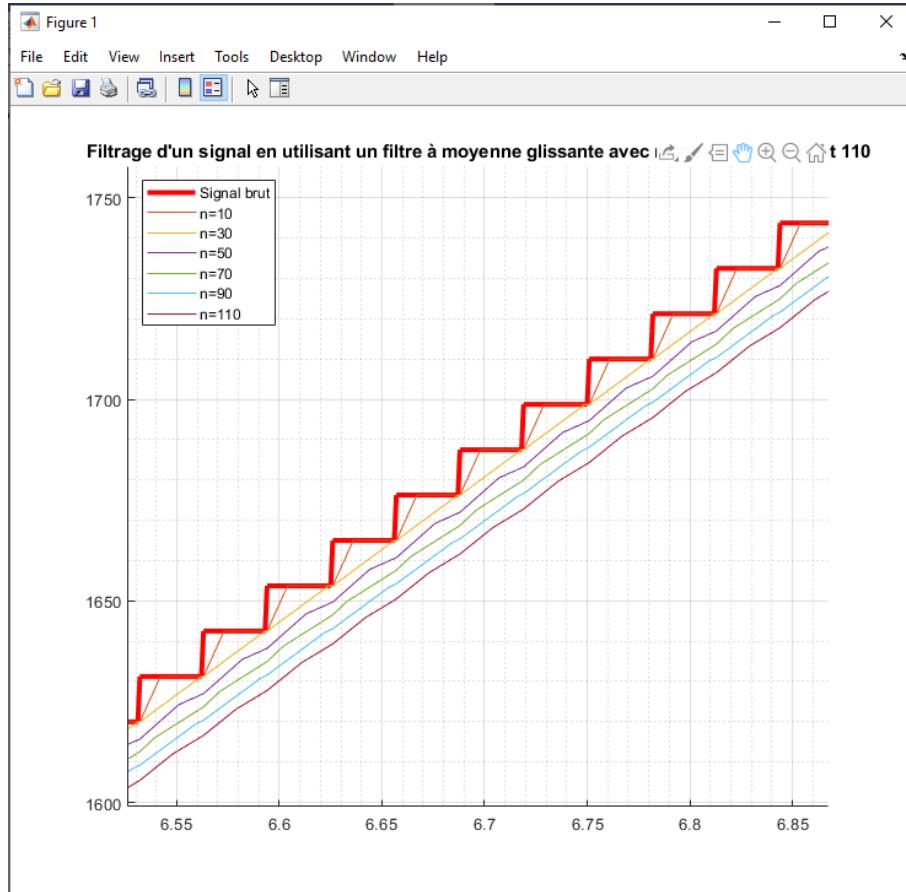


Figure 728 : zoom sur la courbe

Nous pouvons constater que le filtre à moyenne glissante a pour effet de lisser le signal mais également de le retarder. Plus la valeur de n est élevée, plus le signal est lissé et plus il est retardé. Il est donc intéressant de tester plusieurs valeurs de n afin de trouver le meilleur compromis.

C. Le filtre passe-bas du premier ordre

Le filtre passe-bas du premier ordre agit sur le signal en coupant les composantes hautes fréquences tout en laissant passer la composante continue.

La constante de temps du filtre va permettre de choisir la bande de fréquences à partir de laquelle le filtre va atténuer l'amplitude du bruit. Le réglage de la constante τ de temps du filtre dépendra donc de la fréquence des perturbations à éliminer (Figure 729 et Figure 730).

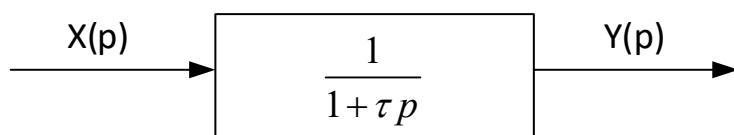


Figure 729 : filtre passe-bas du premier ordre dans le domaine de Laplace

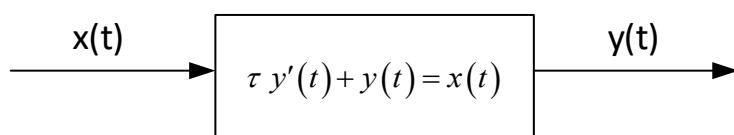


Figure 730 : filtre passe-bas du premier ordre dans le domaine temporel

Dans le domaine temporelle la fonction filtrée $y(t)$ est solution de l'équation différentielle suivante :

$$\boxed{\tau y'(t) + y(t) = x(t)}$$

Si le signal est échantillonné avec une période d'échantillonnage $T_e = h$ et en utilisant la formule de différence finie progressive, il est possible de discréteriser cette équation différentielle :

$$\tau \left(\frac{y_{k+1} - y_k}{h} \right) + y_k = x_k$$

On obtient alors l'équation de récurrence qui permet de calculer l'échantillon y_{k+1} en fonction de y_k et de x_k :

$$\boxed{y_{k+1} = \frac{h}{\tau} x_k + \left(\frac{\tau - h}{\tau} \right) y_k}$$

Cette relation de récurrence sera utilisée par la fonction qui permettra de coder le filtre passe-bas du premier ordre.

La valeur de τ sera choisie en fonction de la fréquence du bruit que l'on souhaite supprimer. Parfois plusieurs essais doivent être réalisés pour obtenir un lissage satisfaisant sans trop retarder le signal filtré par rapport au signal d'origine. Dans tous les cas il faudra trouver le meilleur compromis sachant que la composante continue subira de toute façon une légère déformation.

La fonction **filtre_ordre_1(t,Y,tau)** (Figure 731) permet le filtrage par un filtre passe-bas du premier ordre du signal Y.

Cette fonction prendra en arguments :

Y : vecteur contenant les valeurs de la fonction à filtrer

t : vecteur contenant les valeurs du temps correspondant au vecteur Y

tau : constante de temps du filtre

La fonction aura comme sorties :

Y_pb_ordre_1 : vecteur contenant les valeurs du signal filtré par moyenne glissante

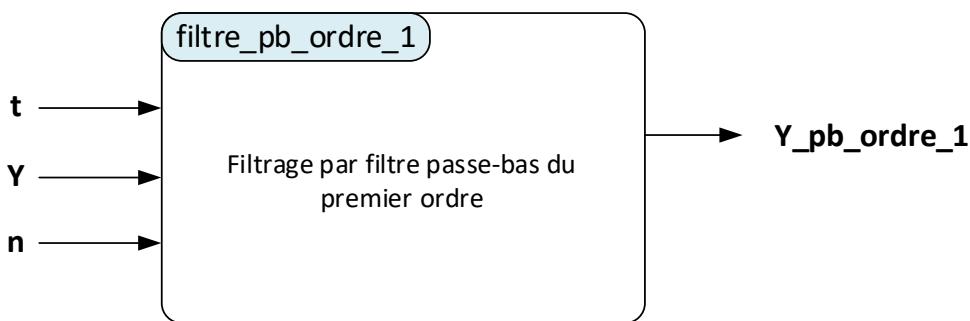


Figure 731 : fonction *filtre_pb_ordre_1(t,Y,tau)*

La Figure 725 montre le codage de la fonction **filtre_pb_ordre_1(t,Y,tau)**.

```

%% filtre_pb_ordre_1(t,Y,tau)
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de filtrer un signal en utilisant un filtre
% passe-bas du premier ordre
% t : vecteur contenant les valeurs du temps correspondant au signal à
% filtrer
% Y : représente le vecteur contenant le valeurs du signal à filtrer
% tau : constante de temps du filtre
function [Y_pb_ordre_1] = filtre_pb_ordre_1(t,Y,tau)
% définition du pas h
h = (t(end) - t(1)) / (length(t)-1);
Y_pb_ordre_1(1) = Y(1);
for k = 1:1:length(Y)-1
    % calcul de Y au rang k+1
    Y_pb_ordre_1(k+1) = Y(k)*(h/tau) + Y_pb_ordre_1(k)*(tau-h)/tau;
end

```

Figure 732 : codage de la fonction filtre_pb_ordre_1(t,Y,tau)

Afin de tester cette fonction sur un signal en prenant plusieurs valeurs pour **tau**, ouvrir le script **filtrage_pb_ordre_1_tau_variable.m**.

```

%% filtrage_pb_ordre_1_tau_variable
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tester la fonction filtre_pb_ordre_1.m qui permet de
% filtrer un signal en utilisant un filtre passe-bas du premier ordre
% Le test est réalisé sur le signal issu d'un codeur
% Le script trace le signal non filtré et les signaux obtenus après
% filtrage en utilisant différentes valeurs pour la constante de temps tau
% du filtre d'échantillons retenus pour faire la moyenne
%%
% Effacement des variables du workspace
clear all;

% chargement des données correspondant au signal du codeur et au temps dans
% le workspace
load filtrage_pb_ordre_1.mat;

% spécification des valeurs mini et maxi de n
tau_min = 0.3;
tau_max = 2;
pas =((tau_max - tau_min) / 5);

% création d'une nouvelle fenêtre graphique
fig = figure('Units','centimeters','Position',[2 5 33 23]);
set(fig,'Color','white');
hold all;

% tracé du signal du codeur non filtré
plot(t,position_codeur,'r','LineWidth',2);

% création d'une cellule pour stocker les lignes de la légende
Titre_Legende = cell(1,1);
Titre_Legende{1,1} = 'Signal brut';
ind_Legende = 2;

```

```

for tau = tau_min : pas : tau_max;
% calcul du signal filtré à l'aide de la fonction filtre_mgl
pos_ordre_1 = filtre_pb_ordre_1(t,position_codeur,tau);
Titre_Legende{1,ind_Legende} = 'tau=' + string(tau); % création de la ligne de
la légende

% Tracé du signal filtré correspondant à la valeur de n dans la même fenêtre
graphique
plot(t,pos_ordre_1);
ind_Legende = ind_Legende + 1;
end

grid on; grid minor;
% affichage de la légende
legend(Titre_Legende,'Location','northwest');

title('Filtrage d''un signal en utilisant un filtre du premier ordre avec tau
comprise entre ...
+ string(tau_min) + ' et ' + string(tau_max));

```

Figure 733 : script permettant de voir l'influence de tau pour un filtre passe-bas du premier ordre

Exécuter le script et observer le résultat sur la Figure 733 et sur la Figure 734.

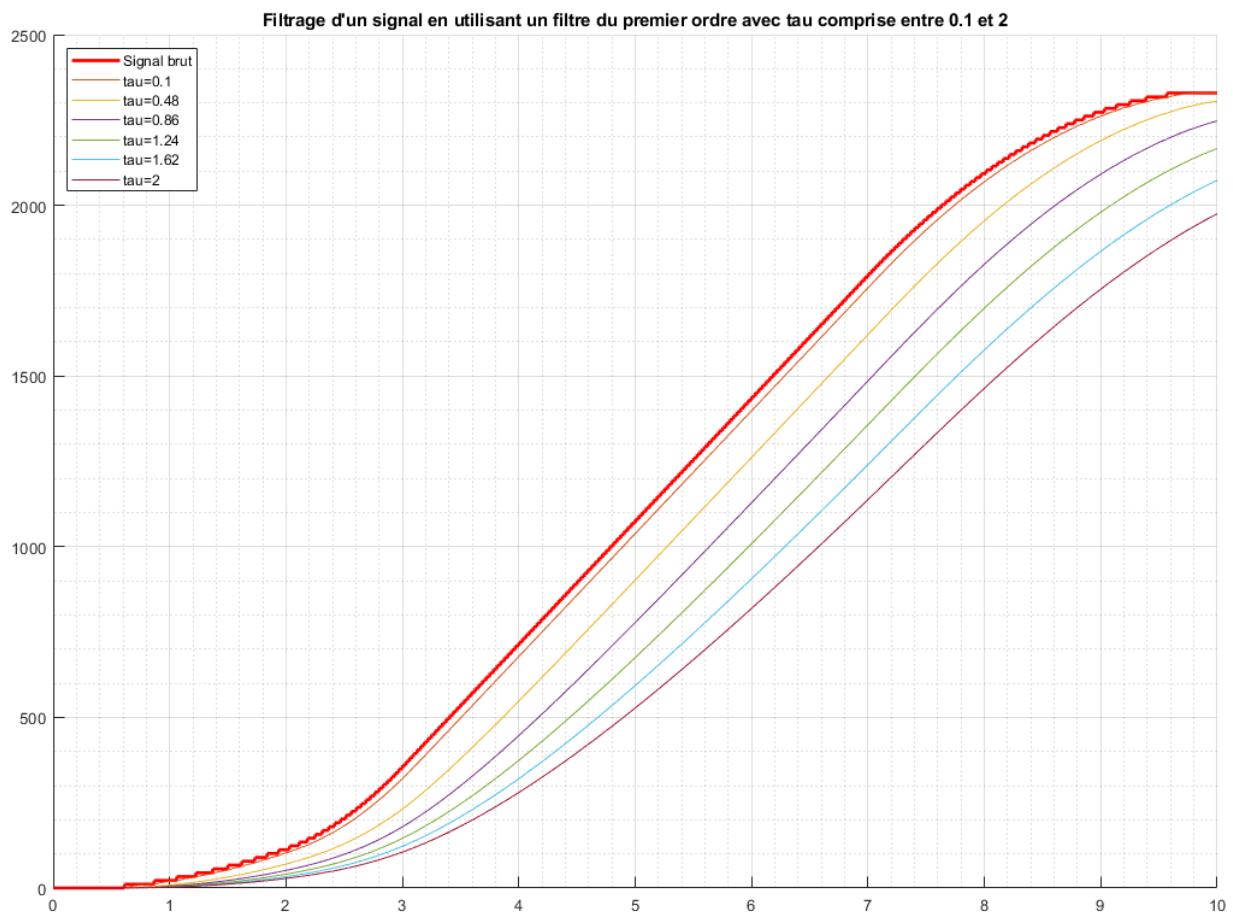


Figure 734 : filtrage d'un signal par un filtre passe-bas du premier ordre avec différentes valeurs de τ

Nous pouvons constater que le filtre passe-bas du premier ordre a pour effet de lisser le signal mais également de le retarder. Plus la valeur de τ est élevée et plus le signal est retardé. Il est donc intéressant de tester plusieurs valeurs de τ afin de trouver une valeur de τ adaptée.

D. Utilisation de Simulink pour le filtrage d'un signal

Simulink propose de nombreuses fonctionnalités permettant de filtrer un signal. Il est possible d'utiliser les blocs classiques comme les fonctions de transfert ou d'utiliser des blocs de la DSP System Toolbox qui propose tous les blocs dédiés au traitement du signal.

Afin de comparer les différentes fonctionnalités, nous allons lui appliquer le processus de traitement indiqué sur la Figure 735. Le signal sera dans un premier temps filtré afin de lui appliquer une méthode de dérivation numérique.

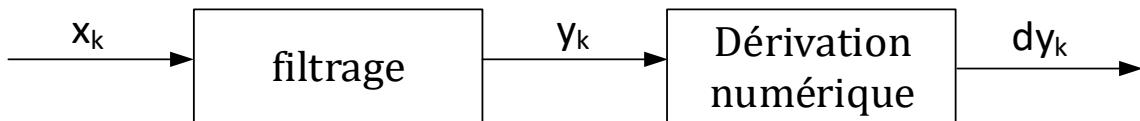


Figure 735 : processus de traitement du signal

Ouvrir le modèle **Simulink_numerical_derivative.slx** et lancer la simulation. Ce modèle permet de visualiser un signal provenant d'un codeur incrémental ainsi que sa dérivée numérique obtenue en utilisant le bloc **Numerical Derivative** de la bibliothèque **Continuous** de Simulink.

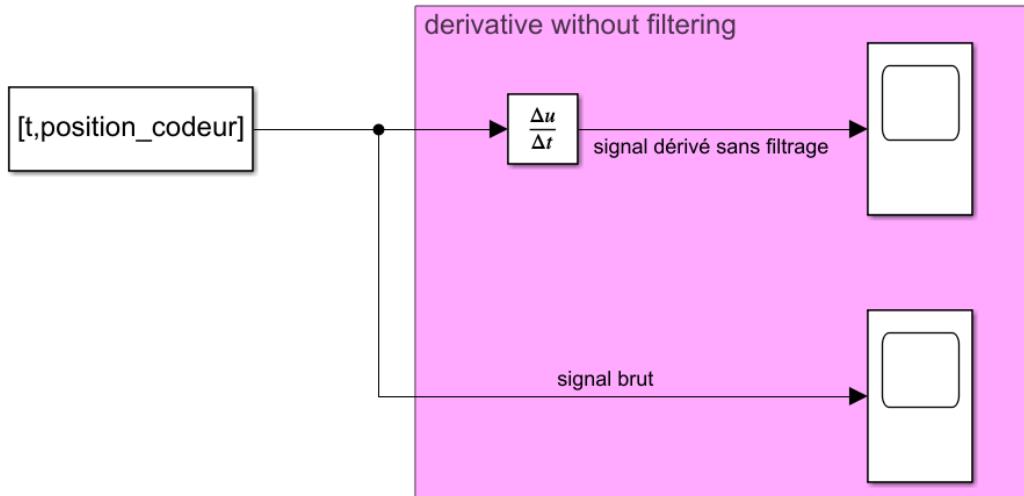


Figure 736 : dérivation numérique en utilisant Simulink

Ouvrir le scope permettant de visualiser le signal brut et observer la forme du signal sur la Figure 737. Nous pouvons constater que ce signal présente des discontinuités qui vont nuire au processus de dérivation numérique.

Ouvrir ensuite le scope correspondant au signal dérivé afin d'observer l'influence de ces discontinuités sur le calcul de la dérivée numérique (Figure 738).

Nous pouvons constater que le signal dérivé n'est pas exploitable, un processus de filtrage est nécessaire afin de lisser le signal avant le processus de dérivation numérique. Plusieurs méthodes seront présentées dans ce paragraphe.

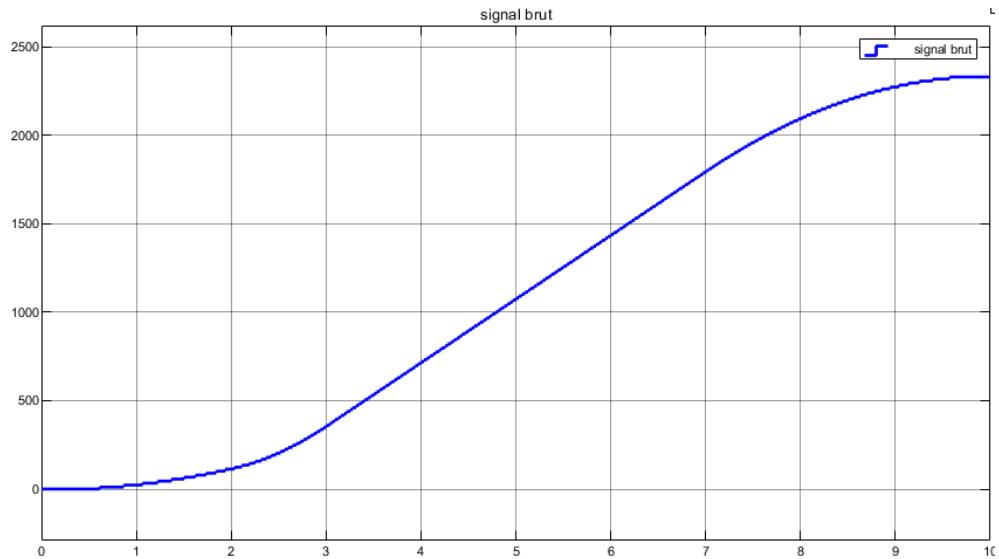


Figure 737 : signal brut à dériver

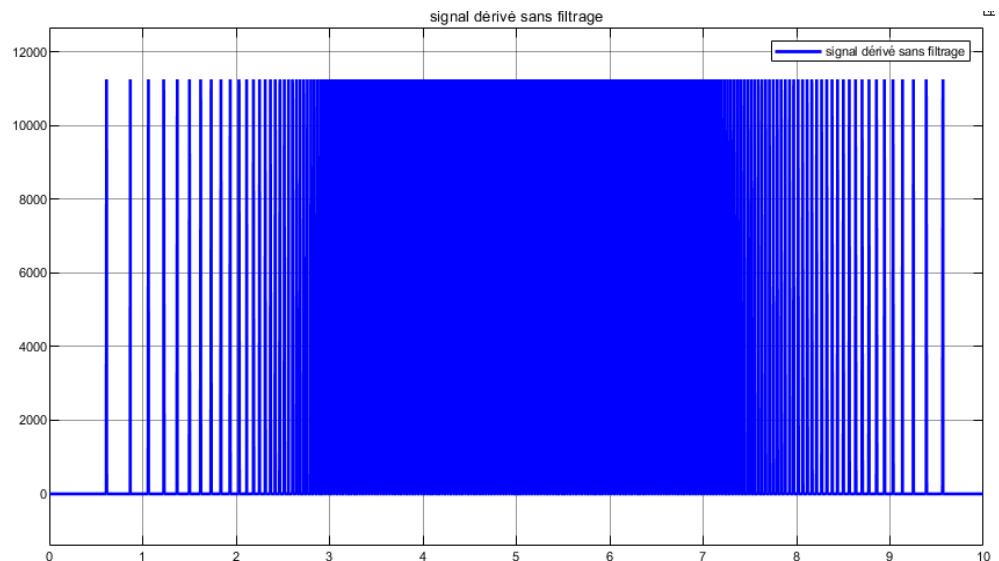


Figure 738 : dérivation d'un signal numérique discontinu sans filtrage

1. Utilisation d'une fonction de transfert de la bibliothèque Continuous de Simulink

Afin de reproduire l'effet d'un filtre passe-bas du premier ordre, il est possible d'utiliser un bloc **Transfer Fcn** de la bibliothèque **Continuous**.

Ouvrir le modèle Simulink **filtrage_Simulink_Transfert_Function.slx** et lancer la simulation (Figure 739). Ce modèle contient une fonction de transfert du premier ordre qui permet de modéliser le comportement d'un filtre passe-bas du premier ordre. La Figure 740 montre le paramétrage de la fonction de transfert $H(s) = \frac{1}{1 + 0.3 s}$, où s représente la variable de Laplace.

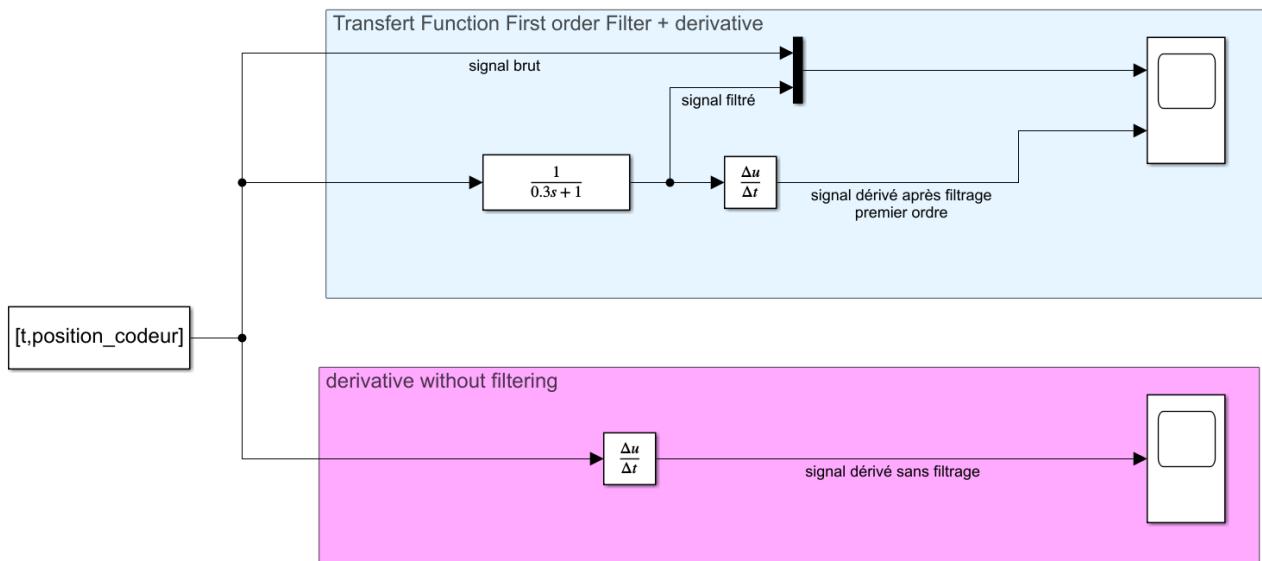


Figure 739 : filtrage réalisé avec une fonction de transfert

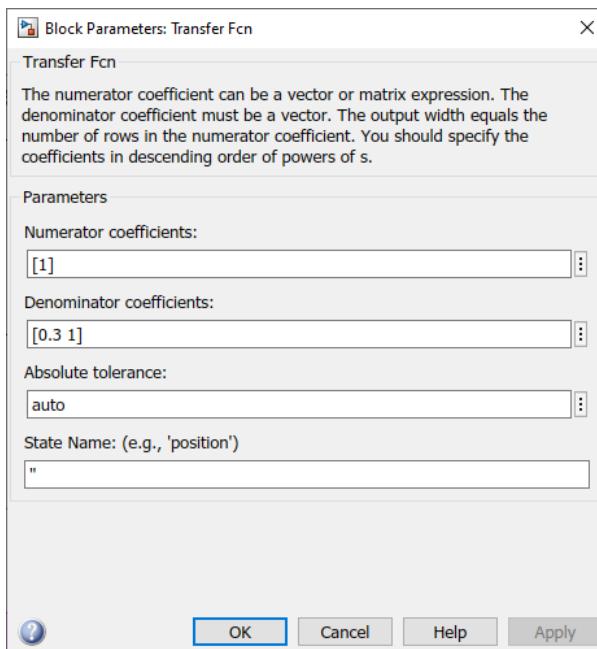


Figure 740 : paramétrage du bloc Transfer Fcn

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 741 permet de voir l'apport du filtrage. Le signal dérivé est maintenant exploitable. La constante de temps a été réglée à 0.3 s. Il est possible de faire d'autre simulation en faisant varier la constante de temps pour visualiser l'effet sur le signal filtré et sur le signal dérivé.

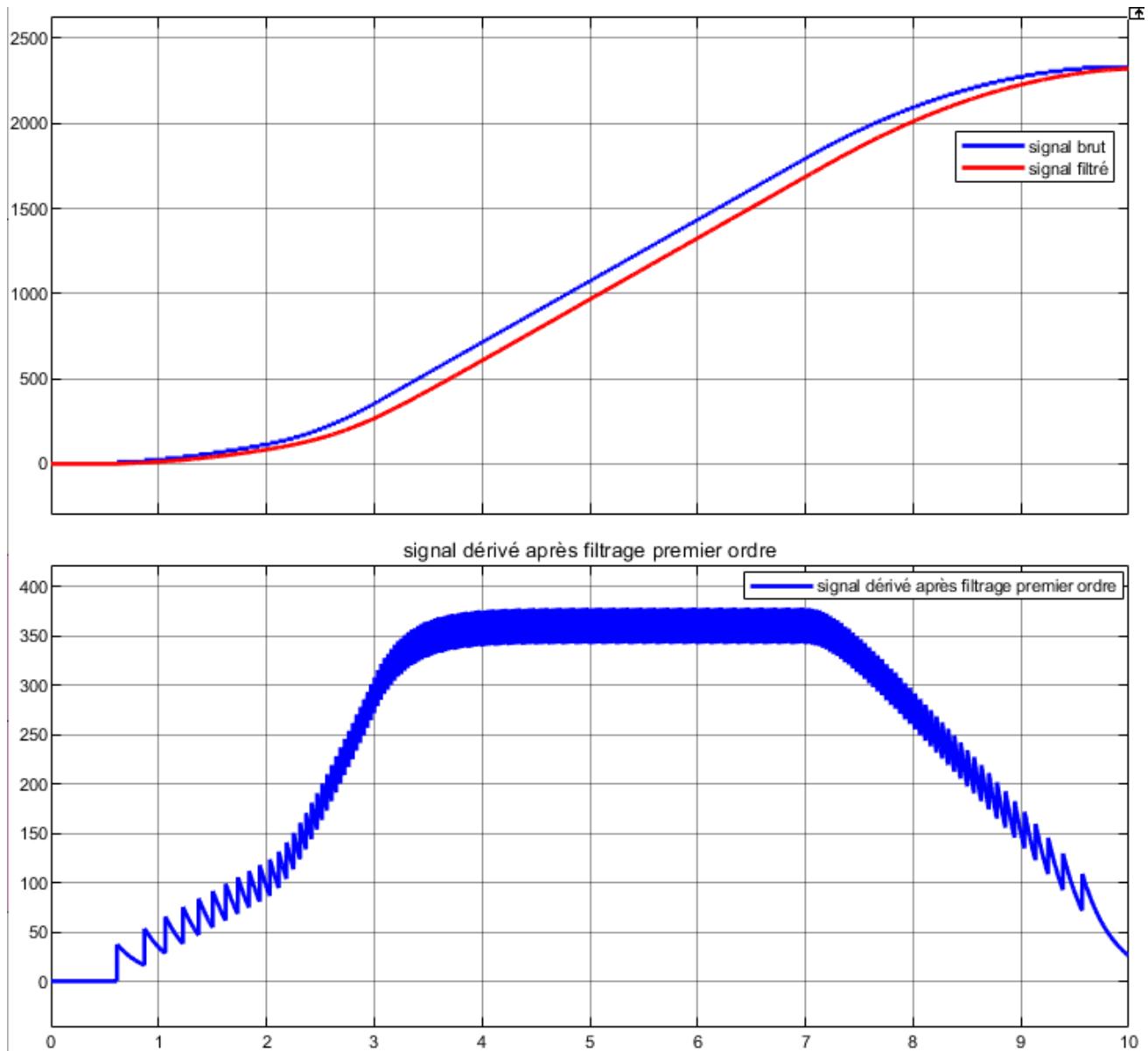


Figure 741 : dérivation numérique d'un signal filtré en utilisant Simulink

2. Utilisation du bloc Analog Filter Design de la DSP Toolbox de Simulink

Afin de reproduire l'effet d'un filtre passe-bas du premier ordre, il est également possible d'utiliser le bloc **Analog Filter Design** de la **DSP Toolbox** de Simulink (Figure 742).

Ouvrir le modèle Simulink **filtrage_Simulink_DSP_toolbox.slx** et lancer la simulation. Ce modèle contient une fonction de transfert du premier ordre qui permet de modéliser le comportement d'un filtre passe-bas du premier ordre.

La Figure 743 montre le paramétrage du bloc **Analog Filter Design**.

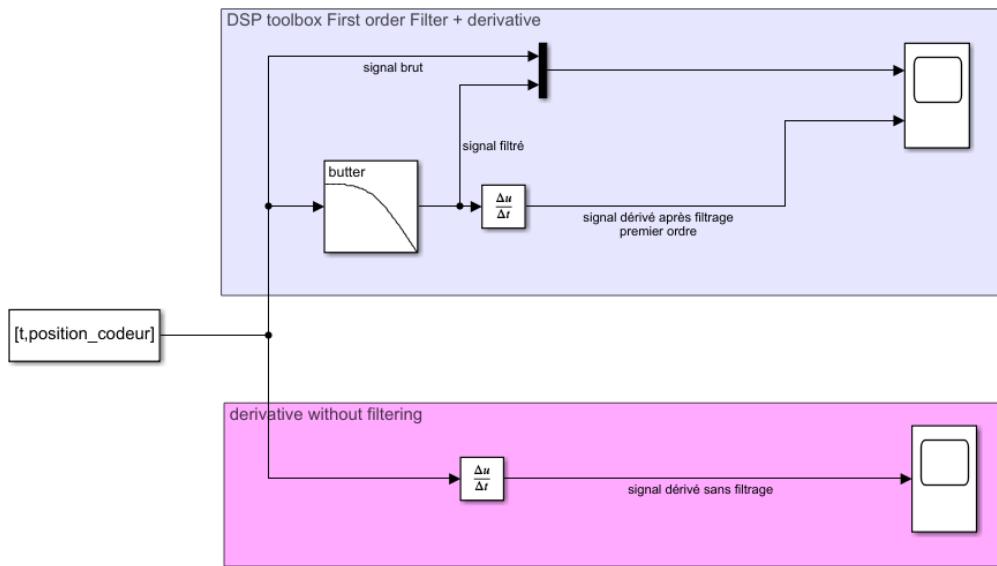


Figure 742 : filtrage réalisé avec le bloc Analog Filter Design

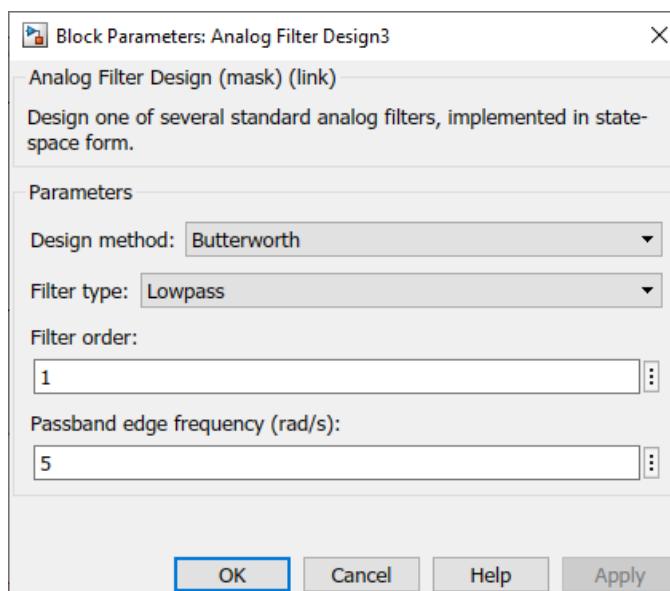


Figure 743 : paramétrage du bloc Analog Filter Design

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 744 permet de voir l'apport du filtrage. Le signal dérivé est maintenant exploitable. En réglant le paramètre « **Passband edge frequency** » à 5 rad/s, nous obtenons un résultat satisfaisant Il est possible de faire d'autre simulation en faisant varier ce paramètre pour visualiser l'effet sur le signal filtré et sur le signal dérivé.

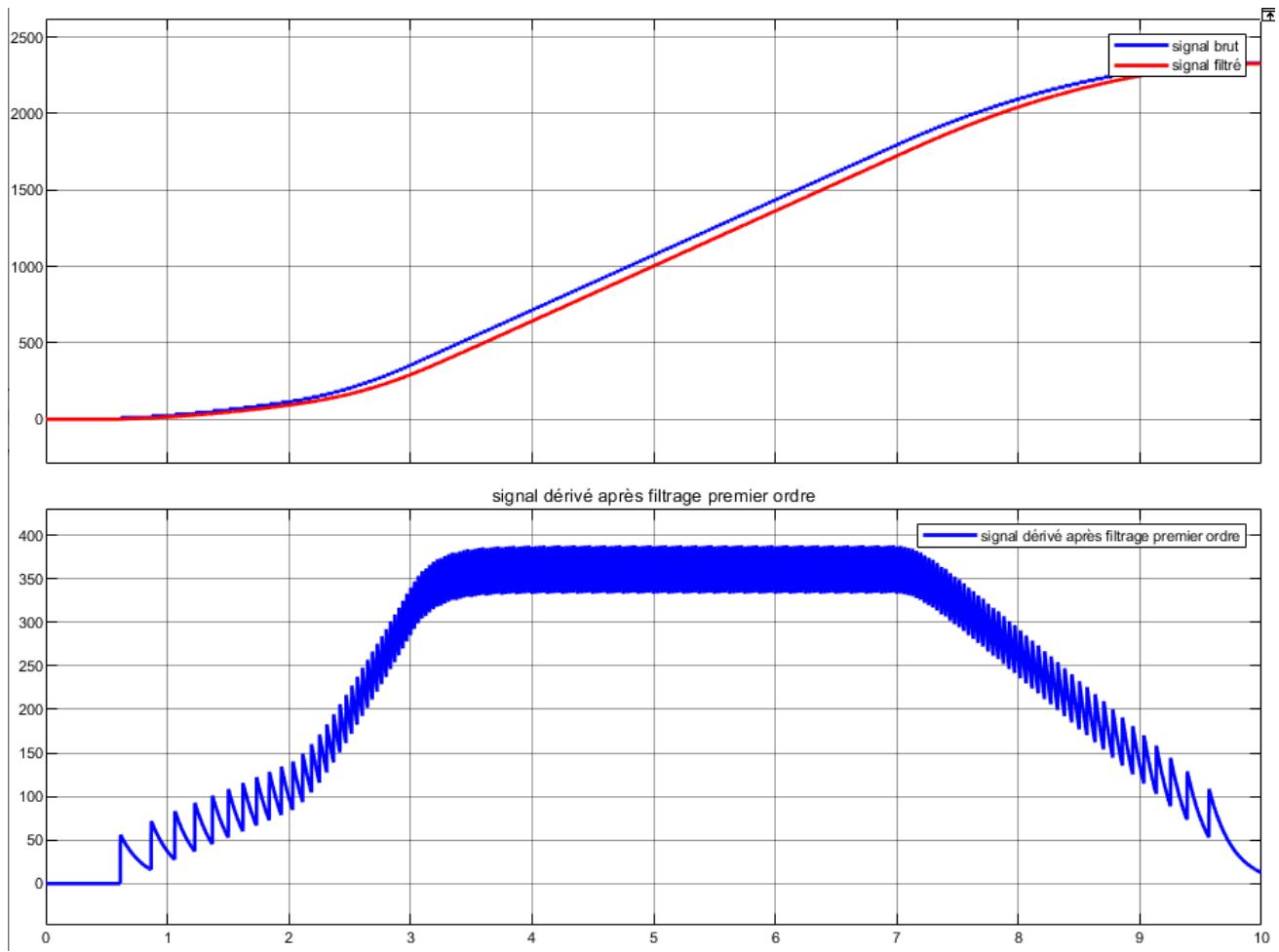


Figure 744 : dérivation numérique d'un signal filtré en utilisant Simulink

3. Utilisation du bloc Moving Average de la DSP System Toolbox

Afin de reproduire l'effet d'un filtre à moyenne glissante, il est également possible d'utiliser le bloc **Moving Average** de la **DSP system Toolbox** de Simulink.

Ouvrir le modèle Simulink **filtrage_Simulink_Moving_average.slx** et lancer la simulation (Figure 745). Ce modèle contient un bloc **Moving Average** de la **DSP System Toolbox**.

La Figure 746 montre le paramétrage du bloc **Moving Average**.

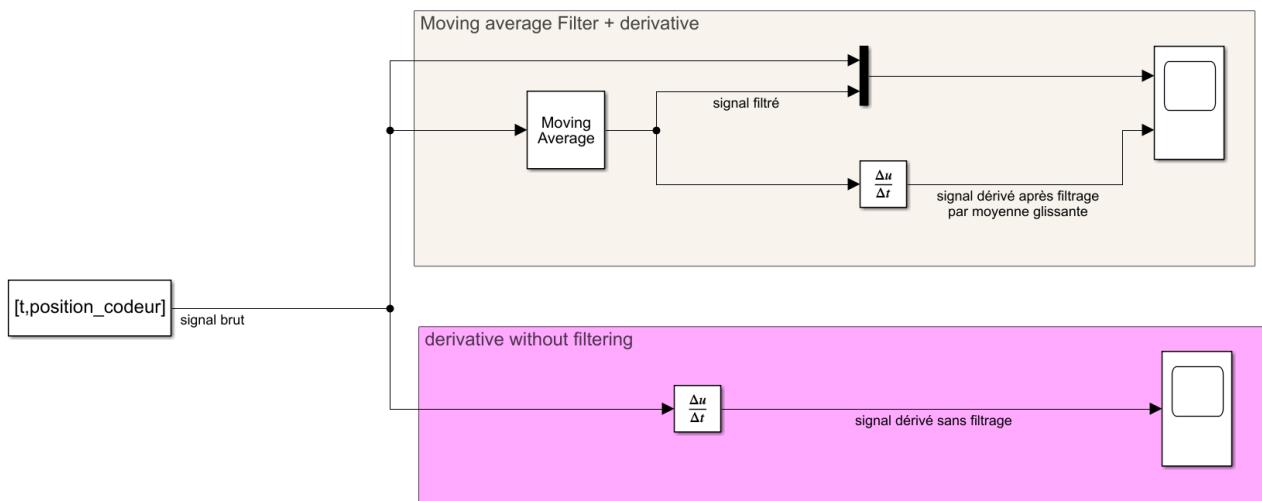


Figure 745 : filtrage réalisé avec le bloc Moving Average

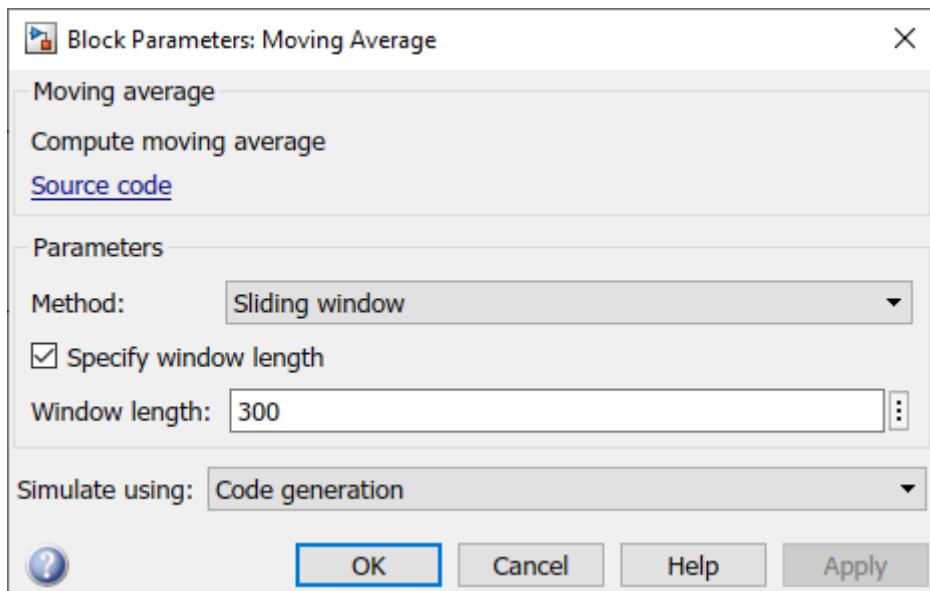


Figure 746 : paramétrage du bloc Moving Average

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 747 permet de voir l'apport du filtrage par moyenne glissante. Le signal dérivé est maintenant exploitable. En réglant le paramètre « **Window length** » à 300, on obtient un résultat satisfaisant (la moyenne est calculée avec les 300 derniers échantillons du signal). Il est possible de faire d'autre simulation en faisant varier ce paramètre pour visualiser l'effet sur le signal filtré et sur le signal dérivé.

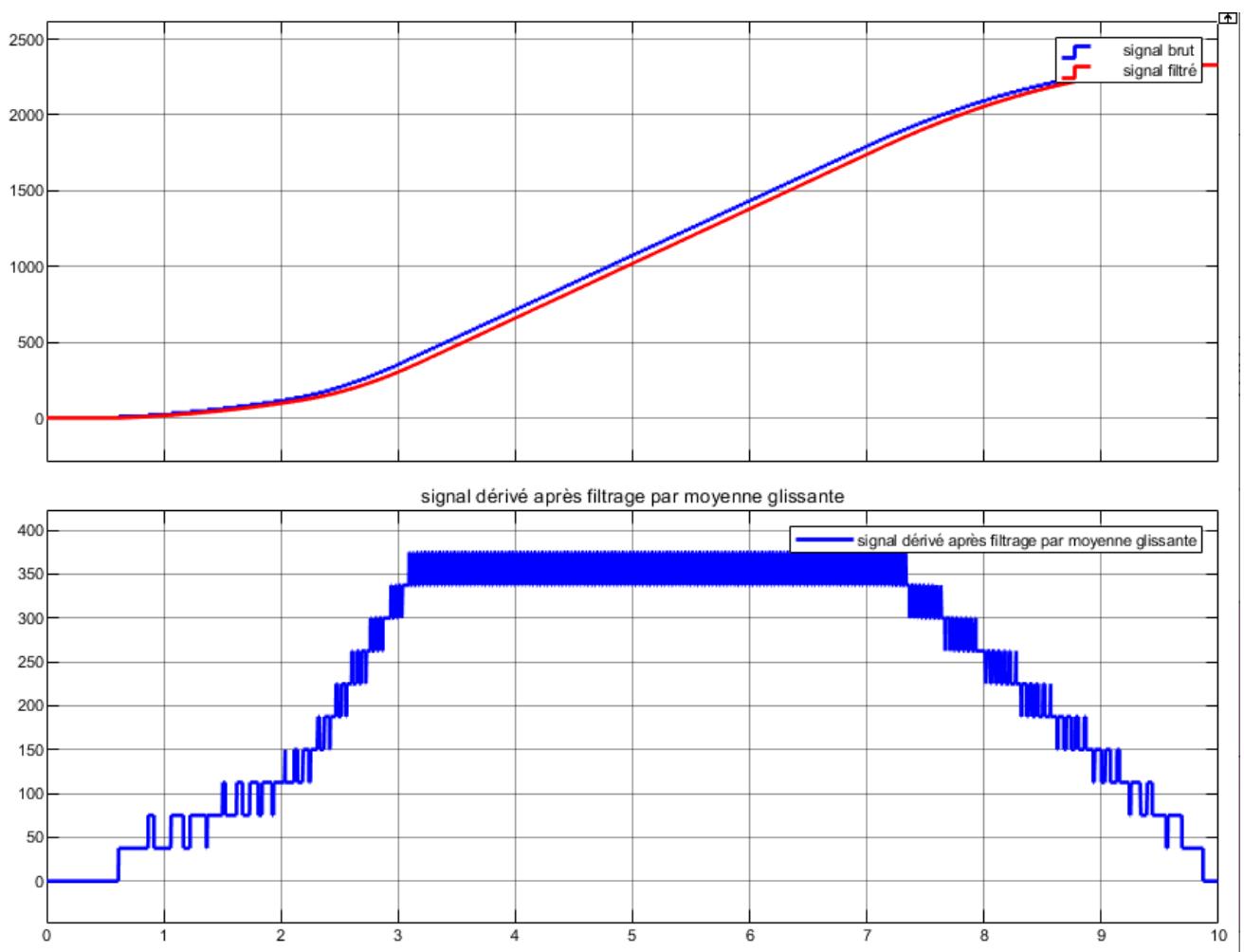


Figure 747 : dérivation numérique d'un signal filtré en utilisant Simulink

VII. Résolution numérique des équations différentielles

De nombreux problèmes scientifiques se modélisent sous la forme d'une équation différentielle. Un processus de résolution doit alors être défini afin de déterminer les solutions. Dans quelques cas particuliers, l'équation différentielle admet une solution qui peut être déterminée par un processus de calcul analytique. En général, les équations différentielles obtenues ne sont pas linéaires et la mise en place d'un processus de résolution numérique est nécessaire afin de déterminer une solution approchée au problème.

A. Résolution d'une équation différentielle d'ordre 1 – Méthode d'Euler

1. Aspects théoriques

La méthode d'Euler va nous permettre de trouver la solution d'une équation différentielle d'ordre 1 donnée avec sa condition initiale sous la forme suivante :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

Cette mise en forme d'une équation différentielle est appelée **problème de Cauchy**. Sous de bonnes hypothèses, le théorème de Cauchy-Lipschitz affirme qu'un tel problème admet une solution unique.

Exemple : Considérons l'équation différentielle suivante :

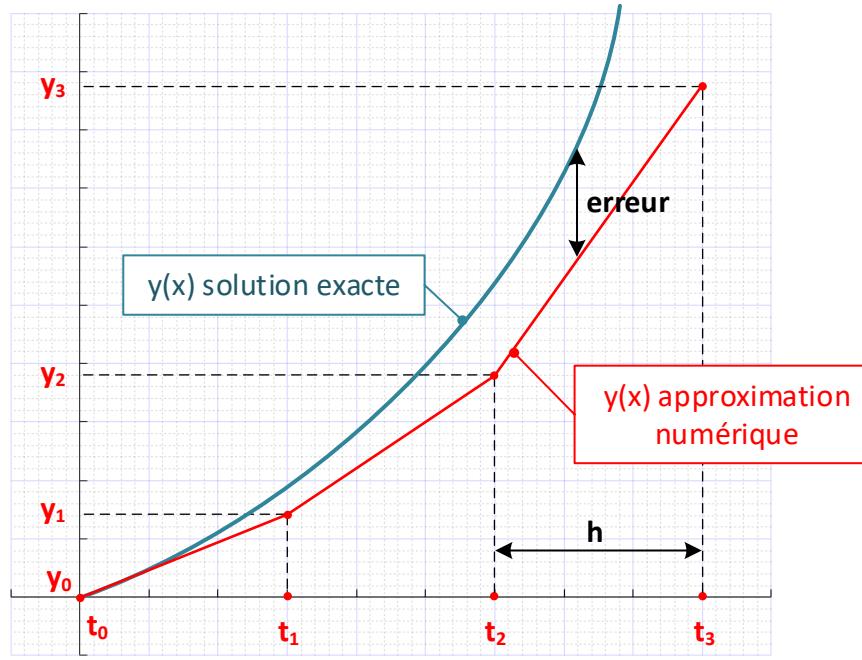
$$y'(t) + y(t) = \sin(t) \text{ avec } y(0) = 1$$

La mise sous la forme d'un problème de Cauchy de cette équation sera donc la suivante :

$$\begin{cases} y'(t) = \sin(t) - y(t) \\ y(0) = 1 \end{cases} \quad \text{avec} \quad f(t, y(t)) = \sin(t) - y(t)$$

Principe de la méthode : La fonction $f(t_i, y(t_i)) = y'(t_i)$ va être utilisée au point t_i pour évaluer le coefficient directeur de la tangente à la courbe représentative de $f(t)$ pour l'intervalle $[t_i ; t_{i+1}]$. La fonction $f(t_i, y(t_i))$ permettra de connaître la tendance de variation de la fonction $f(t)$ au niveau du point t_i . La solution numérique de l'équation différentielle sera constituée des tangentes aux points t_i pour chaque intervalle $[t_i ; t_{i+1}]$.

La Figure 748 illustre les 3 premières itérations de la méthode d'Euler. Les différentes valeurs des y_i permettront de reconstituer la solution numérique.



- L'intervalle d'étude $[a ; b]$ est discréterisé en N intervalles de longueur identique h .
- Sur l'intervalle $[x_0 ; x_1]$, l'équation de la tangente au point t_0 sera donné par :

$$T_0(t) = y'(t_0)(t - t_0) + y(t_0) = f(t_0, y_0)(t - t_0) + y_0 \quad \text{en exploitant } y'(t) = f(t, y(t))$$

La valeur de y_1 est donnée par $T_0(t_1)$:

$$y_1 = T_0(t_1) = f(t_0, y_0)(t_1 - t_0) + y_0 \quad \text{avec} \quad (t_1 - t_0) = h$$

$$y_1 = h.f(t_0, y_0) + y_0$$

- Sur l'intervalle $[x_i ; x_{i+1}]$, l'équation de la tangente au point t_i sera donné par :

$$T_i(t) = y'(t_i)(t_{i+1} - t_i) + y(t_i) = f(t_i, y_i)(t - t_i) + y_i \quad \text{en exploitant } y'(t) = f(t, y(t))$$

La valeur de y_{i+1} est donnée par $T_i(t_{i+1})$:

$$y_{i+1} = T_i(t_{i+1}) = f(t_i, y_i)(t_{i+1} - t_i) + y_i \quad \text{avec} \quad (t_{i+1} - t_i) = h$$

$$y_{i+1} = h.f(t_i, y_i) + y_i$$

La solution numérique approchée sera alors reconstituée en joignant par des segments les points (t_k, y_k) tels que :

$$\begin{cases} t_{k+1} = t_k + h \\ y_{k+1} = h \cdot f(t_k, y_k) + y_k \end{cases}$$

2. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentés dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Equations_déifferentielles**

Nous prendrons comme exemple l'équation différentielle suivante :

$$\begin{cases} 5y'(t) + y(t) = 10 \\ y(t_0) = 0 \end{cases}$$

La mise sous la forme d'un problème de Cauchy nous permet de déterminer la fonction : $f(t, y(t))$

$$\begin{cases} y'(t) = f(t, y(t)) = \frac{10 - y(t)}{5} \\ y(t_0) = 0 \end{cases} \quad \text{avec} \quad f(t, y(t)) = \frac{10 - y(t)}{5}$$

L'expression mathématique de cette fonction sera définie dans la fonction **F_1.m** (Figure 749). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du premier ordre.

```
%% F_1.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_1.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle en utilisant la méthode d'Euler
% cette fonction correspond à l'équation différentielle suivante:
% 5(dy/dt) + y = 10
% dy/dt = F(t,y) = (10 - y) / 5

function f = F_1(t,y);
f = (10 - y) / 5;
end
```

Figure 749 : codage de la fonction F1_m qui contient l'expression de la fonction f(t,y)

Pour calculer les solutions de cette équation différentielle, nous allons créer la fonction **fEuler(f,a,b,CI,nb_points)** (Figure 750).

Cette fonction prendra en arguments :

- f** : fonction contenant l'expression de $f(t,y)$
- a** : borne inférieure de l'intervalle d'étude
- b** : borne supérieure de l'intervalle d'étude
- CI** : condition initiale de l'équation différentielle
- nb_points** : nombre de points à calculer

La fonction aura comme sorties :

t: vecteur contenant les instants correspondant aux points où les valeurs de la solution sont calculées
y: vecteur contenant les valeurs de la fonction solution de l'équation différentielle

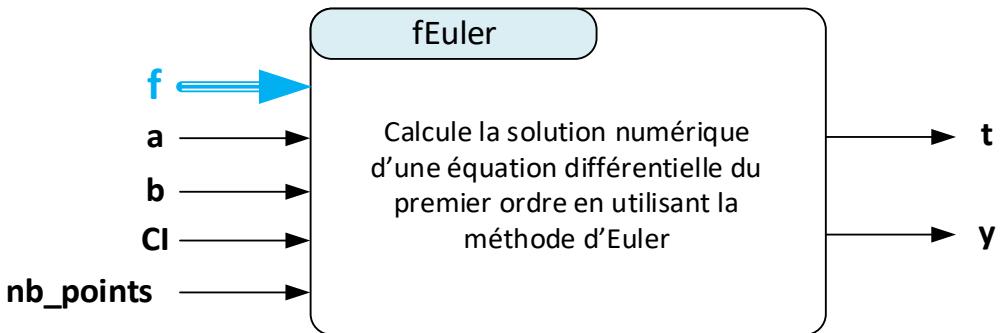


Figure 750 : fonction fEuler (f, a, b, CI, nb_points)

La Figure 751 montre le codage de la fonction **fEuler** (f,a,b,CI,nb_points).

```
%%fEuler
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle du premier ordre mise sous la forme
d'un problème
% de Cauchy
% Equation scalaire: dy/dt = F(t,y)

function [t,y] = fEuler(f,a,b,CI,nb_points)
% calcul du pas h en fonction de a,b et nb_points
h = (b-a)/nb_points;

% définition du vecteur t
t = [a:h:b];

y(1) = CI;
for k = 1:1:nb_points
    y(k+1) = y(k) + h * f(t(k),y(k));
end;
```

Figure 751 : codage de la fonction fEuler(f, a, b, CI, nb_points)

Afin de procéder à la résolution de l'équation différentielle, ouvrir le script **Euler.m**.

Le script est structuré en deux sections (Figure 752) :

- Définition de l'intervalle d'étude de la fonction **[a ;b]** et du nombre de points **nb_points** à calculer pour déterminer la solution numérique de l'équation différentielle.
- Appel de la fonction **fEuler**, puis tracé de la solution numérique dans une fenêtre graphique

```
%% Euler
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle du premier ordre en utilisant
% la méthode d'Euler
%
% la fonction F(y,t) est stockée dans la fonction F_1.m
%% définition de l'intervalle d'étude [a;b] de la fonction considérée
```

```

a = 0;
b = 30;

%spécification du nombre de points
nb_points = 10;

%% Résolution d'une équation différentielle du premier ordre

% spécification de la condition initiale
y0 = 0;
% Résolution avec fEuler
[t,y] = fEuler(@F_1,a,b,y0,nb_points);

figure; hold all
plot(t,y,'LineWidth',2);
grid on; grid minor;
title({'Solution numérique d''une équation différentielle' ; 'du premier ordre' ;
(méthode d''Euler)'})
legend(string(nb_points) + ' points sont calculés ')
axis([0 30 0 12]);

```

Figure 752 : résolution d'une équation différentielle du premier ordre

Exécuter le script avec un nombre de points très faible (nb_points = 10) afin de visualiser le processus de résolution et d'identifier les différents segments qui constituent la solution numérique.

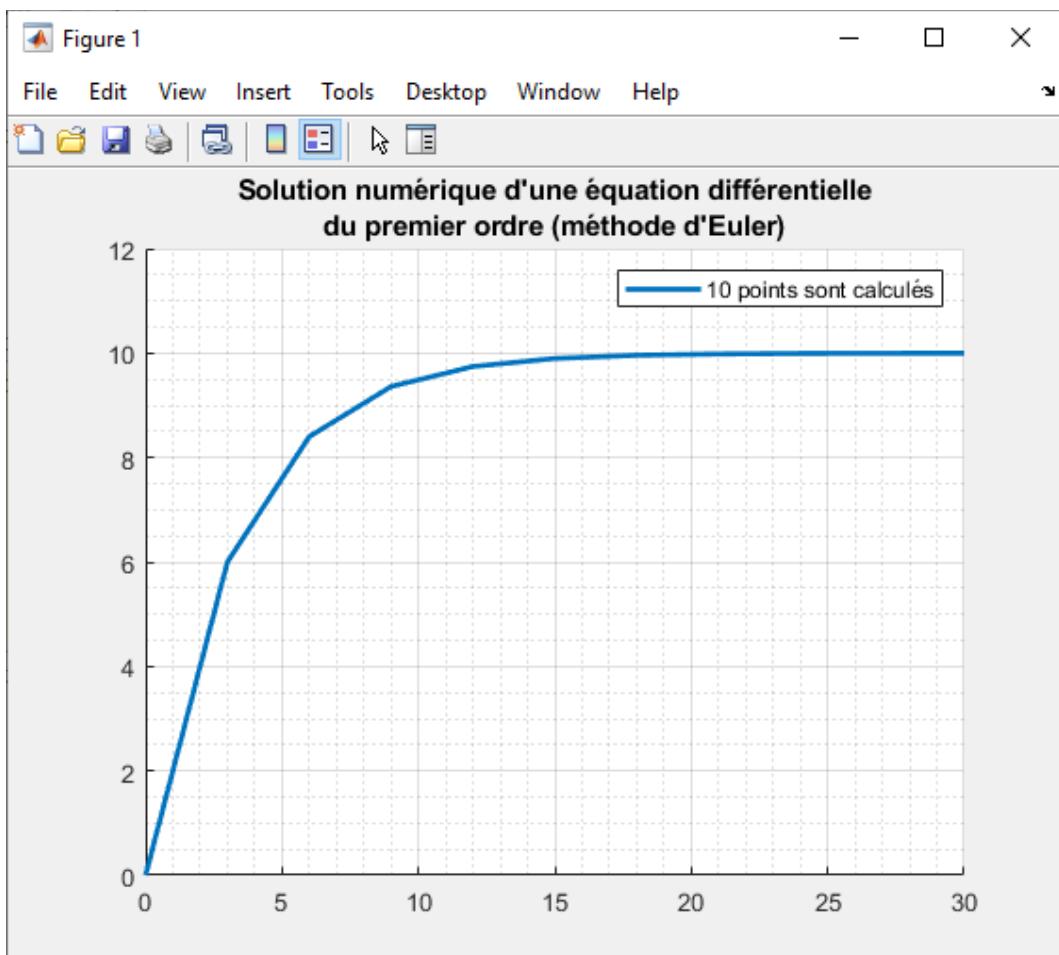


Figure 753 : solution numérique calculée avec 10 points

Exécuter à nouveau le script avec un nombre de points plus important (nb_points = 100)

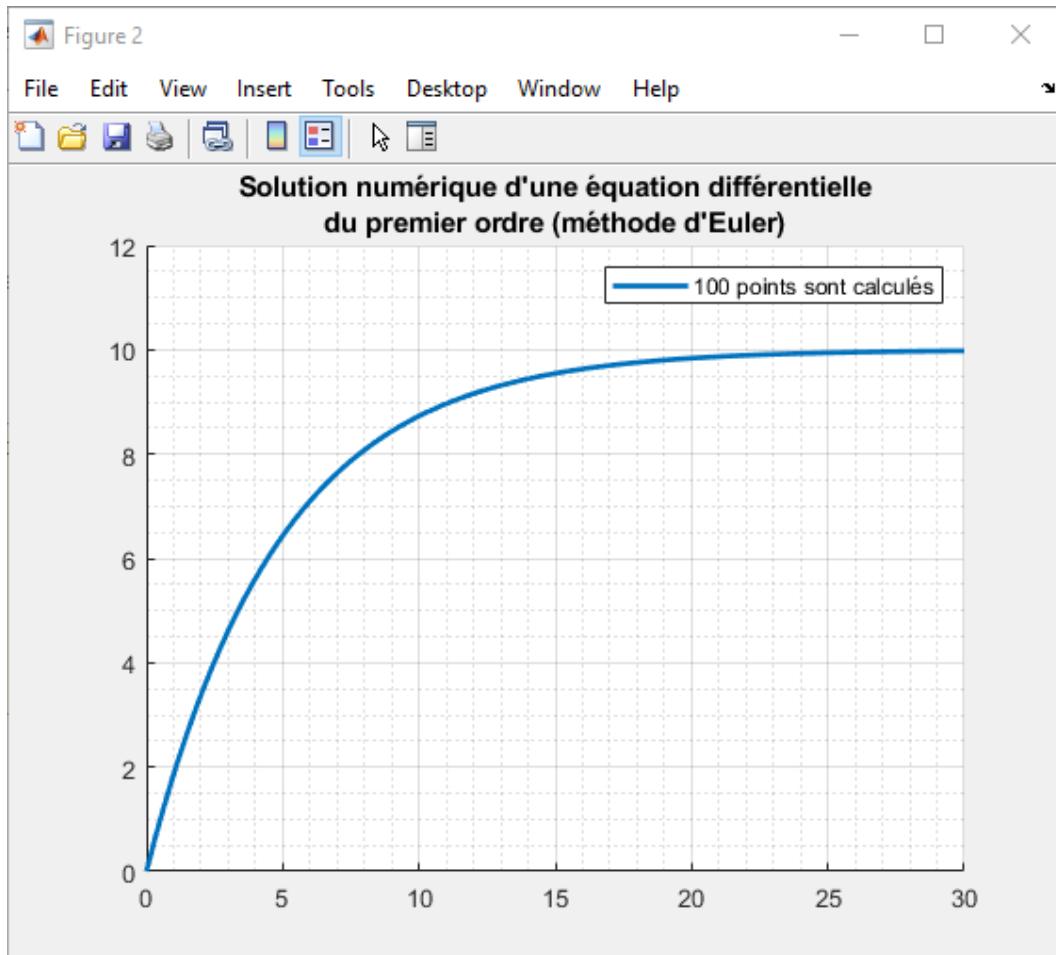


Figure 754 : solution numérique calculée avec 100 points

B. Résolution d'une équation différentielle d'ordre n – Méthode d'Euler

1. Aspects théoriques

La méthode d'Euler peut être généralisée et utilisée pour résoudre des équations différentielles d'ordre n en utilisant la vectorisation. Le principe consiste à transformer une équation différentielle d'ordre n en un système de n équations différentielles d'ordre 1.

Exemple : Considérons l'équation différentielle suivante :

$$y''(t) + a y'(t) + b y(t) = g(t)$$

Avec pour conditions initiales : $y(t_0) = C_0$ et $y'(t_0) = C_1$

Posons : $\begin{cases} y_1(t) = y(t) \\ y_2(t) = y'(t) \end{cases}$

L'équation différentielle peut alors s'écrire en fonction de $y_1(t)$ et $y_2(t)$:

$$y'_2(t) + a y_2(t) + b y_1(t) = g(t)$$

Il est alors possible de se ramener au système suivant contenant deux équations différentielles d'ordre 1 :

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = g(t) - a y_2(t) - b y_1(t) = \varphi(t, y_1(t), y_2(t)) \end{cases}$$

En considérant les vecteurs $Y(t) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, $Y'(t) = \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix}$ et $F(t, Y(t)) = \begin{pmatrix} y_2 \\ \varphi(t, y_1(t), y_2(t)) \end{pmatrix}$

Le système peut alors s'écrire sous forme vectorielle :

$$\boxed{Y'(t) = F(t, Y(t))}$$

avec

$$\boxed{F(t, Y(t)) = \begin{pmatrix} y_2 \\ g(t) - a y_2(t) - b y_1(t) \end{pmatrix}}$$

Le problème se ramène à un problème de Cauchy avec une équation différentielle vectorialisée de dimension 2 et sa condition initiale Y_0 de dimension 2 également.

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y_0 = \begin{bmatrix} y'_1(t_0) = C_0 \\ y'_2(t_0) = C_1 \end{bmatrix} \end{cases}$$

Ce raisonnement peut se généraliser à l'ordre n :

En considérant une équation différentielle d'ordre n mise sous la forme suivante :

$$y^{(n)}(t) = \varphi(t, y(t), y'(t), \dots, y^{(n-1)}(t))$$

Posons :

$$\begin{cases} y_1(t) = y(t) \\ y_2(t) = y'(t) \\ y_3(t) = y^{(2)}(t) \\ \vdots \\ y_n(t) = y^{(n-1)}(t) \end{cases}$$

L'équation différentielle peut s'écrire comme un système de n équations différentielles d'ordre 1 :

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = y_3(t) \\ \vdots \\ y'_{n-1}(t) = y_n(t) \\ y'_n(t) = \varphi(t, y(t), y'(t), \dots, y^{(n-1)}(t)) \end{cases}$$

En considérant les vecteurs :

$$Y(t) = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad Y'(t) = \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_n \end{pmatrix} \quad \text{et} \quad F(t, Y(t)) = \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ y_n \\ \varphi(t, y(t), y'(t), \dots, y^{(n-1)}(t)) \end{pmatrix}$$

Le système peut s'écrire sous la forme vectorielle suivante :

$$Y'(t) = F(t, Y(t))$$

Le problème se ramène à un problème de Cauchy avec une équation différentielle vectorialisée de dimension n et sa condition initiale Y_0 de dimension n également.

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y_0 = \begin{bmatrix} C_0 \\ \vdots \\ C_n \end{bmatrix} \end{cases}$$

La méthode d'Euler permettant de résoudre une équation différentielle d'ordre 1 peut alors s'appliquer. Les grandeurs manipulées par les algorithmes ne seront plus de dimension 1 mais deviendront des vecteurs de dimension n. Ce formalisme permettra de coder très simplement un algorithme de résolution d'équations différentielles d'ordre n. C'est également ce formalisme qui est utilisé par tous les solveurs de MATLAB qui permettent de résoudre les équations différentielles (ode45, ode23...). Il est donc indispensable pour un utilisateur de prendre en main ce formalisme.

2. Codage en langage MATLAB

Pour calculer les solutions d'une équation différentielle d'ordre n, nous allons créer la fonction **my_ode** (*f,a,b,CI,nb_points*) (Figure 755).

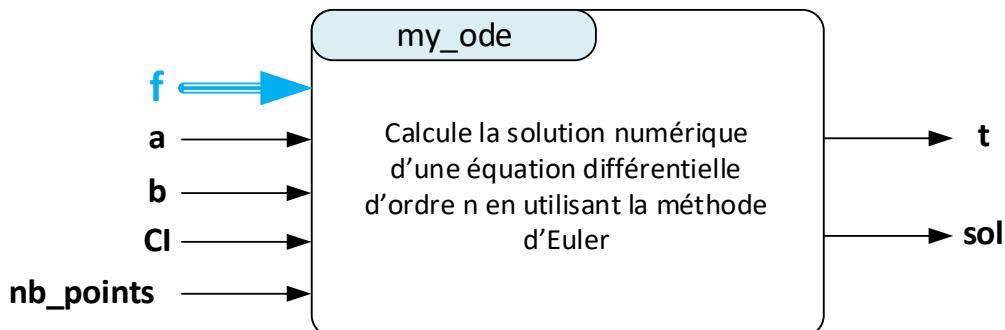


Figure 755 : fonction **my_ode** (*f, a, b, CI, nb_points*)

Cette fonction prendra en arguments :

f : fonction contenant l'expression de $f(t,y)$. Cette fonction sera un vecteur colonne de dimension n.

a : borne inférieure de l'intervalle d'étude

b : borne supérieure de l'intervalle d'étude

CI : condition initiale de l'équation différentielle. La condition initiale sera un vecteur colonne de dimension n.

nb_points : nombre de points à calculer

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points où les valeurs de la solution sont calculées

sol : matrice contenant les valeurs de toutes les fonctions y_i définies dans la méthode. La structure de cette matrice est donnée sur la Figure 756. La première colonne de cette matrice regroupe les valeurs calculées de la fonction solution recherchée $y(t)$.

$$sol = \begin{bmatrix} y & y' & y^{(n)} \\ y_1(t_0) & y_2(t_0) & \dots & y_n(t_0) \\ y_1(t_1) & y_2(t_1) & \dots & y_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(t_N) & y_2(t_N) & \dots & y_n(t_N) \end{bmatrix}$$

Figure 756 : structure de la matrice sol

La Figure 757 montre le codage de la fonction **my_ode** (f,a,b,CI,nb_points).

```
%% my_ode
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle mise sous la forme d'un problème
% de Cauchy
% Equation vectorielle: dy/dt = F(t,y)

function [t,sol] = my_ode_colonne(f,a,b,CI,nb_points)
h = (b-a)/nb_points;
t = [a:h:b];

% La première ligne de la matrice solution est constituée du vecteur
% condition initiales CI
sol(1,:) = CI;

% Chaque ligne de la matrice est construite en utilisant le schéma d'Euler
% Il est nécessaire de transposer l'expression "h * f(t(k),sol(k,:))" afin
% de la rendre de même dimension que les autres termes de l'équation
for k = 1:1:nb_points
    sol(k+1,:) = sol(k,:) + h * f(t(k),sol(k,:))';
end
end
```

Figure 757 : codage de la fonction my_ode (f,a,b,CI,nb_points)

Afin de tester notre solveur **my_ode**, nous allons utiliser un script qui va rechercher la solution d'une équation différentielle du second ordre de trois manières différentes :

- En utilisant **my_ode**
- En utilisant **ode23** (solveur intégré à MATLAB)
- En recherchant la solution analytique en calcul symbolique

L'équation sera la suivante :

$$\begin{cases} 5y''(t) + y'(t) + y(t) = 2 \\ y(0) = 0 \text{ et } y'(0) = 0 \end{cases}$$

Que l'on peut mettre sous la forme :

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = \frac{2 - y_2(t) - y_1(t)}{5} \Rightarrow F(t, Y) = \begin{bmatrix} y_2(t) \\ \frac{2 - y_2(t) - y_1(t)}{5} \end{bmatrix} \\ Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases}$$

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = \frac{2 - y_2(t) - y_1(t)}{5} \\ Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases}$$

$$Y'(t) = F(t, Y(t)) \text{ avec } F(t, Y) = \begin{bmatrix} y_2(t) \\ \frac{2 - y_2(t) - y_1(t)}{5} \end{bmatrix} \text{ et } Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

L'expression mathématique de cette équation différentielle sera définie dans la fonction **F_2.m** (Figure 758). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du second ordre. Dans cette fonction la sortie **F_2** calculée sera une matrice (2,1).

```
%% F_2.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_2.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle du second ordre en utilisant la méthode d'Euler
% cette fonction correspond à l'équation différentielle suivante:
% 5.d2y/dt2 + dy/dt + y = 2
% f(t,y) est une matrice 2-by-1 tel que :
% f(1) = y(2)
% f(2) = 2-y(2)-y(1))/5

function f = F_2(t,y);
f = [y(2) ; (2-y(2)-y(1))/5];
end
```

Figure 758 : codage de la fonction F_2.m qui contient l'expression de la fonction f(t,y)

Ouvrir le script num_ode_comparaison.m (Figure 759).

Ce script va permettre de tracer dans une même fenêtre graphique la solution calculée avec **my_ode**, **ode23** et la **solution analytique**.

Il est possible de définir en début de script l'intervalle de calcul [a ; b] et le nombre de points **nb_points** utilisés pour le calcul.

```
%% num_ode_comparaison
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my_ode et ode_23 pour la résolution d'une équation
% différentielle du deuxième ordre
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;

% spécification du nombre de points et du pas h
nb_points = 100;
h= (b-a) / nb_points;
```

```

%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques

% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [0;0];

% Résolution avec my_ode
[t_my_ode_2,sol_my_ode_2] = my_ode(@F_2,a,b,CI,nb_points);

% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécessaire au calcul (tspan). Dans my_ode, il fallait rentrer [a;b] et le
% nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t_ode23_2,sol_ode23_2] = ode23(@F_2,tspan,CI);

%% Résolution analytique en calcul symbolique
%définition des variables et de la fonction symbolique
syms t;
syms y(t);

% définition des dérivées symboliques successives de f(t)
D1y=diff(y,1);
D2y=diff(y,2);

% Construction de l'équation à résoudre et résolution à l'aide de la
% fonction dsolve et spécification des conditions initiales
equ=5*D2y+D1y+y==2;
sol_symbolique(t)=dsolve(equ,y(0)==CI(1),D1y(0)==CI(2));

% calcul des valeurs du vecteur sol_ana qui contient les valeurs de la
% solution calculée analytiquement

t_ana = [a:h:b];
sol_ana = sol_symbolique(t_ana);

% conversion de la fonction symbolique en double précision
sol_ana = double(sol_ana);

%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correccpond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode_2,sol_my_ode_2(:,1),'b','LineWidth',2);
plot(t_ode23_2,sol_ode23_2(:,1),'k--','LineWidth',2);
plot(t_ana,sol_ana,'r');
grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
    'd''une équation différentielle du second ordre',string(nb_points) + ' points
    sont calculés '} );

legend('Solution calculée avec my_ode','Solution calculée avec ode23',...
    'Solution calculée analytiquement');

```

Figure 759 : comparaison des différentes méthodes de résolution d'une équation différentielle

Exécuter le script avec un nombre de points calculés très faible (`nb_points = 100`) et observer le résultat

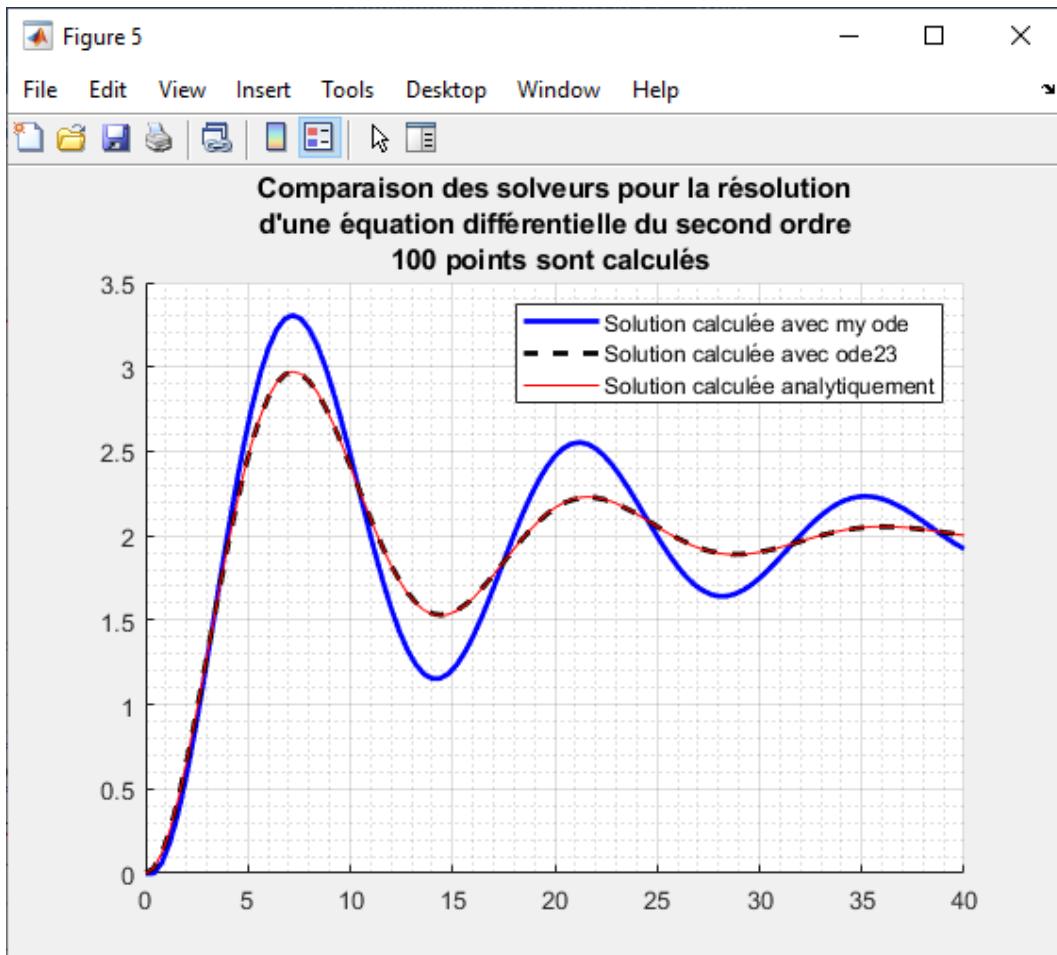


Figure 760 : comparaison des différentes méthodes de résolution d'une équation différentielle (100 points calculés)

Il apparaît que notre solveur **my_ode** nous donne une approximation de la solution avec une erreur très importante. Le solveur **ode23** est très proche de la solution analytique exacte et s'avère bien plus performant.

La simplicité du codage que nous avons adopté pour coder **my_ode** (6 lignes de code) le rend très sensible au nombre de points retenus pour le calcul. En toute logique et conformément à la méthode utilisée pour coder notre solveur la diminution du pas et l'augmentation du nombre de points calculés devrait améliorer considérablement sa précision.

Relancer le script en choisissant un nombre de point plus important (`nb_points = 1000`).

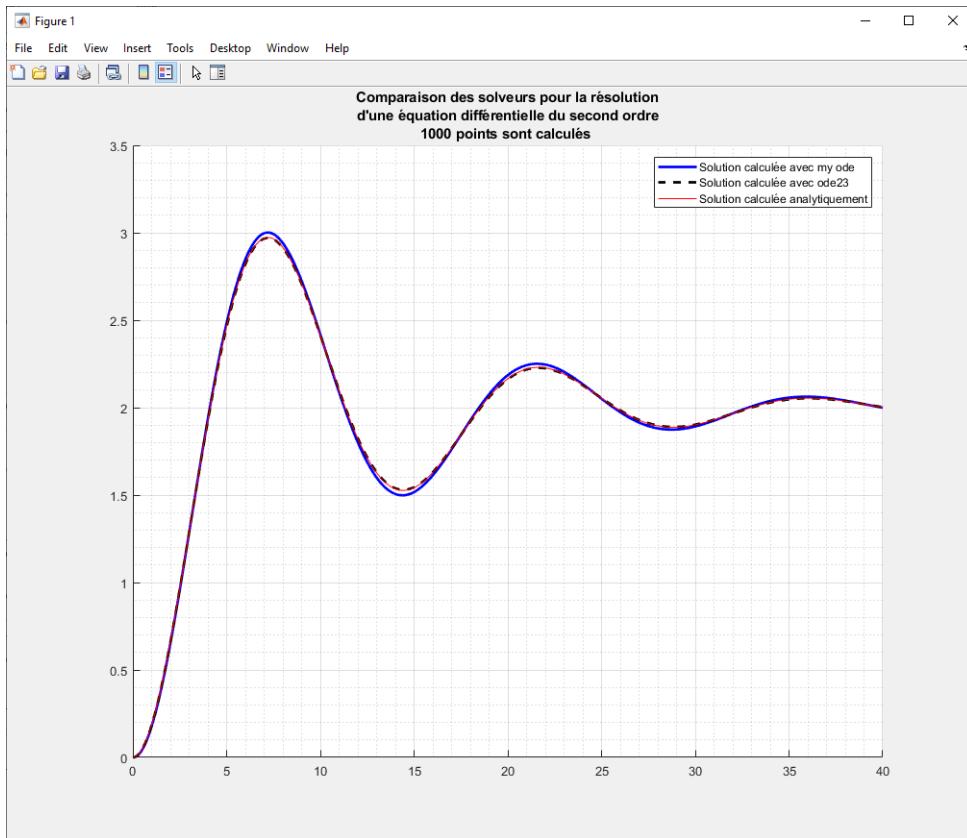


Figure 761 : comparaison des différentes méthodes de résolution d'une équation différentielle (1000 points calculés)

Nous pouvons constater que le solveur **my_ode** nous fournit une solution acceptable avec 1000 points calculés.

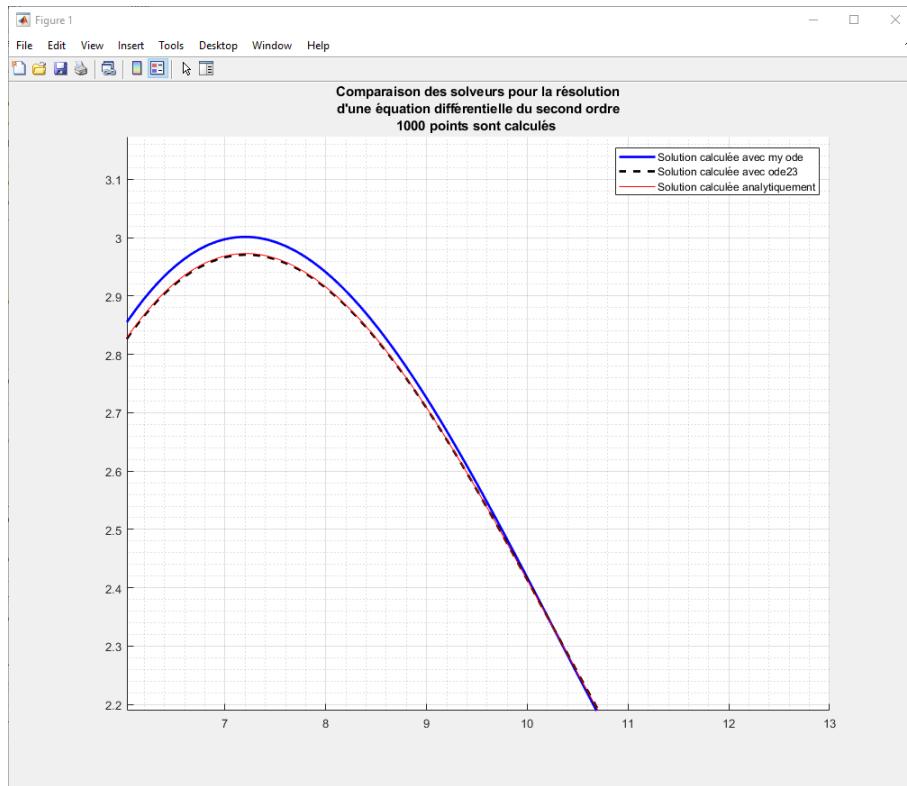


Figure 762 : zoom sur la courbe

A partir de 10000 points notre solveur donne le même ordre de grandeur de précision que **ode23**. Le temps de calcul est par contre considérablement augmenté.

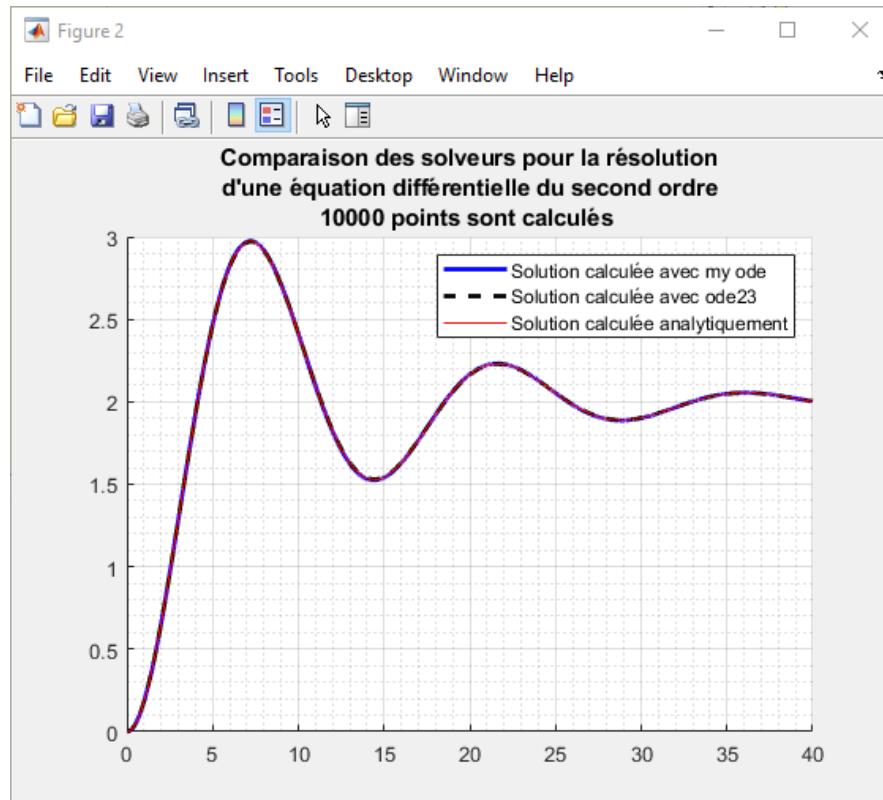


Figure 763 : comparaison des différentes méthodes de résolution d'une équation différentielle(10000 points calculés)

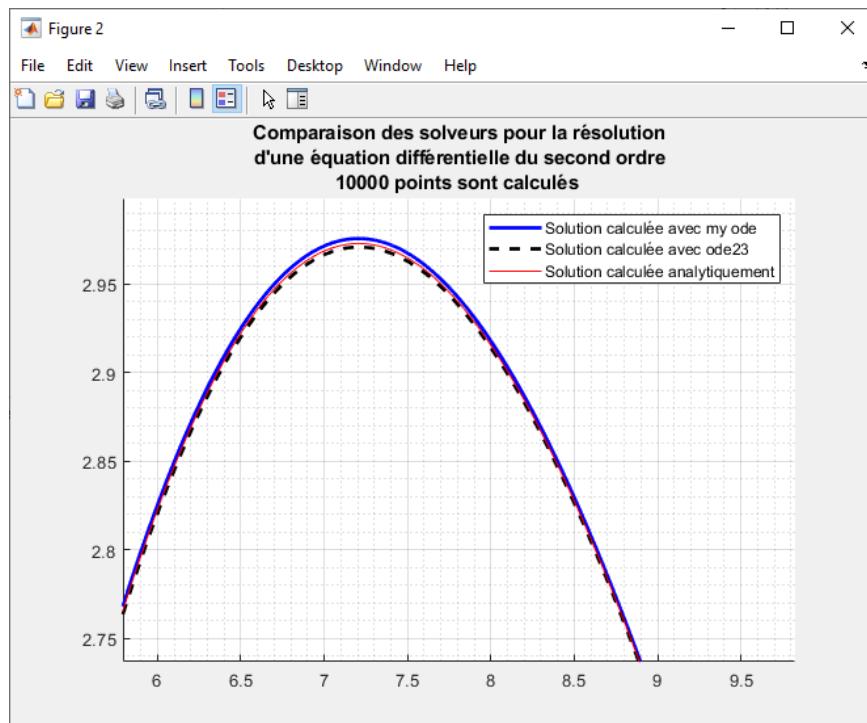


Figure 764 : zoom sur la courbe

3. Robustesse du processus. Résolution d'une équation différentielle d'ordre 3

Afin de tester notre solveur sur une équation différentielle d'ordre supérieur à 2, nous allons considérer l'équation différentielle du troisième ordre suivante :

$$\begin{cases} y'''(t) + 3y''(t) + 3y'(t) + y(t) = 10 \\ y(0) = 1, y'(0) = 0 \text{ et } y''(0) = 3 \end{cases}$$

Que l'on peut mettre sous la forme :

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = y_3(t) \\ y'_3(t) = 10 - 3y_3(t) - 3y_2(t) - y_1(t) \end{cases} \Rightarrow F(t, Y) = \begin{bmatrix} y_2(t) \\ y_2(t) \\ 10 - 3y_3(t) - 3y_2(t) - y_1(t) \end{bmatrix}$$

$$Y_0 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = y_3(t) \\ y'_3(t) = 10 - 3y_3(t) - 3y_2(t) - y_1(t) \end{cases}$$

$$Y_0 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$Y'(t) = F(t, Y(t)) \text{ avec } F(t, Y) = \begin{bmatrix} y_2(t) \\ y_2(t) \\ 10 - 3y_3(t) - 3y_2(t) - y_1(t) \end{bmatrix} \text{ et } Y_0 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

L'expression mathématique de cette équation différentielle sera définie dans la fonction **F_3.m** (Figure 765). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du second ordre. Dans cette fonction la sortie **F_3** calculée sera une matrice (3,1).

```

%% F_3.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_2.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle du second ordre en utilisant la méthode d'Euler
% cette fonction correspond à l'équation différentielle suivante:
% d3y/dt3 + 3.d2y/dt2 + 3dy/dt + y = 10
% f(t,y) est une matrice 3-by-1 tel que :
% f(1) = y(2)
% f(2) = y(3)
% f(3) = 10 - 3*y(3) - 3*y(2) - y(1))/5

function f = F_3(t,y);
f = [y(2) ; y(3) ; (10 - 3*y(3) - 3*y(2) - y(1))/5];
end

```

Figure 765 : codage de la fonction F_3.m

Ouvrir le script **solve_ode_3.m** (Figure 766).

Ce script va permettre de tracer dans une même fenêtre graphique la solution de cette équation différentielle du troisième ordre calculée avec **my_ode** et avec **ode23**. Il n'y a aucune modification à faire sur la fonction **my_ode** qui peut traiter des matrices de toute dimension.

Il est possible de définir en début de script l'intervalle de calcul **[a ; b]** et le nombre de points **nb_points** utilisés pour le calcul.

```

%% solve_ode_3
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my_ode et ode_23 pour la résolution d'une équation
% différentielle du troisième ordre
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;

% spécification du nombre de points et du pas h
nb_points = 4000;
h= (b-a) / nb_points;

%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques

% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [1;0;3];

% Résolution avec my_ode
[t_my_ode_3,sol_my_ode_3] = my_ode(@F_3,a,b,CI,nb_points);

% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécessaire au calcul (tspan). Dans my_ode, il fallait rentrer [a;b] et le
% nombre de points ce qui revient exactement au même
tspan = [a:h:b];
[t_ode23_3,sol_ode23_3] = ode23(@F_3,tspan,CI);

%représentation graphique de la solution

```

```

figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correspond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode_3,sol_my_ode_3(:,1),'b','LineWidth',2);
plot(t_ode23_3,sol_ode23_3(:,1),'k--','LineWidth',2);

grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
    'd''une équation différentielle du troisième ordre',string(nb_points) + ...
    'points sont calculés '});

legend('Solution calculée avec my_ode','Solution calculée avec ode23');

```

Figure 766 : résolution d'une équation différentielle d'ordre 3 avec my_ode et ode23

Exécuter le script et observer le résultat sur la Figure 767.

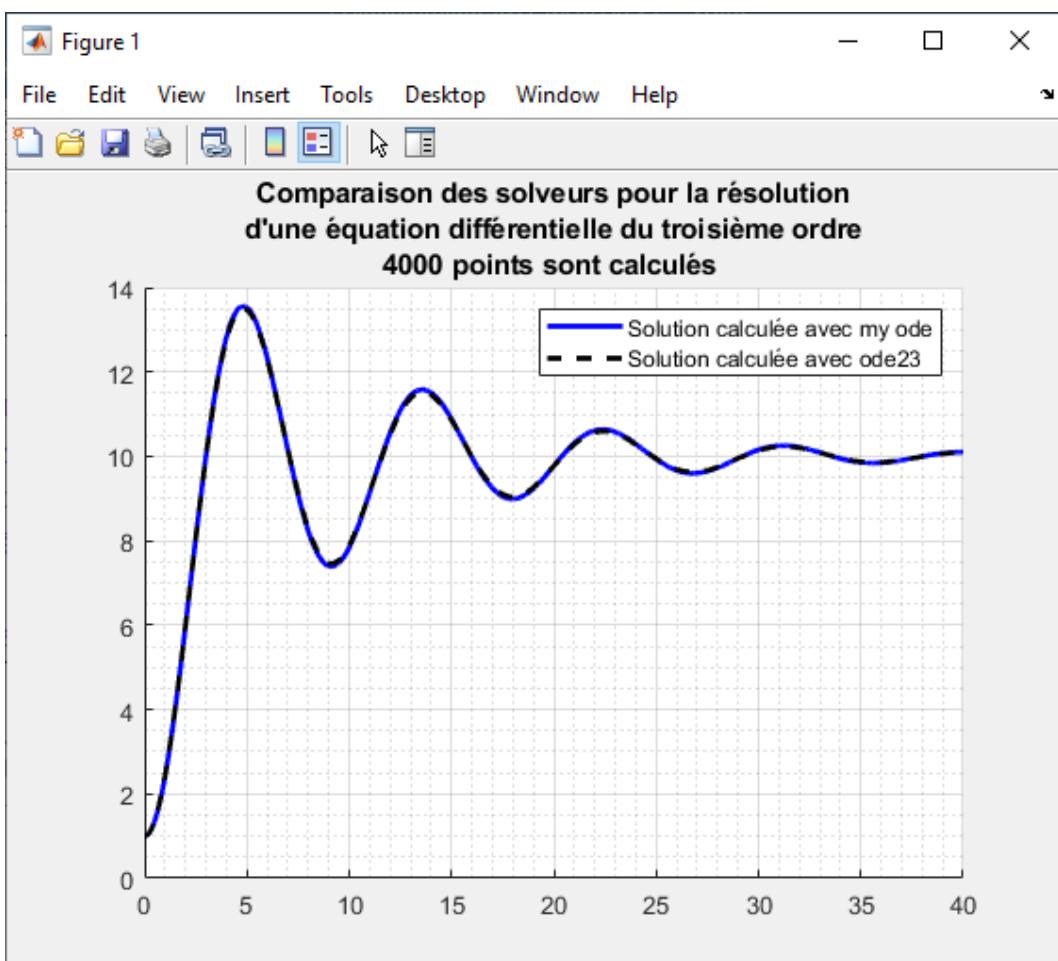


Figure 767 : solution d'une équation différentielle du troisième ordre

Le calcul étant réalisé avec 4000 points, le résultat obtenu avec **my_ode** est satisfaisant.

4. Robustesse du processus. Application à la résolution de l'équation de Van der Pol

La documentation de la fonction **ode23** de MATLAB propose la résolution de l'équation différentielle de Van der Pol.

$$y_1''(t) - (1 - y_1^2(t)) + y_1(t) = 0$$

Ouvrir le script **solve_ode_Van_der_Pol.m** permet de tester la résolution de cette équation à l'aide de **my_ode** et de **ode23**. La fonction F(t,y) correspondant à l'équation de Van der Pol est définie en fin de script.

```
%% solve_ode_Van_der_Pol
% Ivan LIEBGOTT © Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my_ode et ode_23 pour la résolution de l'équation
% différentielle de Van der Pol (@ Mathworks documentation)
% La fonction qui définit la fonction F(t,y) est définie en fin de script

% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;

% spécification du nombre de points et du pas h
nb_points = 10000;
h= (b-a) / nb_points;

%% Résolution de l'équation différentielle de Van der Pol

% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [2;0];

% Résolution avec my_ode
[t_my_ode_vdp,sol_my_ode_vdp] = my_ode(@vdp,a,b,CI,nb_points);

% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécessaire au calcul (tspan). Dans my_ode, il fallait rentrer [a;b] et le
% nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t_ode23_vdp,sol_ode23_vdp] = ode23(@vdp,[a b],CI);

%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correspond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode_vdp,sol_my_ode_vdp(:,1), 'c', 'LineWidth', 2);
plot(t_ode23_vdp,sol_ode23_vdp(:,1), 'k--', 'LineWidth', 2);

grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
    'd''une équation différentielle du troisième ordre',string(nb_points) + ...
    'points sont calculés '});

legend('Solution calculée avec my_ode','Solution calculée avec ode23');

function f = vdp(t,y)
f = [y(2); (1-y(1)^2)*y(2)-y(1)];
end
```

Figure 768 : résolution de l'équation différentielle de Van der Pol

Exécuter le script et observer le résultat sur la Figure 769.

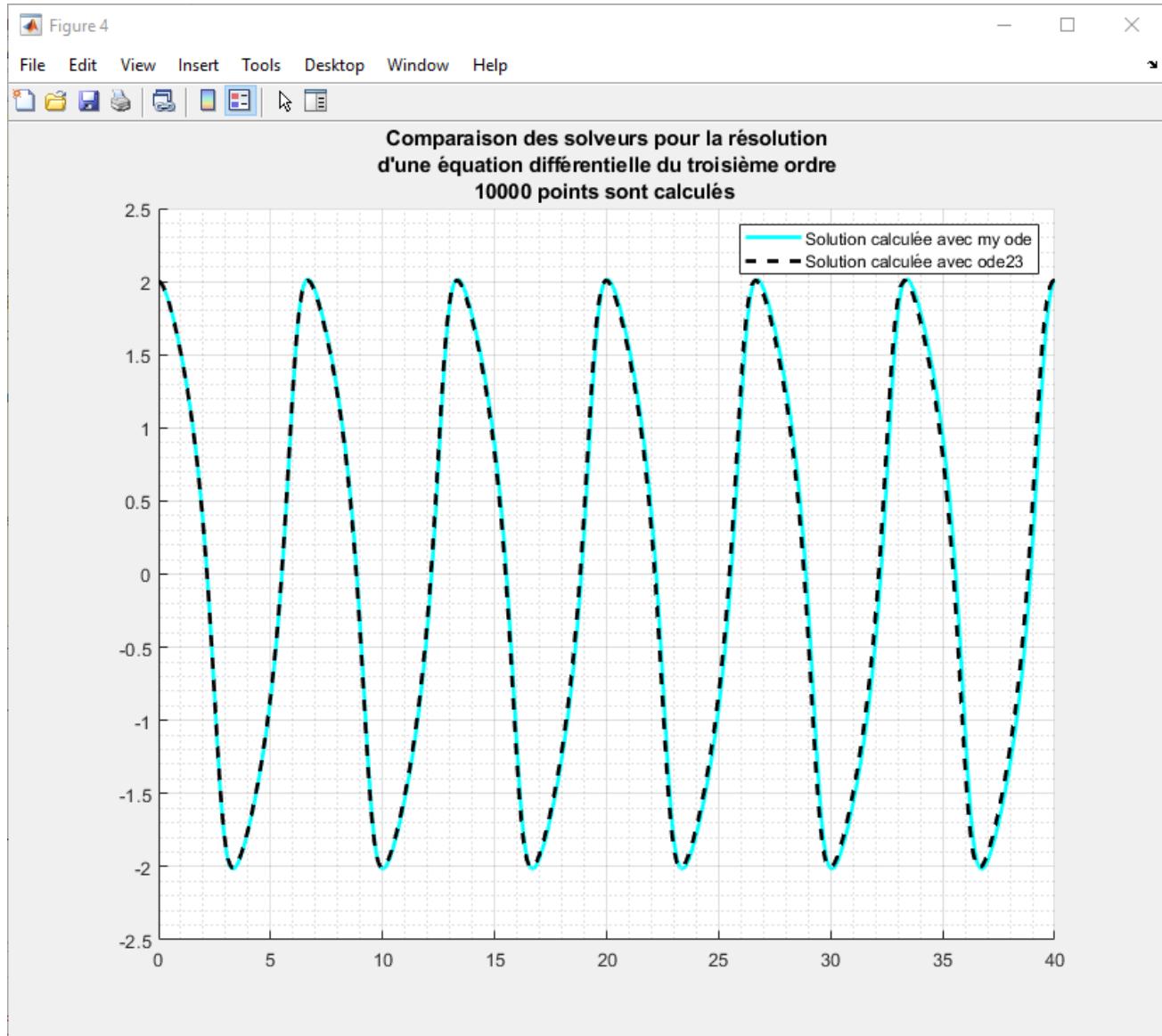


Figure 769 : solution de l'équation différentielle de Van der Pol, 10000 points calculés

La prise en compte d'un grand nombre de points (10000) garantit pour cet exemple la précision de la résolution pour notre solveur **my_ode**.

Vous pouvez donc utiliser le solveur **my_ode** pour résoudre toutes les équations différentielles d'ordre quelconques, linéaires et non linéaires à conditions de choisir un nombre important de points de calcul.

C. Applications

1. Modélisation d'un oscillateur mécanique

La modélisation d'un problème se ramène très souvent à la résolution d'une équation différentielle. Prenons l'exemple de l'oscillateur mécanique de la Figure 770. Le mouvement est déclenché par un écartement de la masse par rapport à sa position d'équilibre de $\Delta x=0.1$ m. La masse évolue ensuite librement.

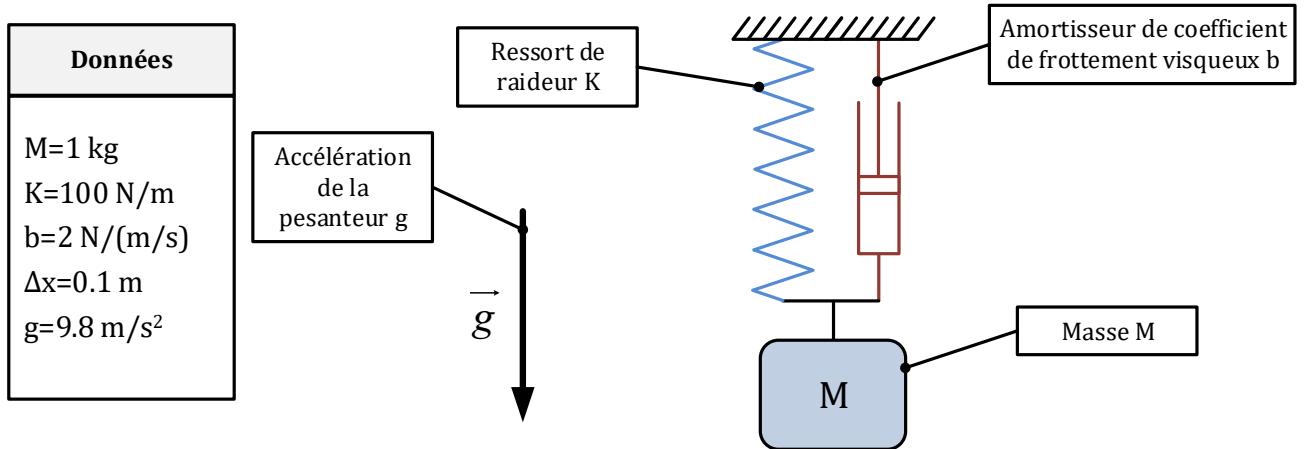


Figure 770 : oscillateur mécanique avec prise en compte de la pesanteur

En isolant la masse et en appliquant le principe fondamental de la dynamique, on obtient l'équation différentielle qui caractérise l'évolution de la position $x(t)$. en fonction du temps.

$$M v'(t) + b v(t) + K x = M g$$

En exploitant le fait que $v = \frac{dx}{dt}$, l'équation différentielle se ramène une équation différentielle du second ordre en $x(t)$. :

$$M x''(t) + b x'(t) + K x = M g$$

Les conditions initiales sont données par : $\begin{cases} x(0) = 0.1 \\ x'(0) = 0 \end{cases}$

2. Les différentes méthodes de résolution d'une équation différentielle avec MATLAB

Nous allons les différentes méthodes que nous propose MATLAB pour résoudre ce type d'équation et trouver l'évolution de $x(t)$. en fonction du temps.

- Résolution en utilisant notre solveur **my_ode** (code MATLAB)
- Résolution en utilisant **ode23** (code MATLAB)
- Résolution en utilisant le calcul symbolique (code MATLAB)
- Résolution à l'aide de Simulink en construisant un modèle Simulink de l'équation différentielle
- Résolution avec Simscape en construisant le modèle multi-phérique de système masse ressort

Toutes ces méthodes vont donner des résultats satisfaisants.

3. Résolution avec my_ode, ode 23 et en calcul symbolique

L'équation différentielle du second ordre peut se mettre sous la forme d'un système d'équations différentielles du premier ordre.

$$\begin{cases} y'_1 = y_2 \\ y'_2 = \frac{1}{M}(M \cdot g - b \cdot y_2 - K \cdot y_1) \end{cases}$$

Pour utiliser les solveurs pour résoudre cette équation différentielle il faut commencer par créer la fonction qui va contenir l'équation différentielle. Le codage de la fonction **F_mass_spring_damper.m** est donné sur la Figure 771.

```
%% F_mass_spring_damper.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_mass_spring_damper.m contient l'expression de la fonction F(t,y) utilisée pour
résoudre
% l'équation différentielle du second ordre correspondant à un oscillateur
% mécanique:
% M.d2y/dt2 + b.dy/dt + K.y = M.g
% F(t,y) est une matrice 2-by-1 tel que :
% f(1) = y(2)
% f(2) = M.g-b.y(2)-K.y(1))/M

function f = F_mass_spring_damper(t,y);

% définition des grandeurs physique qui interviennent dans l'équation
m = 1;      % mass : kg
k = 100;    % spring rate : N/m
b = 2;       % damping coefficient : N/(m/s)
g = 9.81;   % gravity acceleration : N/kg

f = [y(2) ; (m*g-b*y(2)-k*y(1))/m];
end
```

Figure 771 : codage de la fonction F_mass_spring_damper.m

Le script **ode_solve_mass_spring_damper.m** permet de résoudre l'équation différentielle en utilisant le solveur **my_ode**, le solveur **ode23** et le calcul symbolique pour déterminer la solution analytique exacte.

```
%% ode_solve_mass_spring_damper.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Permet la résolution de l'équation différentielle de l'oscillateur
% mécanique en utilisant my_ode et ode23
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 5;

% spécification du nombre de points
nb_points = 5000;
```

```

h= (b-a) / nb_points;

%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques

% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [0.1;0];

% Résolution avec my_ode
[t_my_ode,sol_my_ode] = my_ode(@F_mass_spring_damper,a,b,CI,nb_points);

% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécessaire au calcul (tspan). Dans my_ode, il fallait rentrer [a;b] et le
% nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t_ode23_2,sol_ode23_2] = ode23(@F_mass_spring_damper,tspan,CI);

%% Résolution analytique en calcul symbolique
%définition des variables et de la fonction symbolique
syms t;
syms y(t);

% définition des grandeurs physique qui interviennent dans l'équation
m = 1;      % mass : kg
k = 100;    % spring rate : N/m
b = 2;      % damping coefficient : N/(m/s)
g = 9.81;   % gravity acceleration : N/kg

% définition des dérivées symboliques successives de f(t)
D1y=diff(y,1);
D2y=diff(y,2);

% Construction de l'équation à résoudre et résolution à l'aide de la
% fonction dsolve et spécification des conditions initiales
equ= m*D2y + b*D1y + k*y ==m*g;
sol_symbolique(t)=dsolve(equ,y(0)==CI(1),D1y(0)==CI(2));

% calcul des valeurs du vecteur sol_ana qui contient les valeurs de la
% solution calculée analytiquement

t_ana = [a:h:b];
sol_ana = sol_symbolique(t_ana);

% conversion de la fonction symbolique en double précision
sol_ana = double(sol_ana);

%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correccspond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode,sol_my_ode(:,1),'b','LineWidth',2);
plot(t_ode23_2,sol_ode23_2(:,1),'k--','LineWidth',2);
plot(t_ana,sol_ana,'r');
grid on; grid minor;
title({'Résolution de l''équation différentielle de l''oscillateur mécanique',...
       'en utilisant 3 méthodes différentes',string(nb_points) + ' points sont
       calculés '});

```

```
legend('Solution calculée avec my ode','Solution calculée avec ode23',...
'Solution calculée en symbolique');
```

Figure 772 : Calcul et tracé de la solution de l'équation différentielle en utilisant 3 méthodes de résolution

Ouvrir le script et **exécuter** le.

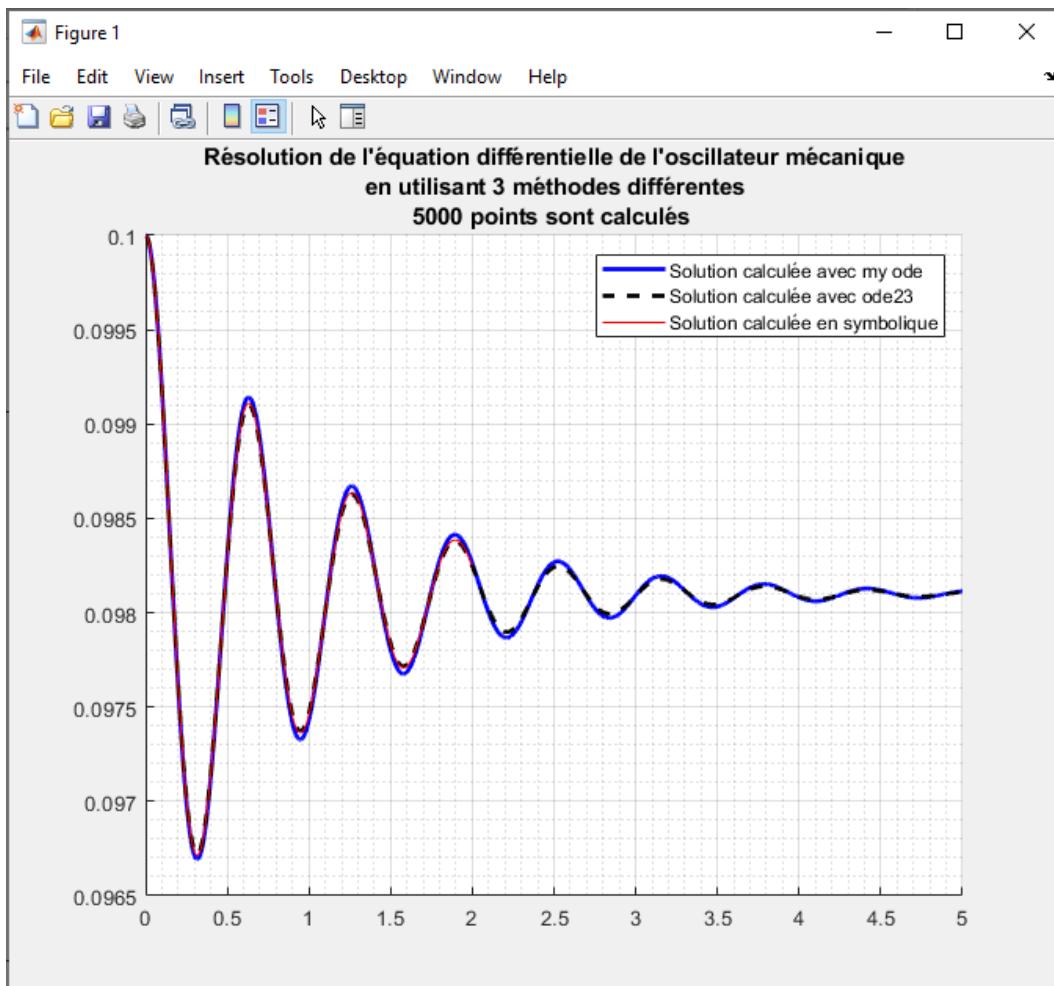


Figure 773 : résolution de l'équation différentielle avec 3 méthodes

Nous pouvons constater que les trois méthodes sont valables et donnent des résultats satisfaisants.

4. Résolution de l'équation différentielle avec Simulink

Pour résoudre une équation différentielle avec Simulink, il suffit de construire un modèle à partir de la forme suivante de l'équation :

$$x''(t) = \frac{1}{M} (M g - b x'(t) - K x)$$

Ouvrir le fichier **simulink_resolution.slx**.

La Figure 774 représente le modèle permettant de résoudre l'équation différentielle de l'oscillateur mécanique en utilisant Simulink.

Condition initial à renseigner dans
l'intégrateur $x(0)=0.1$

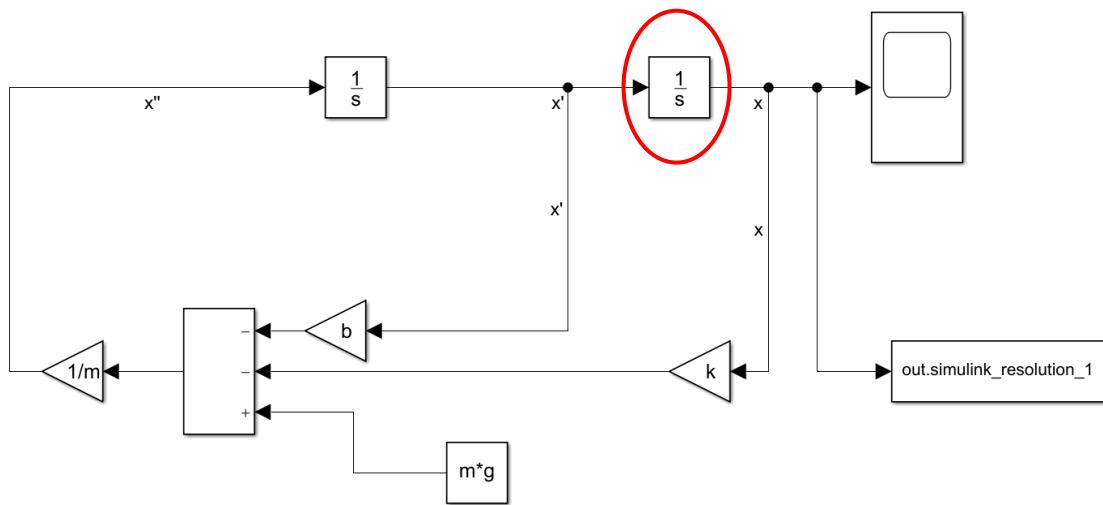


Figure 774 : modèle permettant la résolution de l'équation différentielle avec Simulink

Afin de prendre en compte la condition initiale $x(0)=0.1$, il faut compléter le bloc intégrateur correspondant à la position x comme indiqué sur la Figure 775.

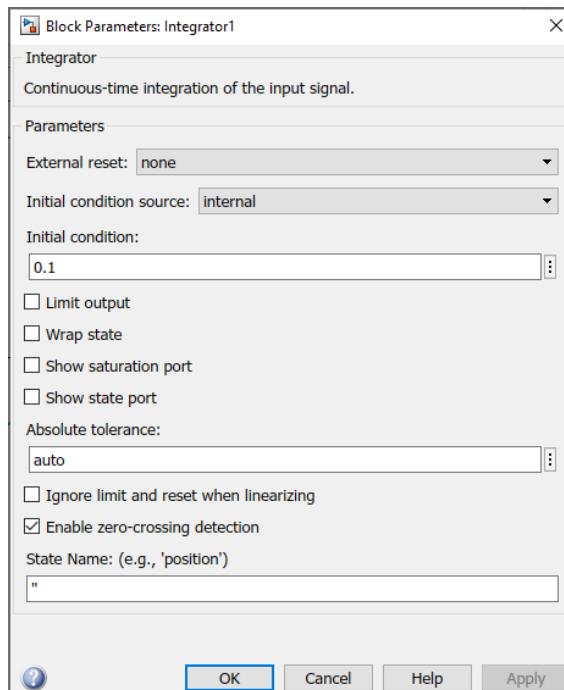


Figure 775 : spécification de la condition initiales $x(0)=0.1$ dans l'intégrateur de la position

Lancer la simulation et observer la réponse dans le scope de la Figure 776.

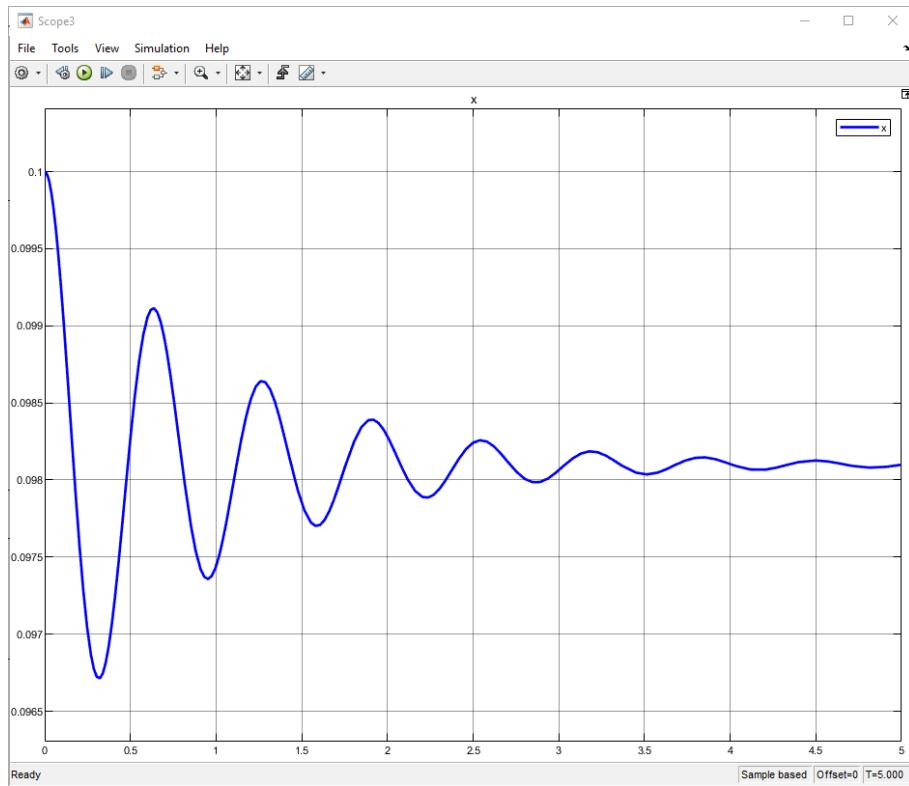


Figure 776 : visualisation de la solution $x(t)$ déterminée avec Simulink

Nous pouvons constater que la réponse est identique à celles déterminées à l'aide des solveurs **my_ode** et **ode23** ainsi qu'en calcul symbolique.

5. Résolution de l'équation différentielle avec Simscape

Ouvrir le fichier **simscape_resolution.slx**.

Afin de résoudre l'équation différentielle avec Simscape, il faut construire le modèle multi-physique de la Figure 777.

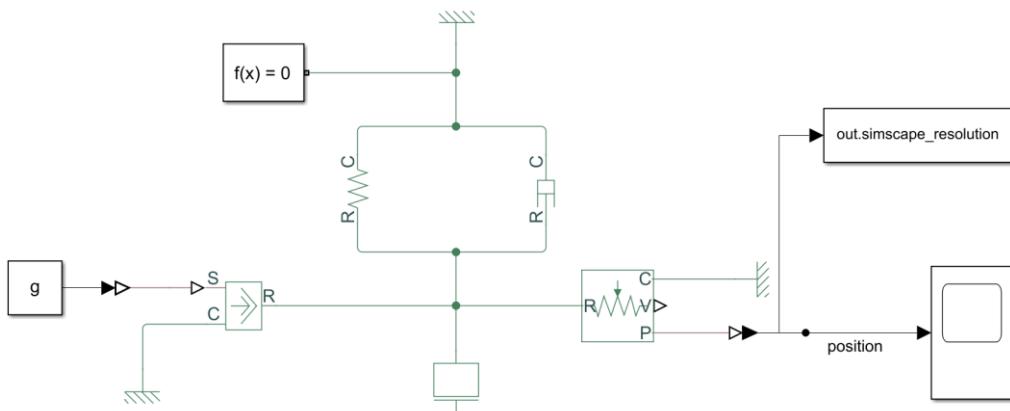


Figure 777 : modèle multi_physique du système masse-ressort-amortisseur

Afin de spécifier la condition initiale, il faut indiquer que le ressort n'est pas à sa position d'équilibre en début de simulation. Pour cela, **double-cliquer** sur le ressort et sélectionner l'onglet **Variables**.

Indiquer ensuite que la déformation initiale est de 0.1 m comme indiqué sur la Figure 778.

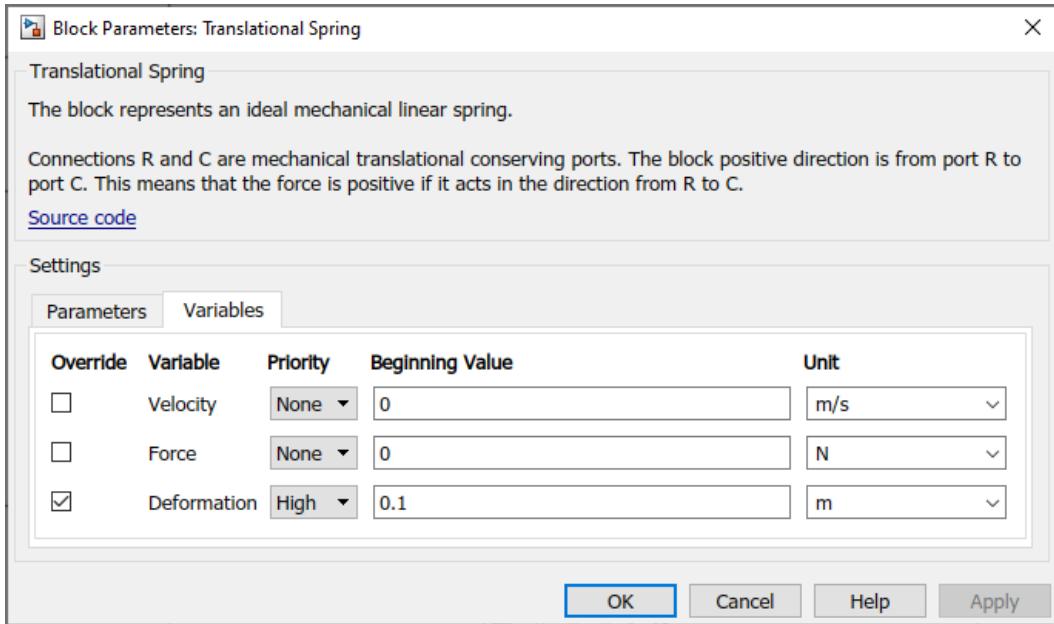


Figure 778 : spécification de la condition initiale dans Simscape

Lancer la simulation et observer la réponse dans le scope de la Figure 779.

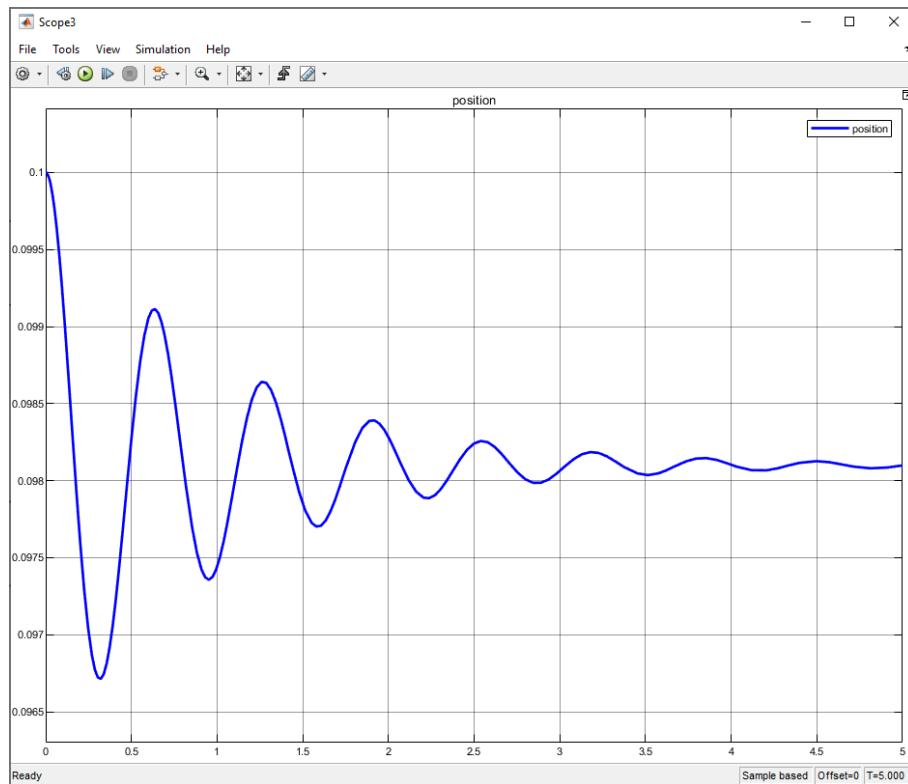


Figure 779 : visualisation de la solution $x(t)$ calculée avec Simscape

Nous pouvons constater que la réponse est identique à celles déterminées par les autres méthodes.

VIII. Résolution de systèmes linéaires – Méthode de Gauss

A. Présentation de la méthode

Ce chapitre présente l'application de la méthode de Gauss dans la résolution d'un système linéaire de n équations à n inconnues admettant une solution unique. La méthode permet de substituer au système initial, un système équivalent qui sera plus facile à résoudre.

Soit le système linéaire de n équations à n inconnues suivant :

$$\begin{cases} a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1 \\ a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n = b_2 \\ \vdots \\ a_{n1} x_1 + a_{n2} x_2 + \dots + a_{nn} x_n = b_n \end{cases}$$

En posant : $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ et $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$

Ce système peut se mettre sous forme matricielle : $AX = B$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Connaissant les matrices A et B , l'algorithme de résolution à mettre en place doit nous permettre de trouver la solution X.

Dans un premier le système sera transformé en système triangulaire équivalent :

$$\begin{cases} a'_{11} x_1 + a'_{12} x_2 + \dots + a'_{1n} x_n = b'_1 \\ 0 + a'_{22} x_2 + \dots + a'_{2n} x_n = b'_2 \\ \vdots \\ 0 + 0 + \dots + a'_{nn} x_n = b'_n \end{cases} \quad \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

Afin de ne travailler que sur une seule matrice pour réaliser les différentes opérations, nous travaillerons avec la matrice augmentée M en concaténant A et B.

$$M = \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \dots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a'_{nn} & b'_n \end{bmatrix}$$

La résolution se fera alors de bas en haut en partant de la dernière équation (une équation pour une inconnue) :

$$x_n = \frac{b'_n}{a'_{nn}}$$

Il suffira ensuite de substituer les inconnues trouvées dans les lignes inférieures pour trouver l'inconnue de la ligne supérieure. Le système est alors résolu en remontant ainsi jusqu'à la première équation.

B. Exemple

Soit le système suivant de 3 équations à 3 inconnues :

$$\begin{cases} x + 4y + z = 4 & (L_1) \\ 2x + 2y + z = 1 & (L_2) \\ x + 4y + 2z = 2 & (L_3) \end{cases} \quad \begin{bmatrix} 1 & 4 & 1 \\ 2 & 2 & 1 \\ 1 & 4 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

1. Recherche du pivot

Nous commençons par chercher le plus grand coefficient devant la première inconnue x . Ce premier coefficient s'appelle le premier **pivot**. Dans notre exemple, le premier pivot pour l'inconnue x se trouve en L_2 .

$$\begin{cases} x + 4y + z = 4 & (L_1) \\ \cancel{2x} + 2y + z = 1 & (L_2) \\ x + 4y + 2z = 2 & (L_3) \end{cases} \quad \begin{bmatrix} 1 & 4 & 1 \\ \cancel{2} & 2 & 1 \\ 1 & 4 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

Le choix du pivot aura une incidence sur la stabilité de l'algorithme. Les choix de pivots trop petits peuvent engendrer des erreurs d'arrondis. Nous admettrons que le choix du plus grand pivot améliore la stabilité de l'algorithme.

2. Echange de lignes

Nous faisons remonter l'équation correspondant au pivot en première position.

$$\begin{cases} \cancel{2x} + 2y + z = 1 & (L_1) \\ x + 4y + z = 4 & (L_2) \\ x + 4y + 2z = 2 & (L_3) \end{cases} \quad \begin{bmatrix} \cancel{2} & 2 & 1 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$$

3. Transvection - Triangularisation

Il faut maintenant faire disparaître l'inconnue correspondant au pivot choisi dans les deux autres équations. Pour cela, nous allons réaliser des opérations de transvection :

$$\boxed{\begin{cases} L_2 \Rightarrow L_2 - \frac{1}{2}L_1 \\ L_3 \Rightarrow L_3 - \frac{1}{2}L_1 \end{cases}}$$

$$\left\{ \begin{array}{l} 2x + 2y + z = 1 \\ x - \frac{2x}{2} + 4y - \frac{2y}{2} + z - \frac{z}{2} = 4 - \frac{1}{2} \\ x - \frac{2x}{2} + 4y - \frac{2y}{2} + 2z - \frac{z}{2} = 2 - \frac{1}{2} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ 3y + \frac{3}{2}z = \frac{3}{2} \end{array} \right. \quad \left[\begin{array}{ccc|c} 2 & 2 & 1 \\ 0 & 3 & 1/2 \\ 0 & 3 & 3/2 \end{array} \right] \left[\begin{array}{c} x \\ y \\ z \end{array} \right] = \left[\begin{array}{c} 1 \\ 7/2 \\ 3/2 \end{array} \right]$$

Nous recommandons la recherche du plus grand pivot avec la seconde inconnue y . Le plus grand pivot pour l'inconnue y est en L_2 . Il n'y pas d'échange de lignes à faire.

$$\left\{ \begin{array}{l} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ 3y + \frac{3}{2}z = \frac{3}{2} \end{array} \right.$$

Il faut refaire une opération de transvection pour faire disparaître la variable y de l'équation 3.

$$\boxed{L_3 \Rightarrow L_3 - L_2}$$

$$\left\{ \begin{array}{l} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ 3y - 3y + \frac{3}{2}z - \frac{z}{2} = \frac{3}{2} - \frac{7}{2} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ z = -2 \end{array} \right. \quad \left[\begin{array}{ccc|c} 2 & 2 & 1 \\ 0 & 3 & 1/2 \\ 0 & 0 & 1 \end{array} \right] \left[\begin{array}{c} x \\ y \\ z \end{array} \right] = \left[\begin{array}{c} 1 \\ 7/2 \\ -2 \end{array} \right]$$

4. Résolution

Le processus de résolution peut alors débuter.

On résout pour commencer l'équation du bas (L_3) :

$$\boxed{z = -2}$$

On injecte la valeur de z pour déterminer y dans la seconde équation (L_2) :

$$3y + \frac{z}{2} = \frac{7}{2} \Rightarrow y = \frac{1}{3} \left(\frac{7}{2} - \frac{z}{2} \right) \Rightarrow \boxed{y = \frac{3}{2}}$$

On injecte les valeurs de y et z pour déterminer x dans la première équation (L_1) :

$$2x + 2y + z = 1 \Rightarrow x = \frac{1}{2} (1 - 2y - z) \Rightarrow \boxed{x = 0}$$

La solution recherchée est
$$X = \begin{bmatrix} 0 \\ 1.5 \\ -2 \end{bmatrix}$$

C. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Pivot de Gauss.**

Ce type d'algorithme de résolution nécessite de nombreuses opérations et peut s'avérer complexe à valider s'il n'est pas structuré rigoureusement avec des fonctions. Une bonne méthode consiste à isoler chaque fonction élémentaire de l'algorithme, de les coder dans des fichiers séparés afin de pouvoir les valider les unes après les autres. Une fois la validation de toutes les fonctions élémentaires effectuée, il sera très simple de construire une dernière fonction permettant de coordonner toutes les fonctions de notre algorithme.

1. Recherche du pivot

Cette recherche consiste à trouver la ligne correspondant au plus grand pivot (en valeur absolue). Il suffira de chercher la ligne avec le plus grand coefficient devant la variable considérée. Pour une colonne k donnée, la fonction devra parcourir tous les coefficients de la matrice qui correspondent à cette colonne en partant de l'élément $A(k,k)$ (Figure 780).

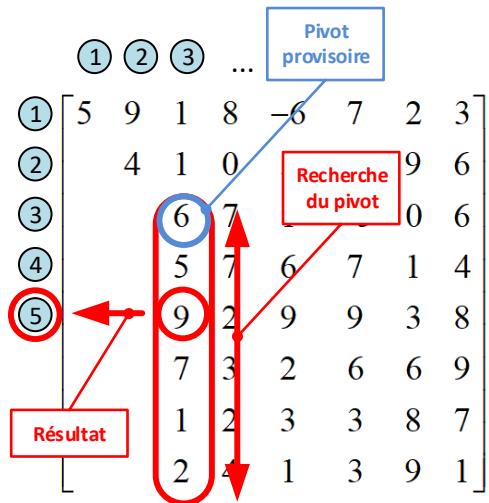


Figure 780 : recherche du numéro de ligne du plus grand pivot

La fonction **num_ligne_pivot.m** (**A,num_col**) permet de réaliser ce processus de recherche (Figure 781).

Cette fonction prendra en arguments :

A: matrice correspondant au système à résoudre

num_col : numéro de la colonne correspondant à la recherche du pivot

La fonction aura comme sorties :

n : numéro de la colonne correspondant au plus grand pivot

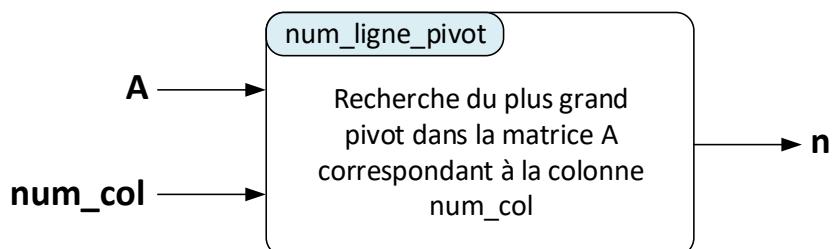


Figure 781 : fonction num_ligne_pivot.m

La Figure 782 montre le codage de la fonction **num_ligne_pivot(A,num_col)**.

```
%% n = num_ligne_pivot(A,num_col)
% Ivan LIEBGOTT © Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
% Recherche la ligne correspondant au plus grand pivot de la colonne
% num_col pour la matrice A

function n = num_ligne_pivot(A,num_col);

% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb_lignes, nb_colonnes] = size(A);

% affectation du pivot provisoire à la ligne correspondant à num_col. La
% ligne du pivot provisoire correspond à la colonne de recherche
n = num_col;
```

```
% on parcourt la colonne en effectuant une comparaison élément par élément
for k = num_col:1:nb_lignes
    if abs(A(k,num_col)) > abs(A(n,num_col))
        % on récupère la valeur du pivot si l'élément comparé est plus
        % grand que le pivot mémorisé
        n = k;
    end
end
end
```

Figure 782 : codage de la fonction num_ligne_pivot(A,num_col)

Afin de valider la fonction, nous allons faire des tests sur une matrice générée aléatoirement.
Tapez dans la fenêtre de commande la commande suivante :

```
>> A = randi(50,5)

A =

 39  15  26  34  13
 22  23  48  15  12
  5  27  32  34  34
 14  23  48  35  43
  8  44  13  4  18
```

La commande randi(50,5) permet de créer une matrice 5x5 en choisissant des nombres entiers aléatoires entre 1 et 50.

Testons maintenant notre fonction en recherchant le plus grand pivot de chaque colonne. Il faut que la fonction **num_ligne_pivot.m** soit dans le path de MATLAB pour que nous puissions l'utiliser dans la fenêtre de commande ou dans un script.

Il faut bien noter que **pour chaque colonne k, la recherche commence à la ligne k** et notre algorithme ne recherche pas le plus grand élément de la colonne.

```
>> num_ligne_pivot(A,1)
ans =
 1 (le plus grand pivot de la colonne 1 est bien en ligne 1, la recherche commence en ligne 1)

>> num_ligne_pivot(A,2)
ans =
 5 (le plus grand pivot de la colonne 2 est bien en ligne 5, la recherche commence en ligne 2)

>> num_ligne_pivot(A,3)
ans =
 4 (le plus grand pivot de la colonne 3 est bien en ligne 3, la recherche commence en ligne 3)

>> num_ligne_pivot(A,4)
ans =
 4 (le plus grand pivot de la colonne 4 est bien en ligne 4, la recherche commence en ligne 4)

>> num_ligne_pivot(A,5)
ans =
```

5 (le plus grand pivot de la colonne 5 est bien en ligne 5, la recherche commence en ligne 5)

Il est également possible de tester notre fonction avec une matrice contenant des nombres positifs et négatifs.

```
>> A = randi([-50,50],5)
```

```
A =
```

```
-23 8 15 21 -4  
27 19 18 -27 16  
-31 5 14 -38 27  
-21 -8 45 11 -15  
-41 15 -29 -5 16
```

```
>> num_ligne_pivot(A,1)
```

```
ans =
```

```
5
```

```
>> num_ligne_pivot(A,2)
```

```
ans =
```

```
2
```

Notre fonction **num_ligne_pivot(A,num_col)** est maintenant validée.

2. Echange de lignes

La fonction **echange_lignes(A,ligne_i,ligne_j)** (Figure 783) permet d'échanger les lignes i et j de la matrice A.

Cette fonction prendra en arguments :

A: matrice

ligne_i : ligne à échanger

ligne_j : ligne à échanger

La fonction aura comme sorties :

A : matrice A modifiée avec les lignes i et j échangées.

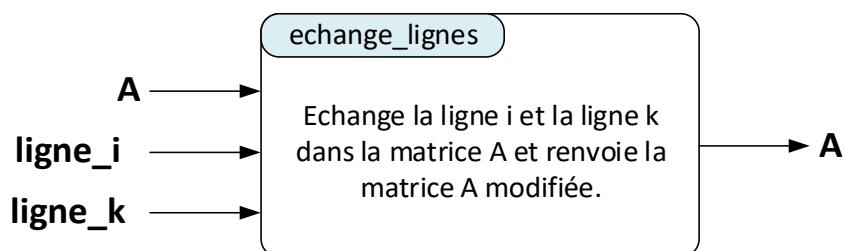


Figure 783 : fonction **echange_lignes(A, ligne_i, ligne_j)**

La Figure 784 montre le codage de la fonction **echange_ligne(A, ligne_i, ligne_k)**.

```
%% B = echange_lignes(A,ligne_i,ligne_k)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
```

```
% Permet d'échanger la ligne_i et la ligne_k dans la matrice A

function A = echange_lignes(A,ligne_i,ligne_k)
% permutation des éléments des lignes i et j
A([ligne_i ligne_k],:) = A([ligne_k ligne_i],:);
end
```

Figure 784 : codage de la fonction echange_ligne(A, ligne_i, ligne_k)

Testons maintenant notre fonction en créant une matrice aléatoirement et en testant la permutation des lignes i et j.

```
>> A = randi(100,5,5)

A =

 76  74  2  83  40
 25  40  34  43  81
 45  69  43  89  76
 69  71  28  40  38
 36  45  20  77  22

>> echange_lignes(A,1,5)

ans =

 36  45  20  77  22
 25  40  34  43  81
 45  69  43  89  76      (les lignes 1 et 5 ont bien permutés par rapport à la matrice A)
 69  71  28  40  38
 76  74  2  83  40

>> echange_lignes(A,2,3)

ans =

 76  74  2  83  40
 45  69  43  89  76
 25  40  34  43  81      (les lignes 2 et 3 ont bien permutés par rapport à la matrice A)
 69  71  28  40  38
 36  45  20  77  22
```

Notre fonction **echange_lignes(A, ligne_i, ligne_j)** est maintenant validée.

3. Transvection

La fonction **transvection(A, Ligne_pivot)** (Figure 785) permet de réaliser les opérations de transvection qui éliminent l'inconnue correspondant au pivot dans toutes les autres équations.

Cette fonction prendra en arguments :

A: matrice augmentée correspondant au système à résoudre

Ligne_pivot : numéro de la ligne du pivot ou se trouve l'inconnue à éliminer dans les autres équations

La fonction aura comme sorties :

A : matrice A modifiée par les opérations de transvection

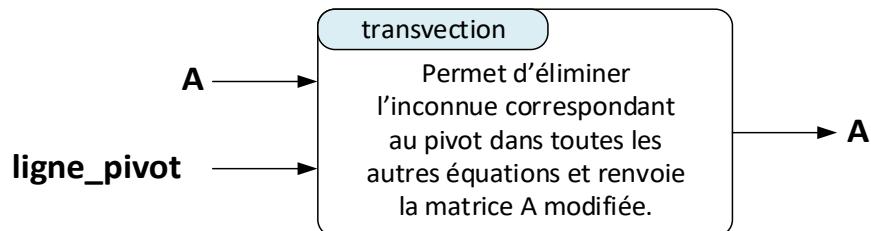


Figure 785 : fonction **transvection(A, num_ligne_pivot)**

La Figure 786 montre le codage de la fonction **transvection(A,num_ligne_pivot)**.

```
%% A = transvection(A,num_ligne_pivot)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
% Permet de soustraire à la ligne Li la Ligne_pivot multipliée par un
% coefficient c.

function A = transvection(A,num_ligne_pivot)
% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb_lignes, nb_colonnes] = size(A);

% la boucle parcourt les lignes en partant de num_ligne_pivot + 1
for k = num_ligne_pivot + 1:1:nb_lignes

    % calcul du coefficient c permettant d'éliminer l'inconnue
    c = A(k,num_ligne_pivot) / A(num_ligne_pivot,num_ligne_pivot);

    % opération de transvection
    A(k,:) = A(k,:) - c * A(num_ligne_pivot,:);
end
end
```

Figure 786 : codage de la fonction **transvection(A, num_ligne_pivot)**

Testons maintenant notre fonction en créant une matrice aléatoirement et en appliquant plusieurs fois la fonction transvection à la matrice M afin de la triangulariser.

```
>> A= transvection(A,1)

A =
 5.0000 14.0000 16.0000 1.0000
 0    -24.2000 -32.8000 0.2000
 0     12.4000 12.6000 17.6000
 0    -16.4000 -29.6000  7.4000

>> A = transvection(A,2)

A =
 5.0000 14.0000 16.0000 1.0000
 0    -24.2000 -32.8000 0.2000
 0      0.0000  -4.2066 17.7025
 0      0        -7.3719  7.2645

>> A = transvection(A,3)
```

```

A =
5.0000 14.0000 16.0000 1.0000
0 -24.2000 -32.8000 0.2000
0 -0.0000 -4.2066 17.7025
0 0.0000 0 -23.7583

```

Notre fonction **transvection(A,num_ligne_pivot)** est maintenant validée.

4. Triangularisation

La fonction **triangularisation(A)** (Figure 787) va triangulariser la matrice augmentée.

Cette fonction prendra en arguments :

A: matrice augmentée correspondant au système à résoudre

La fonction aura comme sorties :

A : matrice A triangularisée

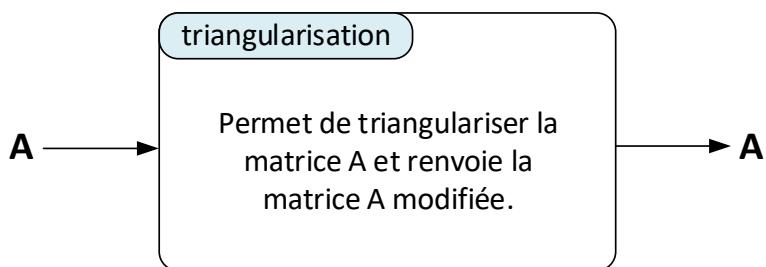


Figure 787 : fonction **triangularisation(A)**

La Figure 788 montre le codage de la fonction **triangularisation(A).m**.

```

%% A = triangularisation(A)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
% Cette fonction permet de triangulariser la matrice augmentée du système

function A = triangularisation(A)
% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb_lignes, nb_colonnes] = size(A);

% la boucle parcourt les lignes, fait remonter la ligne de plus grand
% pivot en utilisant la fonction echange_lignes et réalise les opérations de
% transvection en utilisant la fonction transvection afin de triangulariser la
% matrice

for k = 1:1:(nb_lignes-1)
ligne_pivot = num_ligne_pivot(A,k);
    % remonté du plus grand pivot
    if ligne_pivot ~= k
        A = echange_lignes(A,ligne_pivot,k);
    end
    % opérations de transvections pour toutes les lignes sous la ligne du
    % pivot
    for p = k+1:nb_lignes-1
        A = transvection(A,p);
    end
end

```

```
end  
end
```

Figure 788 : codage de la fonction **triangularisation(A)**

Afin de valider la fonction **triangularisation** nous allons construire une matrice augmenter de 5 lignes et 6 colonnes.

```
>> A = randi(5,5,6)  
  
A =  
  
1 1 1 2 3 4  
2 3 2 2 4 5  
5 1 3 1 4 3  
1 5 1 2 1 1  
2 1 5 1 1 5  
  
>> triangularisation(A)  
  
ans =  
  
5.0000 1.0000 3.0000 1.0000 4.0000 3.0000  
0 2.6000 0.8000 1.6000 2.4000 3.8000  
0 0 0.1538 1.3077 1.4615 2.2308  
0 0 0 8.0000 6.0000 9.0000  
0 0 0 0 -12.620 -15.1875
```

Notre fonction **triangularisation(A)** est maintenant validée. La matrice est prête pour la phase de résolution.

5. Résolution

La fonction **resolution(M)** (Figure 789) va résoudre le système en déterminant la valeur des inconnues.

Cette fonction prendra en arguments :

M: matrice augmentée correspondant au système à résoudre

La fonction aura comme sorties :

X : vecteur contenant les solutions du système d'équation

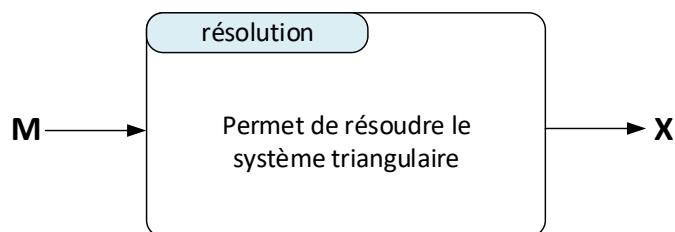


Figure 789 : fonction **réolution.m**

La matrice augmentée M comporte n lignes et $p=n+1$ colonnes et le vecteur X , solution du système

$$AX = B \text{ est de la forme } X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

A ce stade, la matrice M a été triangulaire et sa forme avant la phase de résolution est la suivante :

$$M = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_n \end{bmatrix} = \begin{bmatrix} M(1,1) & M(1,2) & \cdots & M(1,n) & M(1,n+1) \\ 0 & M(2,2) & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & M(n-1,n+1) \\ 0 & 0 & 0 & M(n,n) & M(n,n+1) \end{bmatrix}$$

Nous commençons par résoudre l'équation du bas correspondant à la ligne n :

$$x_n = \frac{M(k,n+1)}{M(n,n)}$$

Ensuite, nous passons à la ligne $(n-1)$, en utilisant la valeur de x_n trouvée précédemment :

$$x_{n-1} = \frac{1}{M(n-1,n-1)} [M(n-1,n+1) - M(k,n).X(n)]$$

Cette relation peut être généralisée au rang k et les autres solutions x_k sont données par la relation suivante qui sera implémentée dans l'algorithme :

$$x_k = \frac{1}{M(k,k)} \left[M(k,n+1) - \sum_{i=k+1}^n M(k,i).X(i) \right]$$

La Figure 790 montre le codage de la fonction **resolution(M)**.

```
%% M = résolution(M)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
% Résolution d'un système triangulaire

function X = resolution(M)
[nb_lignes, nb_colonnes] = size(M);
X = zeros(nb_lignes,1);
% on parcourt les lignes en partant du bas et en remontant
for k = nb_lignes:-1:1
    somme = 0;
    % calcul de la somme en injectant les valeurs des inconnues déjà
    % déterminées
    for i = k+1:1:nb_lignes
        somme = somme + M(k,i) * X(i);
```

```

end
% calcul de la solution X(k)
X(k) = (1/M(k,k))*(M(k,nb_lignes + 1) - somme);
end
end

```

Figure 790 : codage de la fonction resolution(M)

Afin de tester notre fonction **resolution(M)**, nous allons saisir la matrice triangulaire augmentée qui correspond au système donné en exemple en début de paragraphe. Ensuite nous allons la triangulariser puis la résoudre à l'aide des fonctions que nous avons définies.

```

>> M = [ 1 4 1 4 ; 2 2 1 1 ; 1 4 2 2]

M =
 1   4   1   4
 2   2   1   1
 1   4   2   2

>> X = resolution(triangularisation(M))

X =

```

0
1.5000
-2.0000

Nous pouvons vérifier que l'algorithme nous donne bien la solution de notre système.

6. Pivot de Gauss

La fonction **Pivot_de_Gauss(A,B)** (Figure 791) prendra en argument les matrices A et B telles que le système à résoudre soit de la forme $AX=B$.

Cette fonction prendra en arguments :

A: matrice A telle que le système à résoudre soit de la forme $AX=B$
B: matrice B telle que le système à résoudre soit de la forme $AX=B$

La fonction aura comme sorties :

X : vecteur contenant les solutions du système d'équation $AX=B$

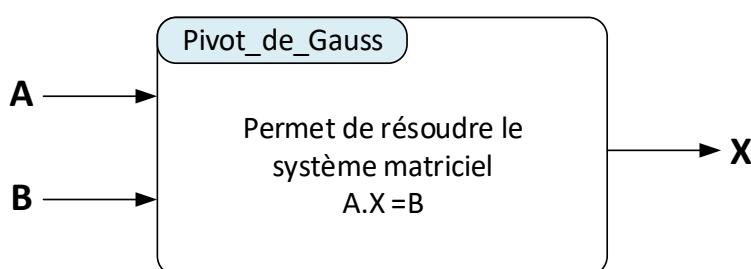


Figure 791 : fonction Pivot_de_Gauss(A,B)

La Figure 792 montre la hiérarchisation des fonctions qui sont utilisées dans l'algorithme du pivot de Gauss.

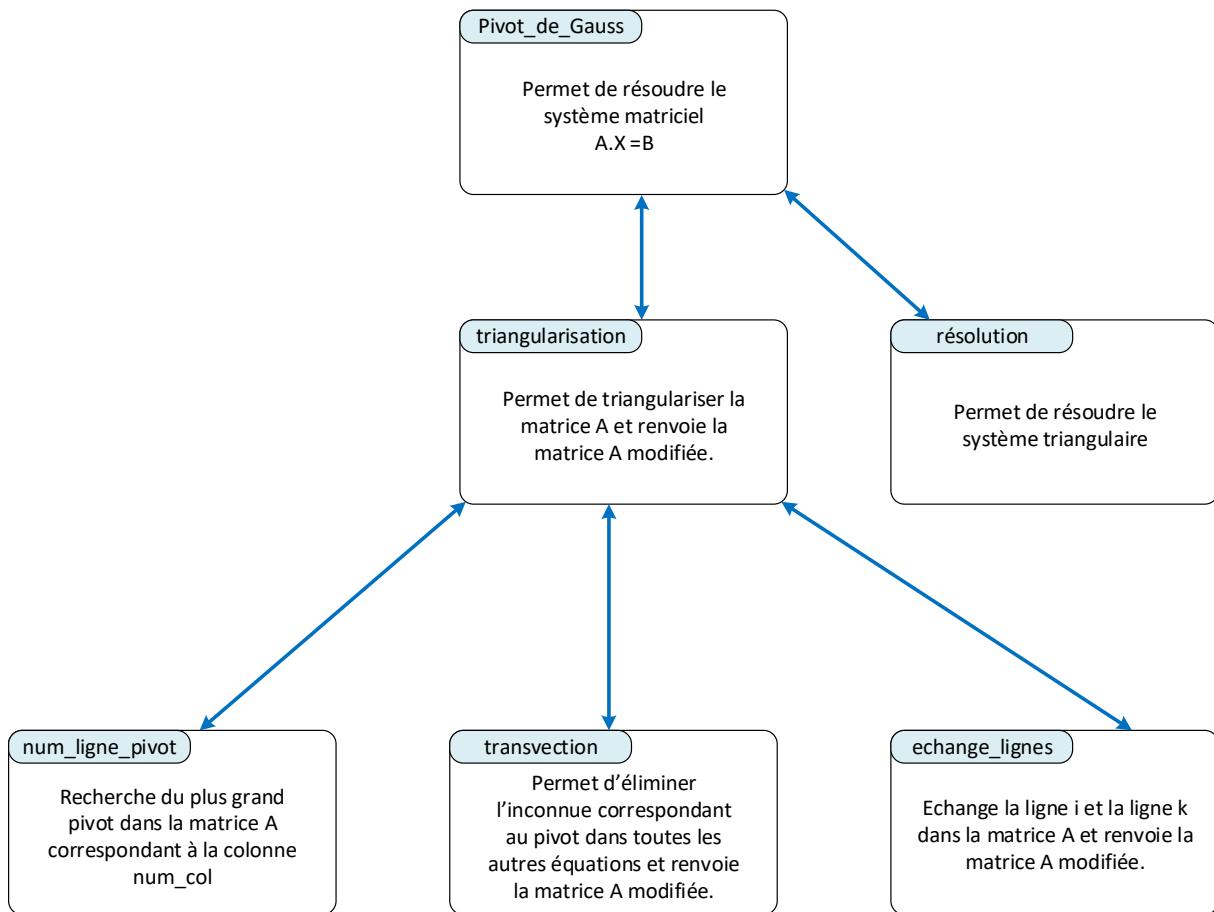


Figure 792 : hiérarchisation des fonctions utilisées dans l'algorithme du pivot de Gauss

La Figure 793 montre le codage de la fonction **Pivot_de_Gauss(A,B)**.

```

%% X = Pivot_de_Gauss(A,B)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Algorithme de résolution d'un système d'équation linéaire en utilisant la
% méthode du pivot de Gauss
% le système est de la forme [A] [X] = [B]
% On forme la matrice augmentée

function X = Pivot_de_Gauss(A,B)
% concaténation de A et de B pour former la matrice augmentée M
M = [A B];

% triangularisation de M
M = triangularisation (M);

% résolution de M
X = resolution(M)
end

```

Figure 793 : codage de la fonction **Pivot_de_Gauss(A,B)**

Afin de tester notre fonction **Pivot_de_Gauss.m(A,B)**, nous allons créer les deux matrice A et B correspondant à notre système et utiliser notre fonction.

```
>> A = [1 4 1; 2 2 1; 1 4 2]
```

```

A =
1 4 1
2 2 1
1 4 2

>> B = [4;1;2]
B =

4
1
2

>> Pivot_de_Gauss(A,B)

X =

0
1.5000
-2.0000

```

Nous pouvons vérifier que la fonction nous donne bien la solution du système d'équations.

D. Robustesse

Afin d'évaluer la robustesse de notre algorithme, nous allons comparer les résultats trouvés avec un calcul réalisé par MATLAB. En langage MATLAB en tapant $A \setminus B$ dans la fenêtre de commande, MATLAB donne la solution X du système d'équation.

```

>> A \ B

ans =

0
1.5000
-2.0000

```

Nous allons tester notre fonction sur un système de 10 équations à 10 inconnues

```

>> A = randi(100,10,10);
>> B = randi(100,10,1);
>> A \ B

ans =

0.4849
-0.6699
0.3289
0.6217
-0.3165

```

```
0.1280
0.1572
1.5665
-0.8841
-0.6414

>> Pivot_de_Gauss(A,B)

X =
0.4849
-0.6699
0.3289
0.6217
-0.3165
0.1280
0.1572
1.5665
-0.8841
-0.6414
```

Nous pouvons constater que les solutions données par MATLAB sont identiques aux solutions déterminées à l'aide de notre algorithme.

Chapitre 12 : MATLAB/Simulink online

I. Présentation

MATLAB/Simulink est également utilisable en version online. Il n'est donc pas forcément nécessaire d'installer MATLAB/Simulink sur son ordinateur personnel pour utiliser les fonctionnalités du logiciel. Une simple connexion à MATLAB/Simulink online à l'aide d'un login et d'un mot de passe permettra de développer des modèles ou du code. La capacité de calcul est déportée vers des serveurs et l'ordinateur joue alors le rôle d'un simple afficheur. Cette solution sera particulièrement intéressante pour les utilisateurs qui ne dispose pas d'un ordinateur performant ou pour ceux qui ne souhaite pas utiliser de l'espace sur leur disque dur avec les fichiers d'installation. Cette solution permet également d'utiliser MATLAB/Simulink dans les laboratoires d'enseignement sans procéder à l'installation et en travaillant toujours sur la dernière version du logiciel avec les étudiants. Ces derniers pourront sans difficultés travailler sur leurs fichiers à la maison en se connectant à MATLAB/Simulink online. L'utilisation de MATLAB/Simulink online est étroitement liée à l'exploitation du cloud dédié de 5Go, appelé **MATLAB Drive** et associé à chaque compte Mathworks. Il sera ainsi très simple d'accéder aux fichiers depuis n'importe quel ordinateur (Figure 794). L'outil **MATLAB Drive Connector** permettra de lier le dossier de **MATLAB Drive** au gestionnaire de fichier de Windows afin de faciliter l'exploitation de ces fichiers. Les fichiers sont alors stockés sur le disque dur de l'ordinateur et sont synchronisés avec le cloud.

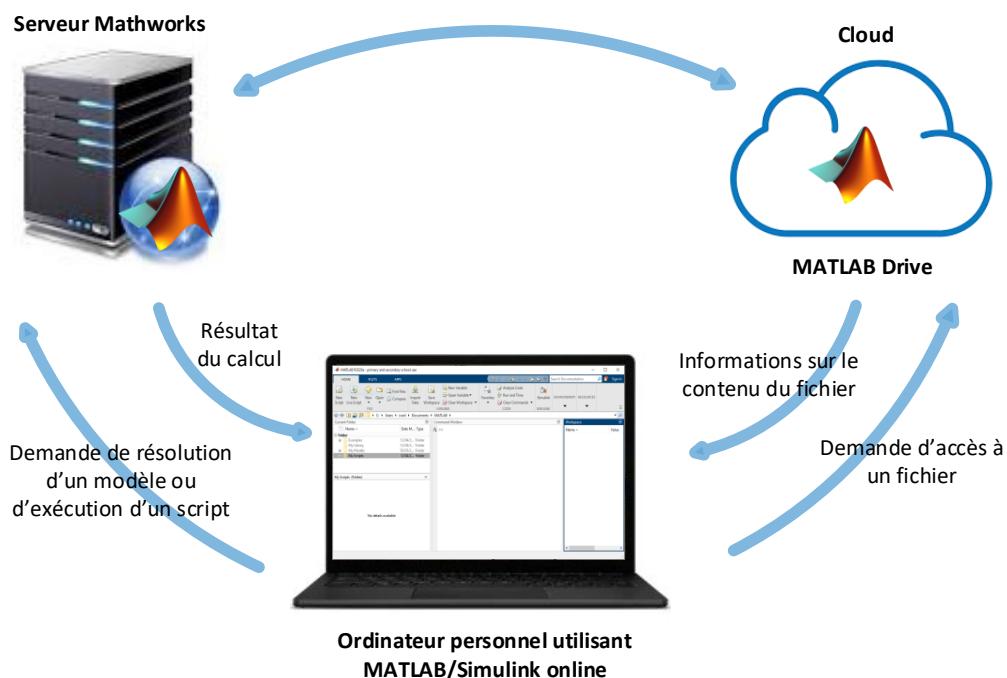


Figure 794 : MATLAB/Simulink online et MATLAB Drive

II. Installation de l'environnement de travail à distance

Afin d'utiliser MATLAB/Simulink online, procéder à l'installation de **MATLAB Drive**, puis de **MATLAB Drive Connector** à partir de liens suivants :

Lien pour installer **MATLAB Drive** :

<https://www.mathworks.com/products/matlab-drive.html>

Lien pour installer **MATLAB Drive Connector** :

<https://www.mathworks.com/products/matlab-drive.html#matlab-drive-connector>

Une fois l'installation réalisée, vous pouvez voir apparaître votre dossier MATLAB Drive dans la version de MATLAB installée sur votre ordinateur (Figure 795)

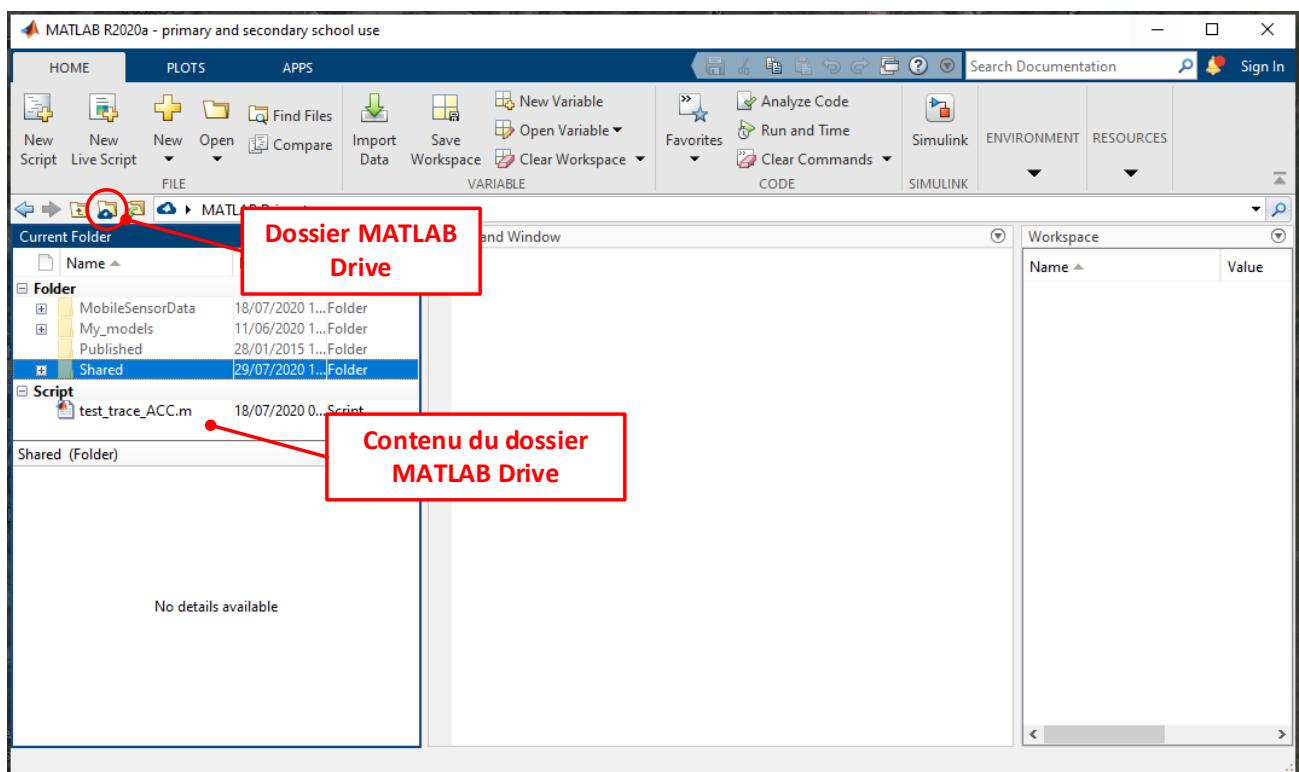


Figure 795 : Dossier MATLAB Drive dans la version installée de MATLAB/Simulink

Vous pouvez également visualiser l'interface de MATLAB Drive Connector dans la barre de tâche de Windows (en bas à droite de votre écran) (Figure 796).

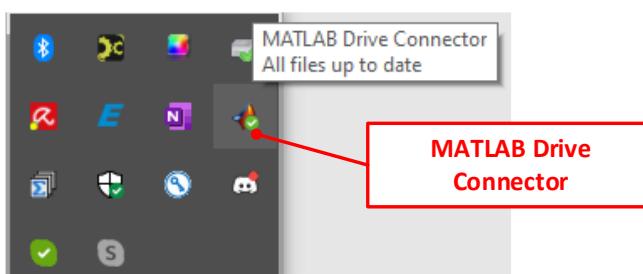


Figure 796 : visualisation de MATLAB Drive Connector dans la barre de tâche de Window

Cliquer sur l'icône de **MATLAB Drive Connector** afin d'ouvrir l'interface (Figure 797).

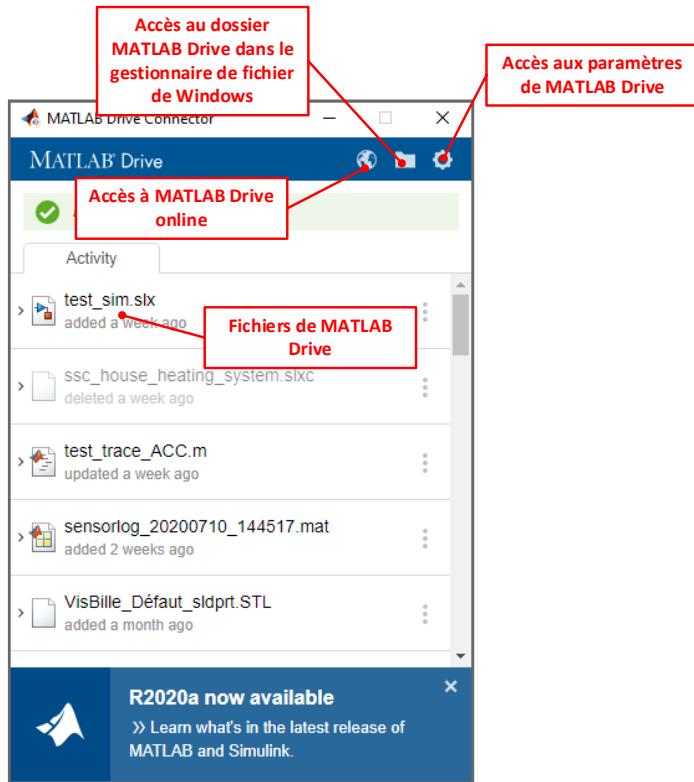


Figure 797 : interface de MATLAB Drive Connector

Il est possible à partir de cette interface :

- D'accéder à MATLAB Drive online
- D'accéder au dossier MATLAB Drive à partir du gestionnaire de fichiers de Windows
- D'accéder aux paramètres de MATLAB Drive pour indiquer la localisation du dossier MATLAB Drive sur le disque dur de l'ordinateur.

Nous verrons plus tard dans ce chapitre comment exploiter cette plateforme de travail.

III. Utilisation de MATLAB/Simulink online

Pour utiliser MATLAB online, il suffit de cliquer sur le lien suivant et de renseigner le login et le mot de passe de son compte Mathworks.

Lien vers **MATLAB online** :

<https://www.mathworks.com/products/matlab-online.html>

Une fois les informations d'authentification saisies, la fenêtre de travail de MATLAB/Simulink online apparaît (Figure 798).

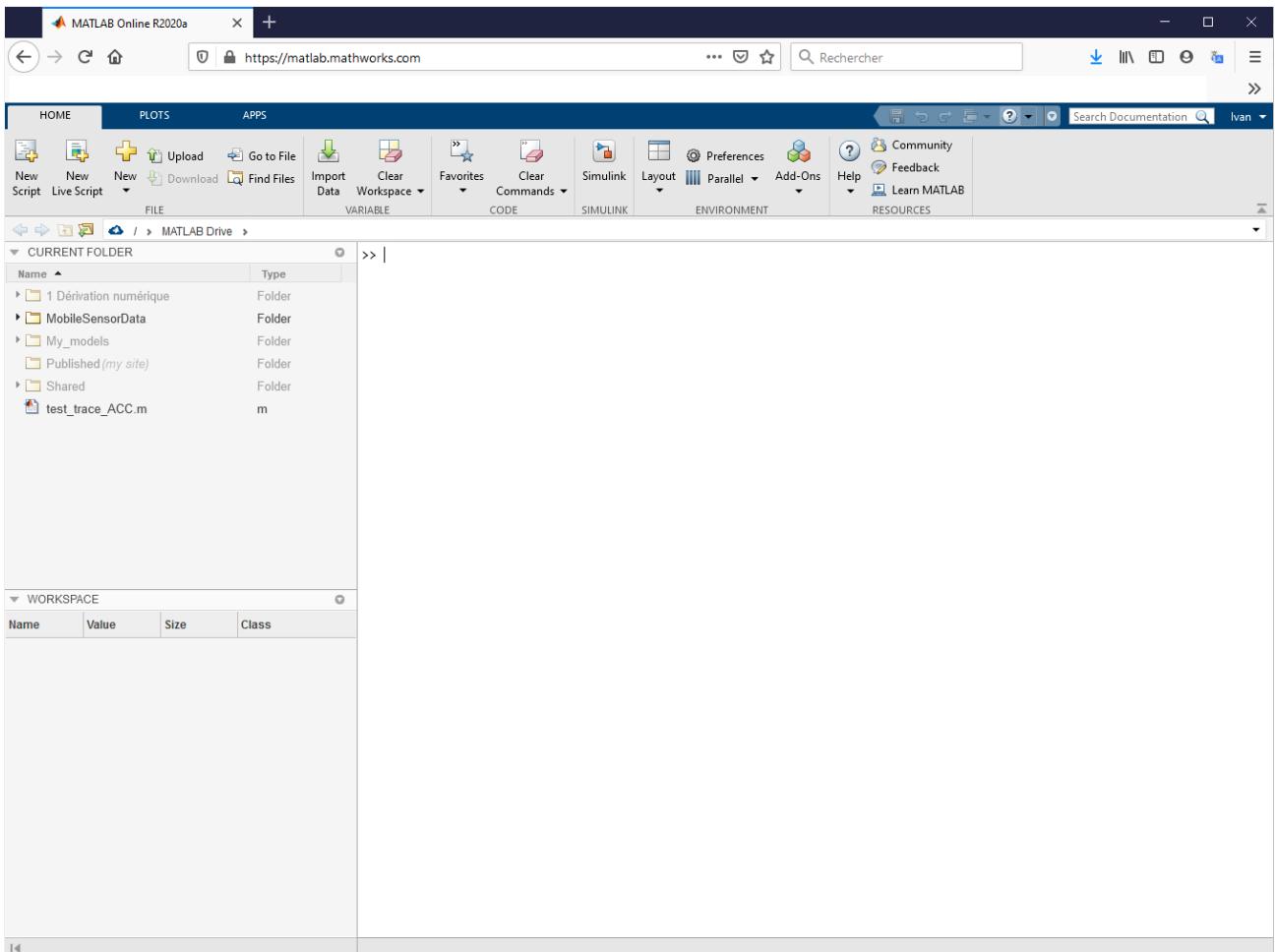


Figure 798 : fenêtre de MATLAB online

Il est maintenant possible de travailler dans l'environnement online, les fonctionnalités et les menus sont strictement identiques à la version MATLAB installée sur le PC.

A. Création d'un script élémentaire avec MATLAB online



Ouvrir un nouveau script en cliquant sur **New Script** et saisir les lignes de code suivantes :

```
untitled2.m
1 %% My first script on line
2 - t = 0 : 0.01:10;
3 - plot(sin(t))
```

Figure 799 : script à saisir online

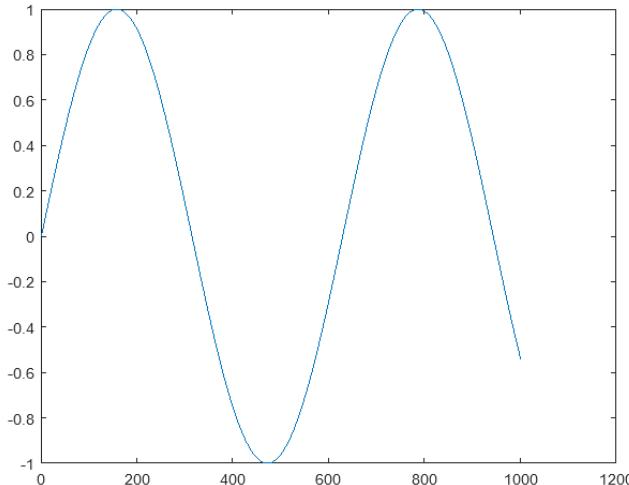


Sauvegarder le script en cliquant sur **Save**.

Le fichier est alors automatiquement sauvegardé dans le dossier courant de MATLAB Drive et sera accessible dans le cloud.



Cliquez sur **Run** pour exécuter le script et visualiser le résultat



B. Création d'un modèle élémentaire avec Simulink online



Cliquez maintenant sur **SIMULINK** pour lancer Simulink online. La Simulink Start Page s'ouvre alors et vous pouvez créer un Blank Model (Figure 800).

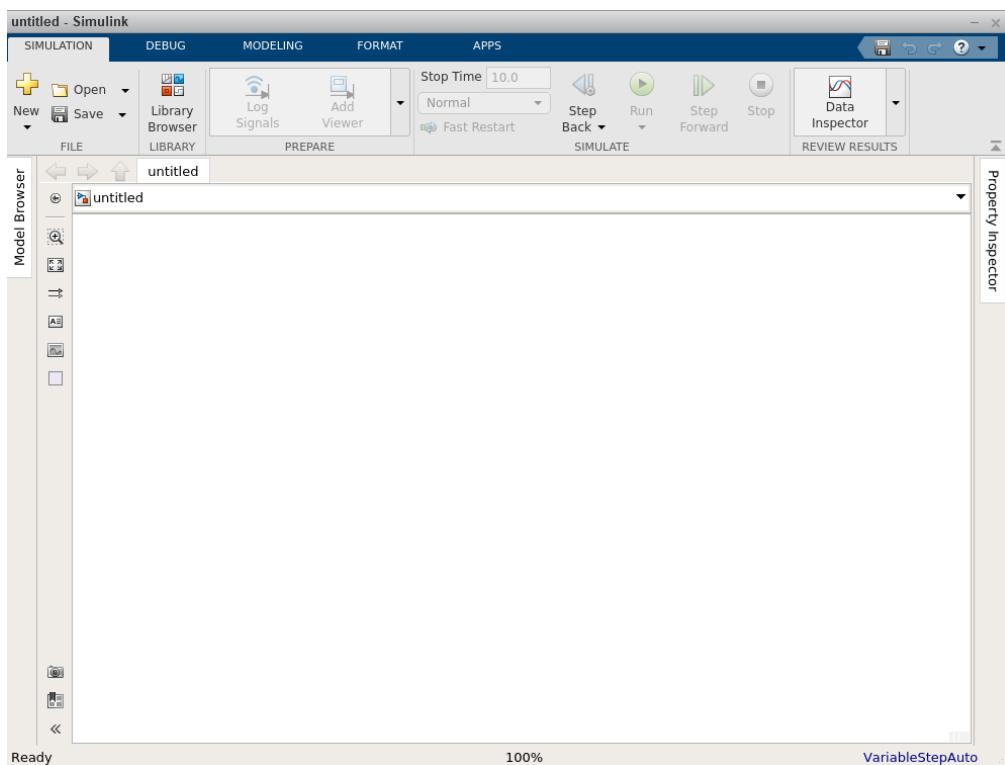


Figure 800 : création d'un Blank Model dans Simulink online

Cliquez alors sur  pour ouvrir la librairie Simulink.

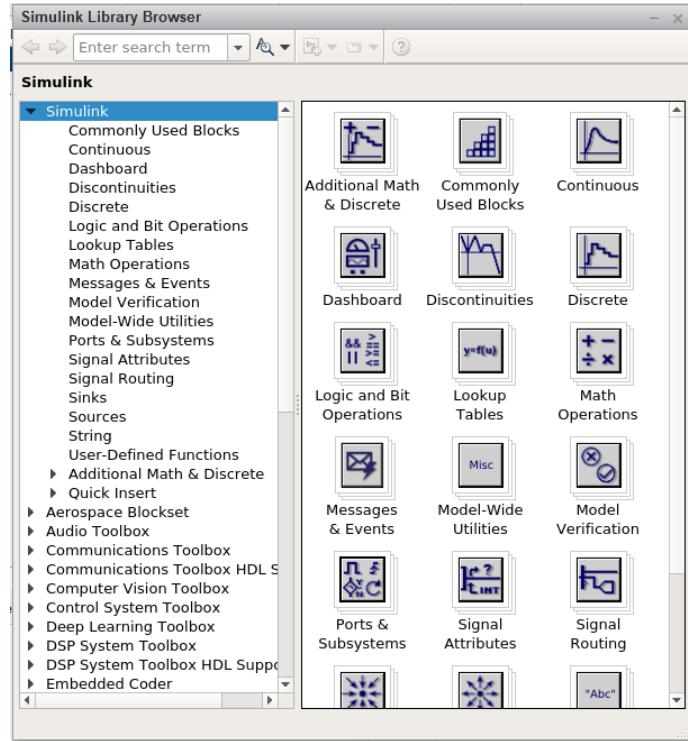


Figure 801 : librairie de Simulink online

Vous pouvez maintenant créer vos modèles en faisant glisser les blocs dans la fenêtre du modèle. Créez un modèle simple et exécutez-le.

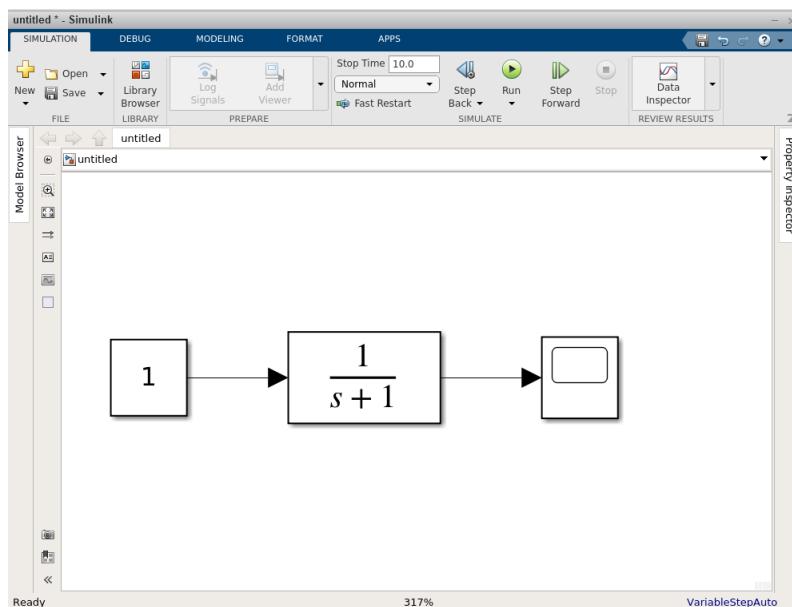


Figure 802 : modèle créé à l'aide de Simulink online

Ouvrir le scope pour visualiser le résultat (Figure 803).

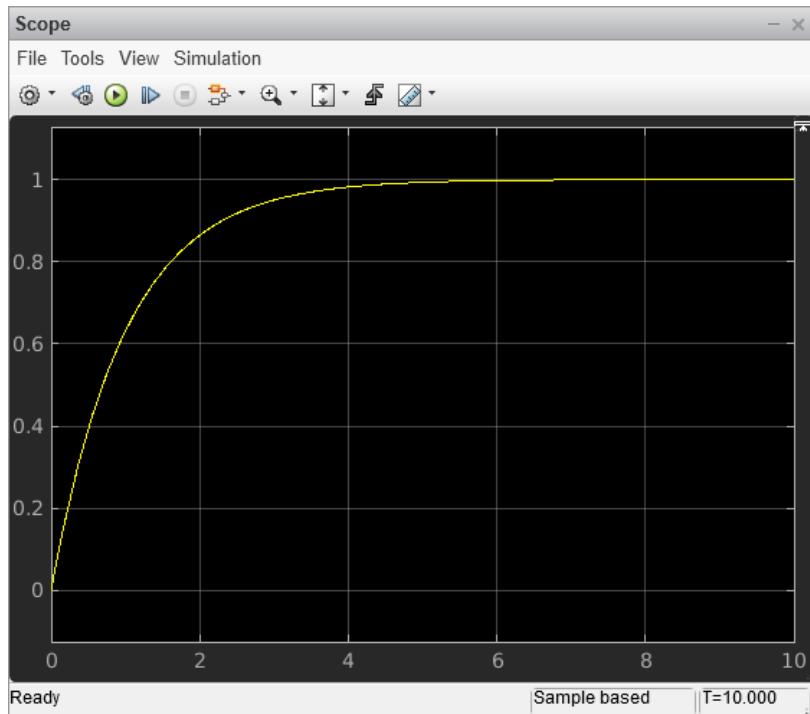


Figure 803 : résultat de la simulation

Sauvegarder le fichier en cliquant sur Save ▾.

Le fichier est alors automatiquement sauvegardé dans le dossier courant de MATLAB Drive et sera accessible dans le cloud.

C. Exploitation d'un modèle multi-physique avec MATLAB/Simulink on line

Nous allons maintenant exécuter un modèle multi-physique complexe en utilisant MATLAB/Simulink online. Pour cela copier tout le contenu du dossier

Chapitre_3_Strategie_de_conception_d_un_modele_multi_physique / Maxpid dans le dossier **MATLAB drive/My_Models**.

Une fois le dossier copié, placer le dossier dans le Path de **MATLAB online** en cliquant avec le bouton droit de la souris sur le dossier **Maxpid** et en sélectionnant **Add to Path/ Selected Folders and Subfolders** (Figure 804).

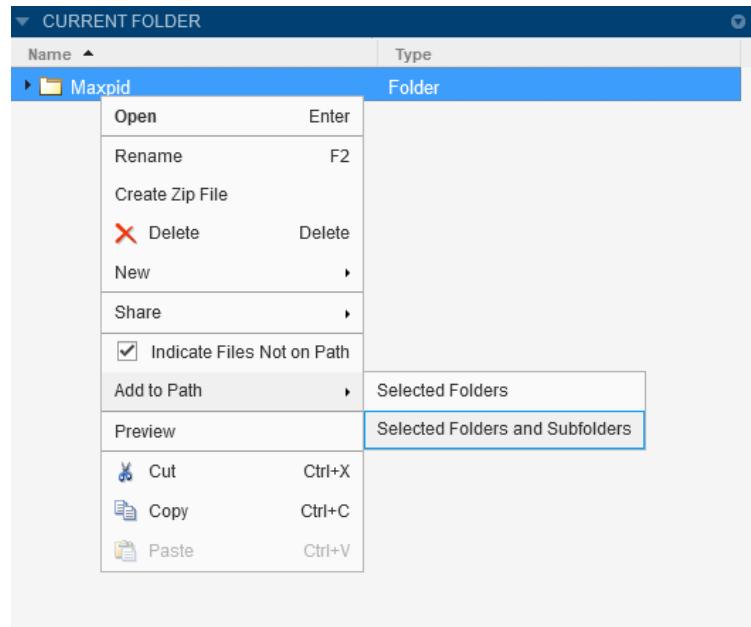


Figure 804 : ajout du dossier Maxpid dans le Path de MATLAB online

Ouvrir le dossier Maxpid et ouvrir le modèle **MAXPID.slx**.

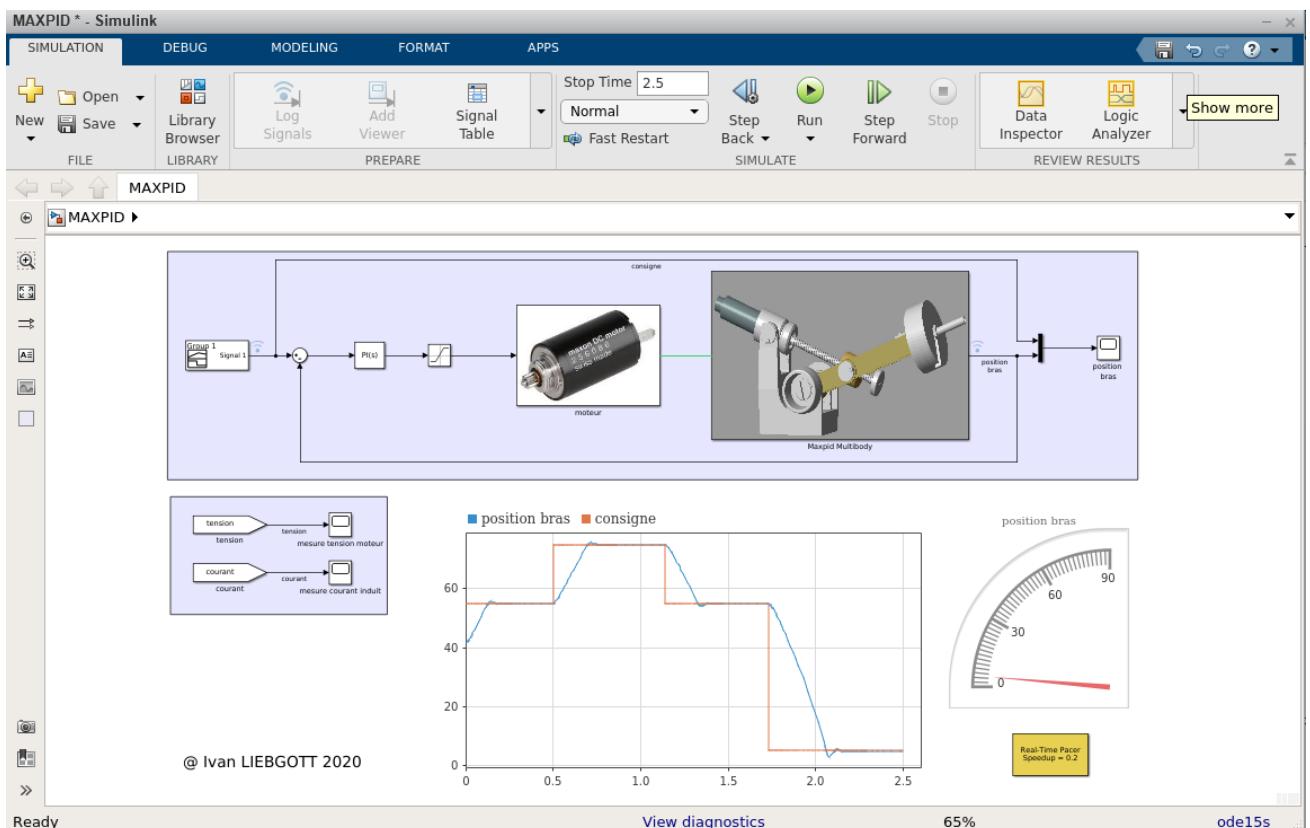


Figure 805 : modèle du robot Maxpid ouvert dans MATLAB/Simulink online

Lancer la simulation du modèle en cliquant sur et observer les résultats obtenus dans le dashboard scope et dans le scope.

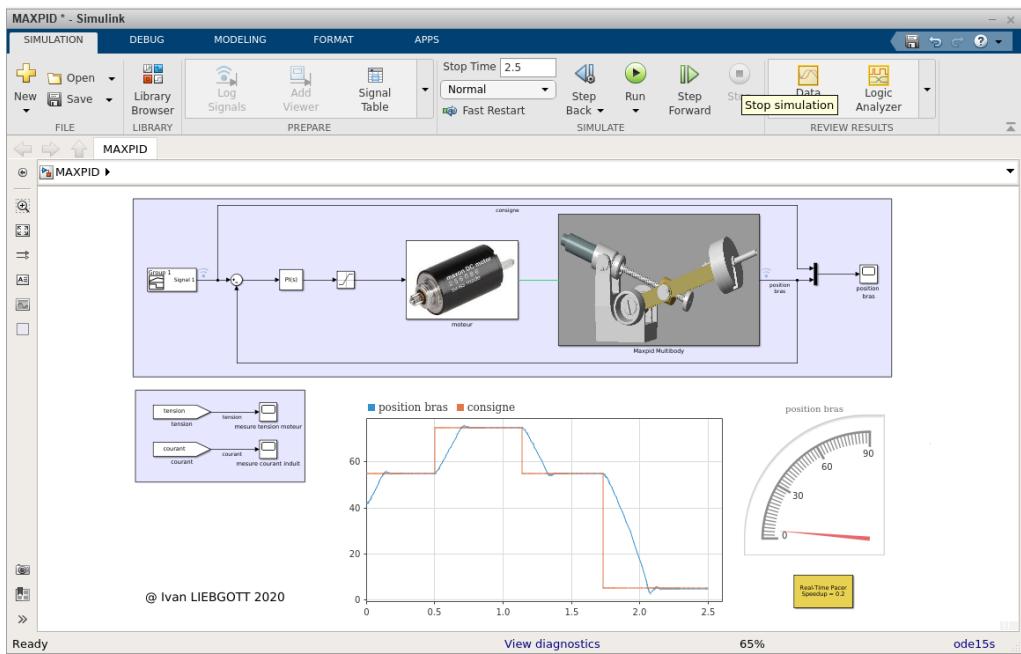


Figure 806 : résultats de la simulation

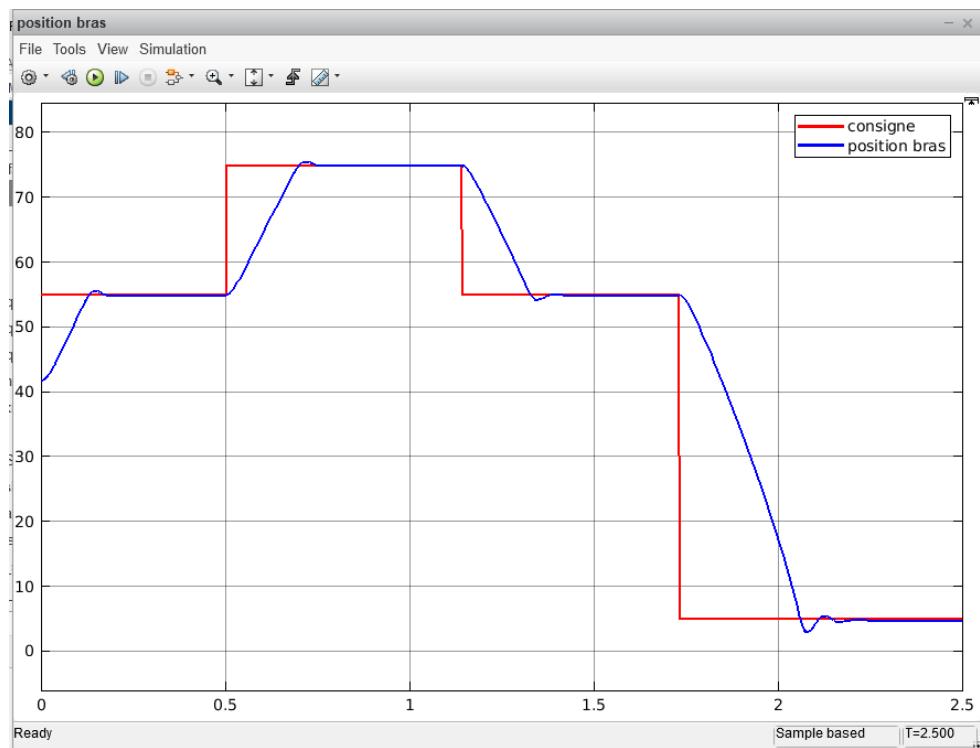


Figure 807 : visualisation des résultats dans le scope

Nous pouvons constater que la simulation se déroule tout à fait normalement à l'exception de la fenêtre Mechanics Explorer qui n'est pas encore disponible sur la version online. Il n'est donc pas possible de visualiser l'animation de la maquette 3D. Cette fonctionnalité sera disponible sur les versions ultérieures.

D. Partager des dossiers contenant ses modèles avec d'autres utilisateurs

Il est très simple à partir du dossier MATLAB Drive de partager ses fichiers ou ses dossiers avec d'autres utilisateurs qui possèdent également un compte Mathworks et un Drive MATLAB. Cette fonctionnalité est très pratique pour mettre à disposition des étudiants le dossier qui contient les fichiers de travail.

Il existe plusieurs méthodes pour partager un dossier de MATLAB Drive.

- En utilisant la version de MATLAB installée sur son ordinateur
- En utilisant MATLAB/Simulink online
- En utilisant MATLAB Drive online

Il est possible de créer un lien qui pointera vers le dossier que l'on souhaite mettre à disposition

1. Partage d'un dossier à l'aide de la version de MATLAB installée sur son ordinateur

Sélectionner **MATLAB Drive** pour qu'il apparaisse dans la fenêtre **Current Folder**.

Cliquer avec le bouton droit de la souris sur le dossier que vous souhaitez partager et sélectionner **Share/Manage Members** (Figure 808).

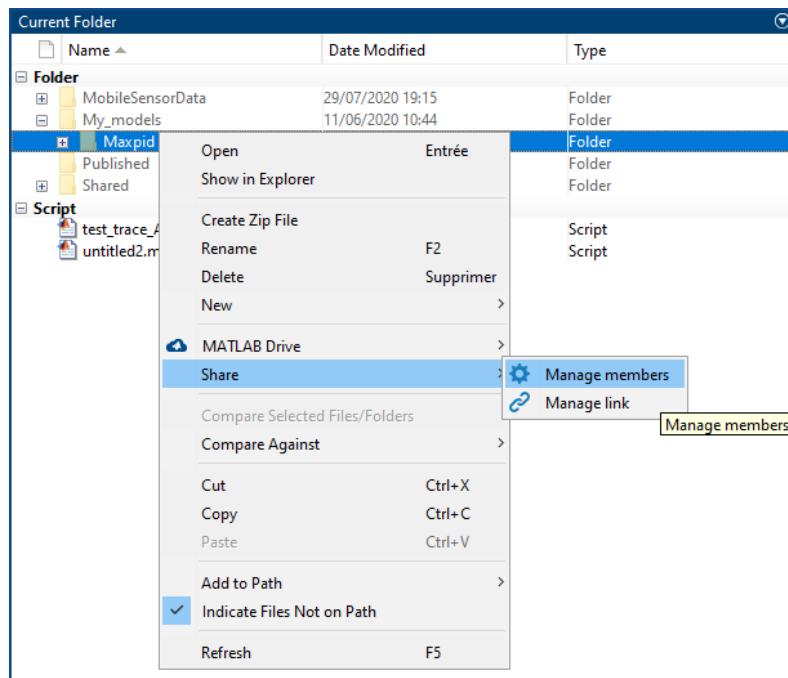


Figure 808 : partage d'un dossier du Drive de MATLAB

La fenêtre MATLAB Drive Sharing s'ouvre et vous permet d'inviter des membres à accéder à votre dossier. Il suffit d'indiquer l'adresse mail de leur compte Mathworks et de préciser si le membre aura un accès en simple consultation (**Can View**) ou s'il aura la possibilité de modifier les fichiers (**Can Edit**) (Figure 809).

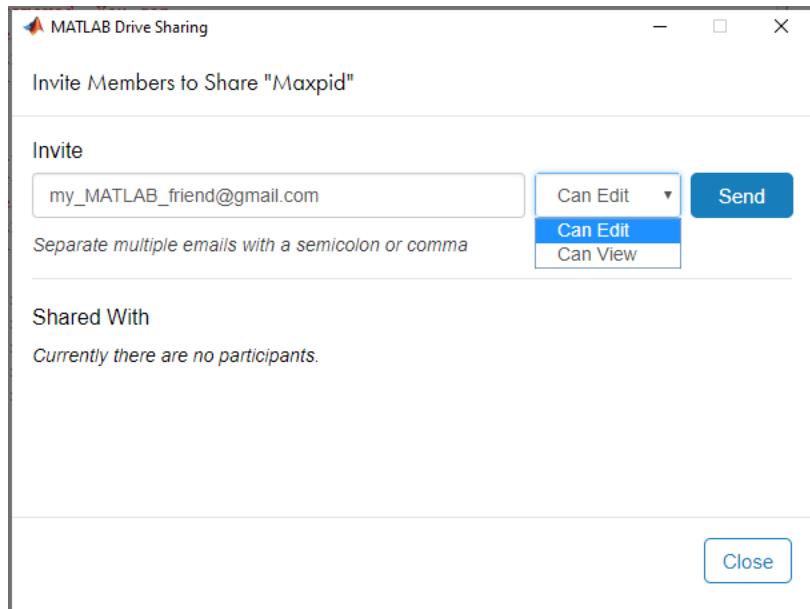


Figure 809 : définition des membres et des droits d'accès au dossier

Un fois l'invitation envoyée, l'utilisateur recevra un mail lui indiquant que vous voulez partager un dossier avec lui et pourra ensuite accéder aux fichiers contenus dans le dossier partagé.

En cliquant ensuite sur **Back**, vous revenez dans la fenêtre initiale et vous pouvez alors modifier à tout moment les droits et les membres qui peuvent accéder à votre dossier (Figure 810).

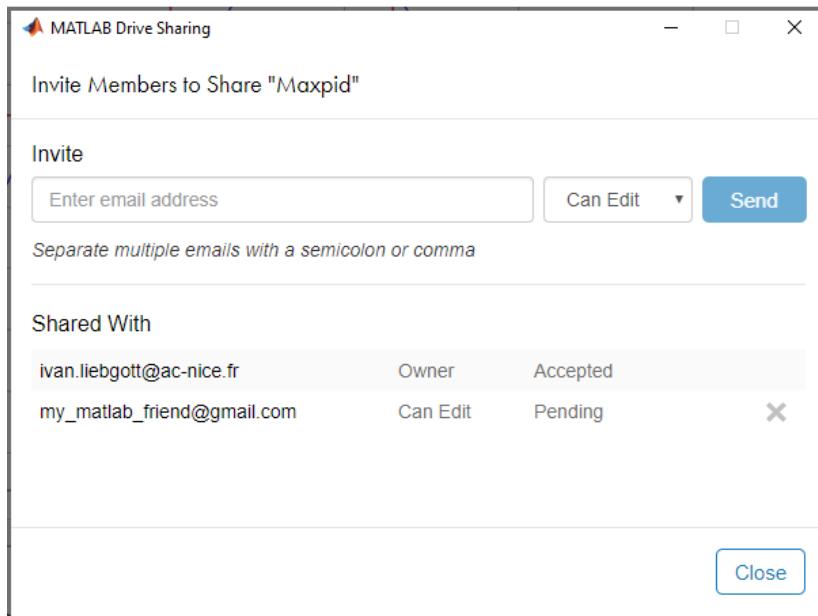


Figure 810 : gestion des droits d'accès et des membres

Il est également possible de créer un simple lien de téléchargement qui pointera sur le dossier partagé que vous pouvez alors diffuser (Figure 811).

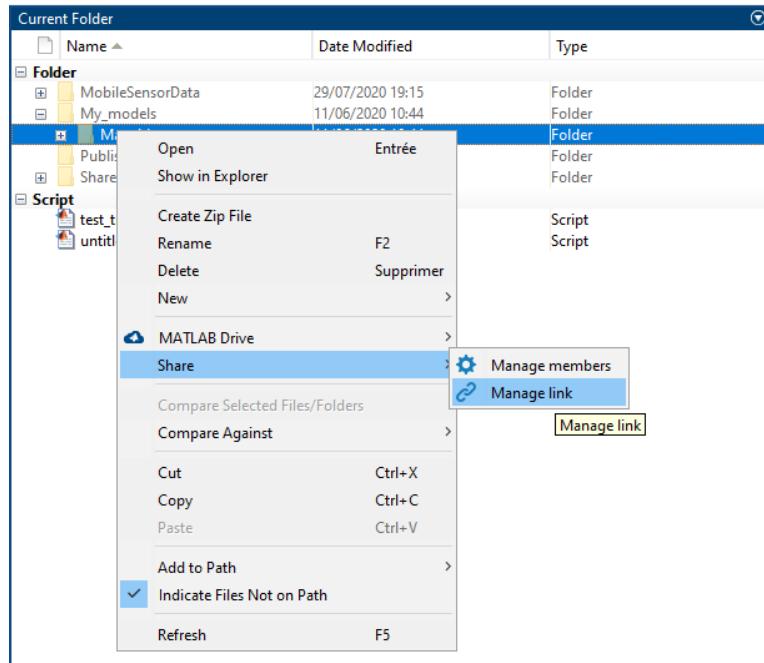


Figure 811 : création d'un lien pour partager un dossier de MATLAB Drive

2. Partage d'un dossier en utilisant MATLAB online

A partir de la fenêtre **CURRENT FOLDER** de **MATLAB Drive**, cliquer avec le bouton droit de la souris sur le dossier que vous voulez partager et sélectionner **Share/Invite Members** ou **Share/Create Link** (Figure 812). Le reste de la procédure est identique au partage en utilisant la version installée de MATLAB.

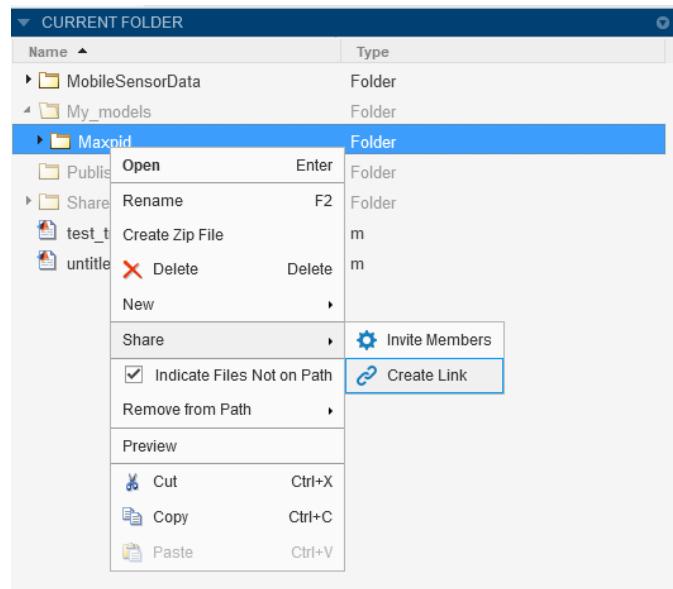


Figure 812 : partage d'un dossier en utilisant MATLAB on line

3. Partage d'un dossier en utilisant MATLAB Drive online

Ouvrir l'interface de **MATLAB Drive Connector** dans la barre de tâche de Windows (en bas à droite de votre écran) (Figure 813).

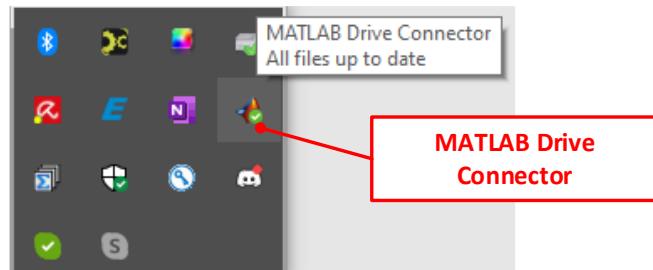


Figure 813 : ouverture de l'interface MATLAB Drive Connector

A partir de la fenêtre de l'interface de **MATLAB Drive Connector**, cliquer sur l'icône pour lancer **MATLAB Drive online** (Figure 814)

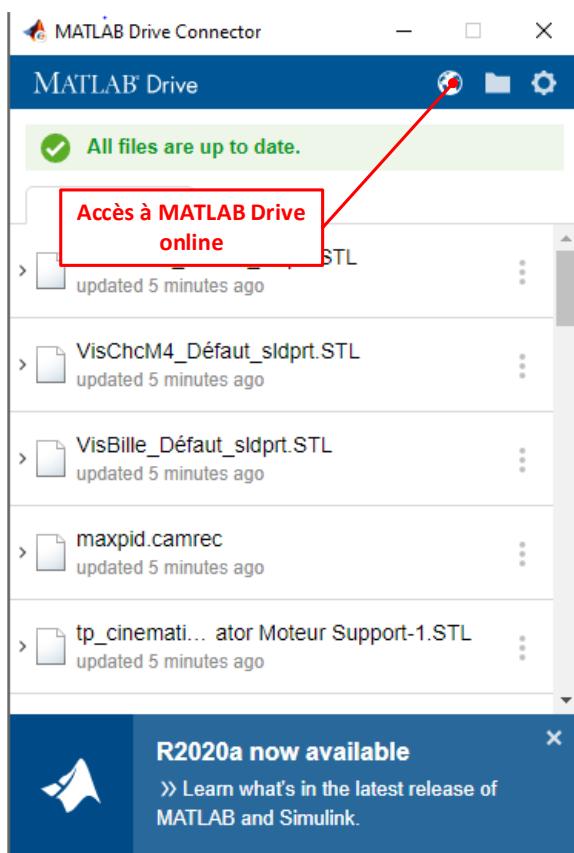


Figure 814 : lancement de MATLAB Drive online

MATLAB Drive online permet d'afficher l'ensemble des dossiers de **MATLAB Drive**.

Cliquer avec le bouton droit de la souris sur le dossier que vous voulez partager et sélectionner **Share/Invite Members** ou **Share/Create Link**. Le reste de la procédure est identique au partage en utilisant la version installée de MATLAB.

The screenshot shows the MATLAB Drive online interface. On the left, there's a sidebar with 'Files', 'Shared Content', and 'Deleted Files'. The main area is titled 'MATLAB Drive' and contains a table of files and folders. A context menu is open over a folder named 'Maxpid' in the 'My_models' folder. The menu options are: Share (highlighted), Rename (F2), Create Link (selected), Invite Members, Move to (Ctrl+X), Copy to (Ctrl+C), Delete, and Upload. The table data is as follows:

Name	Size	Date Modified	Owned By
MobileSensorData		29/07/2020 19:15	Me
My_models		29/07/2020 19:53	Me
Maxpid		11/06/2020 10:44	Me
Published		28/01/2015 12:36	Me
Shared		29/07/2020 13:51	Me
test_trace	1 KB	18/07/2020 09:51	Me
untitled2.n	1 KB	29/07/2020 14:45	Me

Figure 815 : partage d'un dossier à partir de MATLAB Drive online

Chapitre 13 : MATLAB Mobile

I. Présentation

MATLAB peut également être utilisé à partir d'un smartphone en téléchargeant l'application **MATLAB Mobile**. Cette application permet :

- de profiter de toutes les fonctionnalités de MATLAB pour le calcul scientifique (création de variables, opérations sur les variables, tracés de courbes, calcul symbolique, résolution d'équation différentielles...)
- de monitorer les mesures issues des capteurs du téléphone (accéléromètre, capteur de champ magnétique, capteur d'orientation, gyroscope, capteur de position terrestre...). Le téléphone peut être utilisé comme capteur pour de nombreuses expériences scientifiques
- d'exécuter des scripts et de visualiser les résultats

Comme pour **MATLAB online**, la capacité de calcul est déportée vers des serveurs et le smartphone joue le rôle d'un simple afficheur, il n'est donc pas nécessaire de disposer d'un smartphone performant pour pouvoir utiliser l'application mobile. Pour pouvoir utiliser l'application mobile, il suffit de disposer d'un compte Mathworks, aucune licence n'est requise et l'application mobile est libre d'utilisation. **MATLAB Mobile** permet d'accéder au contenu de **MATLAB Drive** dans lequel l'utilisateur peut stocker des fichiers créés à partir de **MATLAB online** ou de la version de MATLAB installée sur un ordinateur. L'interaction entre **MATLAB Mobile**, **MATLAB online** et **MATLAB Drive** permet à l'utilisateur de disposer en permanence de l'ensemble de ses fichiers de travail depuis toutes les plates formes (Figure 816).

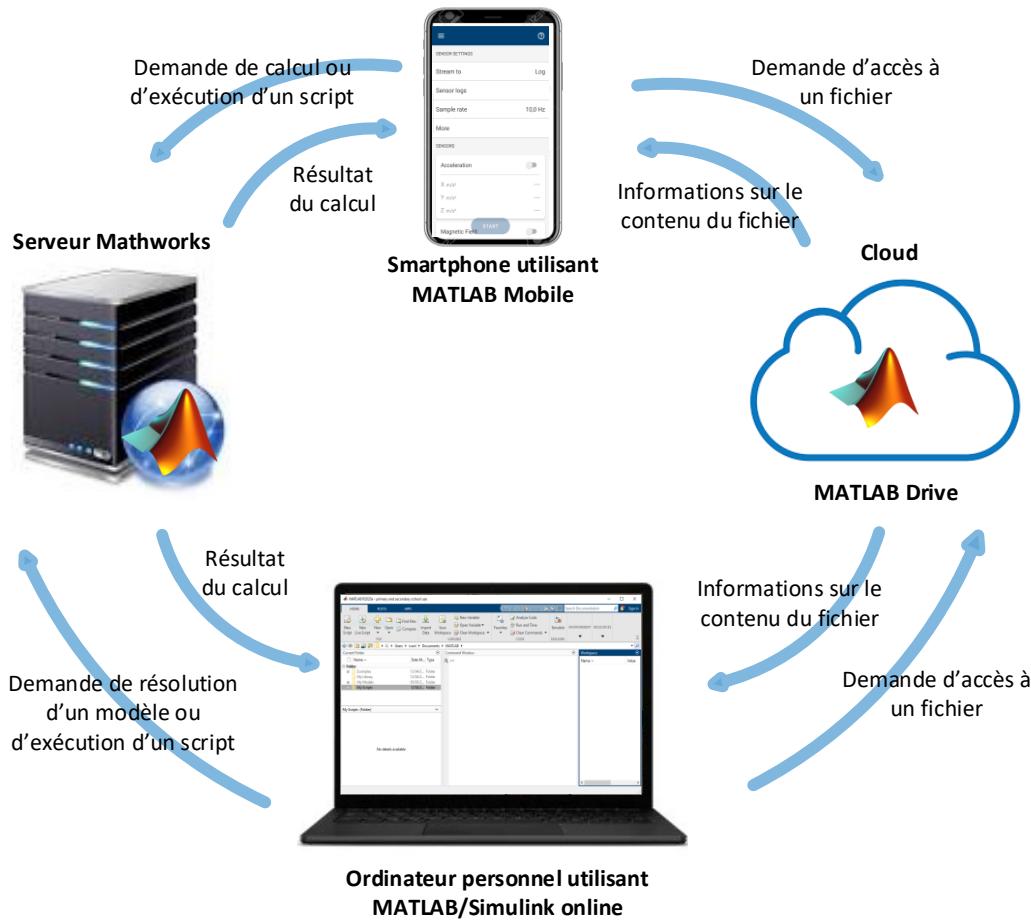


Figure 816 : MATLAB Mobile

II. Les fonctionnalités de MATLAB Mobile

1. Taper des lignes de commandes

Au lancement de l'application, la fenêtre de la Figure 817 apparaît. Il est alors possible de saisir des commandes dans la fenêtre de commande et de visualiser les résultats obtenus.

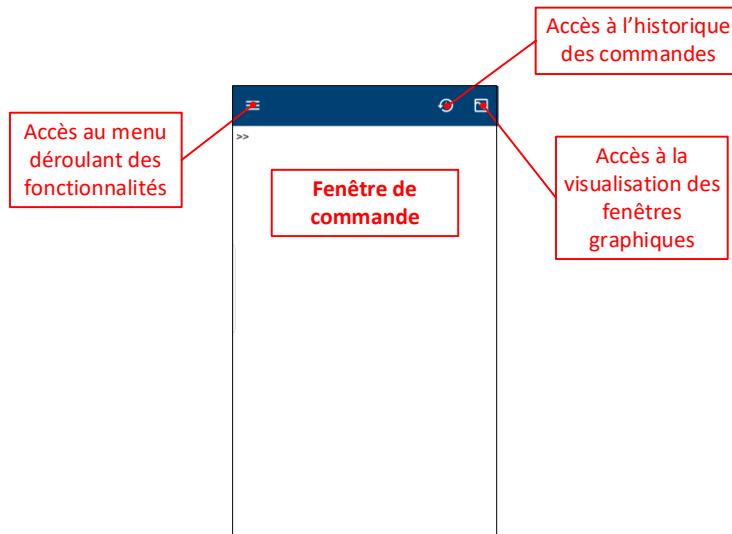


Figure 817 : ouverture de MATLAB Mobile

Taper les lignes de commandes de la Figure 818 dans l'application et observer le résultat obtenu.

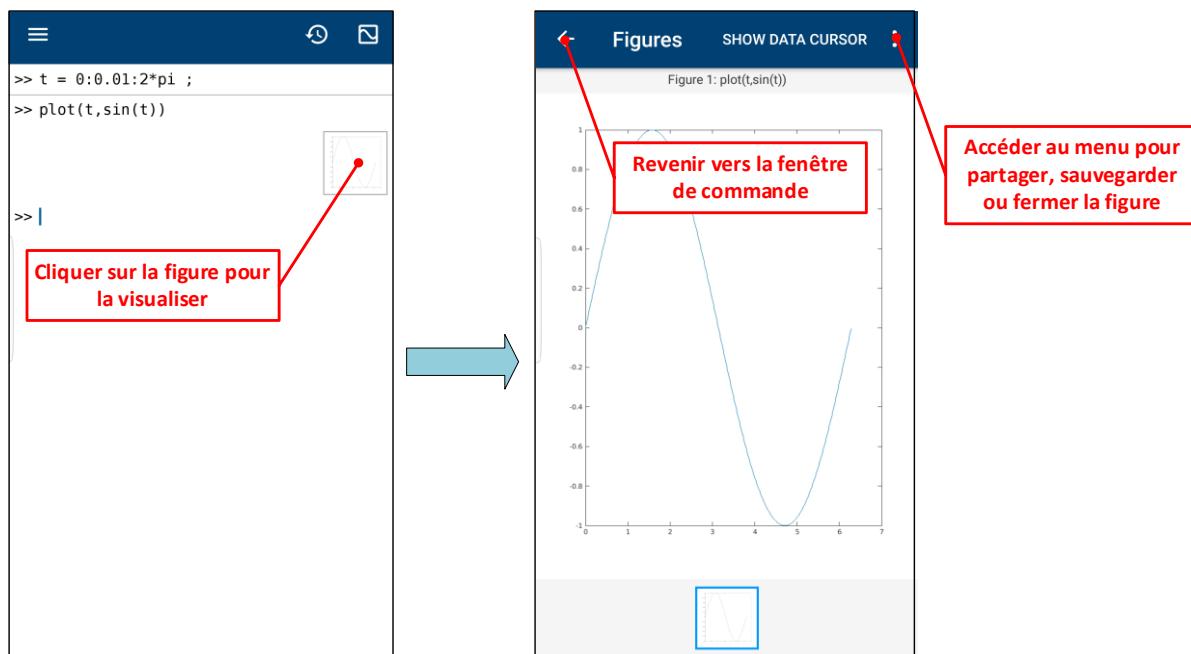
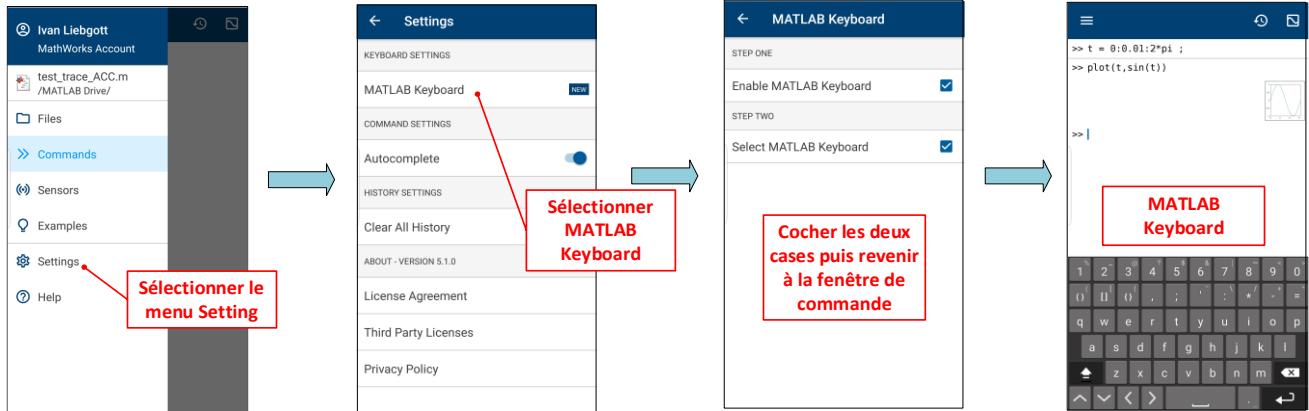


Figure 818 : utilisation des lignes de commandes dans MATLAB Mobile

2. Configurer le clavier MATLAB Keyboard

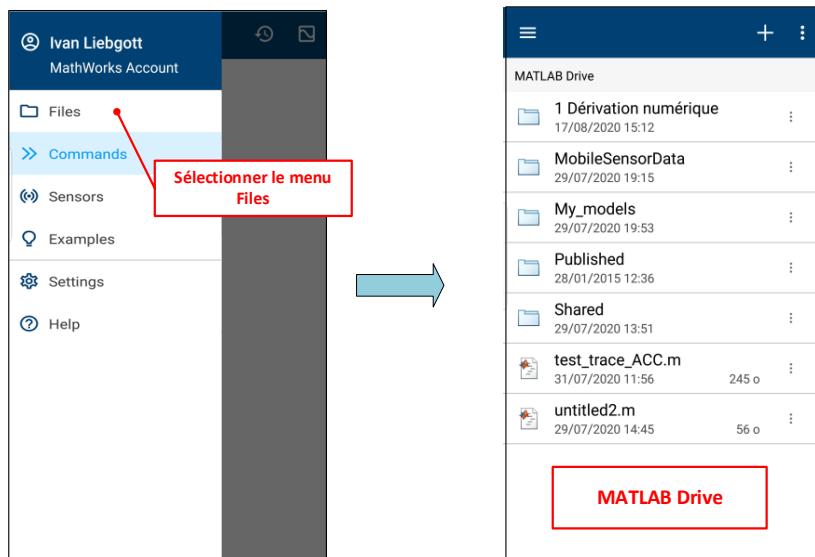
Afin de saisir plus facilement les lignes de commandes, il est possible d'utiliser un clavier spécialement dédié à l'utilisation de MATLAB. Pour cela suivre la procédure indiquée sur la Figure 819.



Le clavier qui apparaît maintenant est parfaitement adapté à la saisie des lignes de commandes, il est conseillé de l'utiliser afin de gagner du temps dans le processus de saisie.

3. Accès à MATLAB Drive

Afin d'accéder à MATLAB Drive depuis MATLAB Mobile, suivez la procédure indiquée sur la Figure 820.



Il est alors possible de naviguer dans les dossiers et d'accéder à tous les fichiers de MATLAB Drive.

4. Exécuter un script depuis MATLAB Mobile

Le dossier « 1 Dérivation Numérique » du chapitre « Ingénierie Numérique avec MATLAB » a été placé dans MATLAB Drive, il est donc visible depuis l'application mobile. Naviguer dans les dossiers pour ouvrir le script **num_deriv_plot_error.m** puis exécuter le script et visualiser les résultats conformément à la Figure 821.

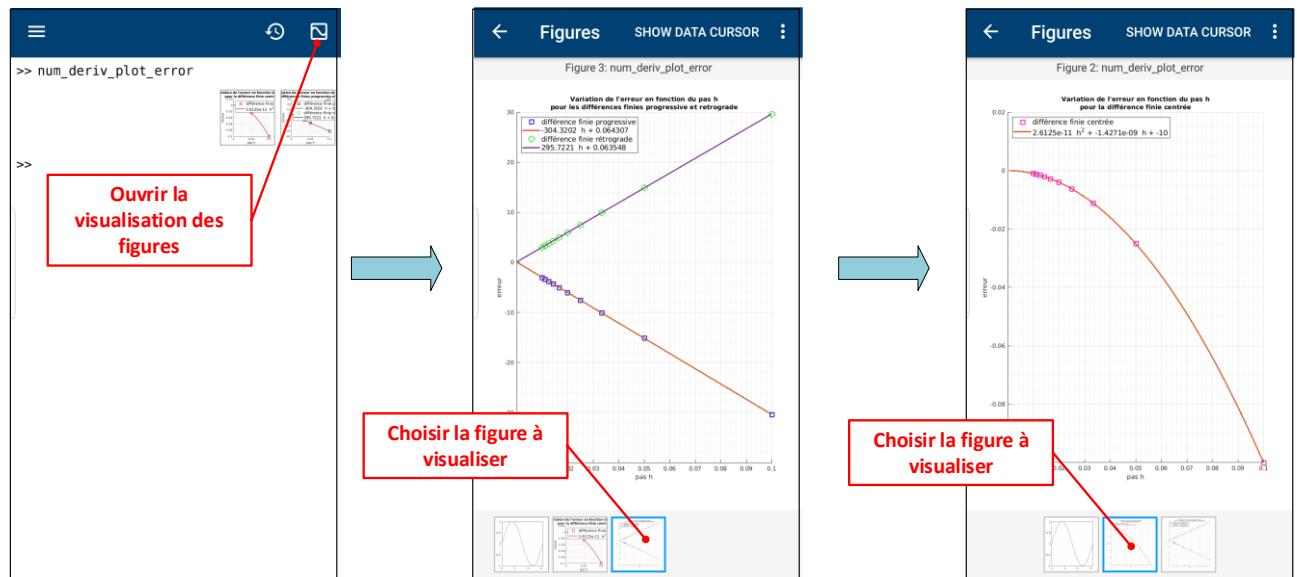
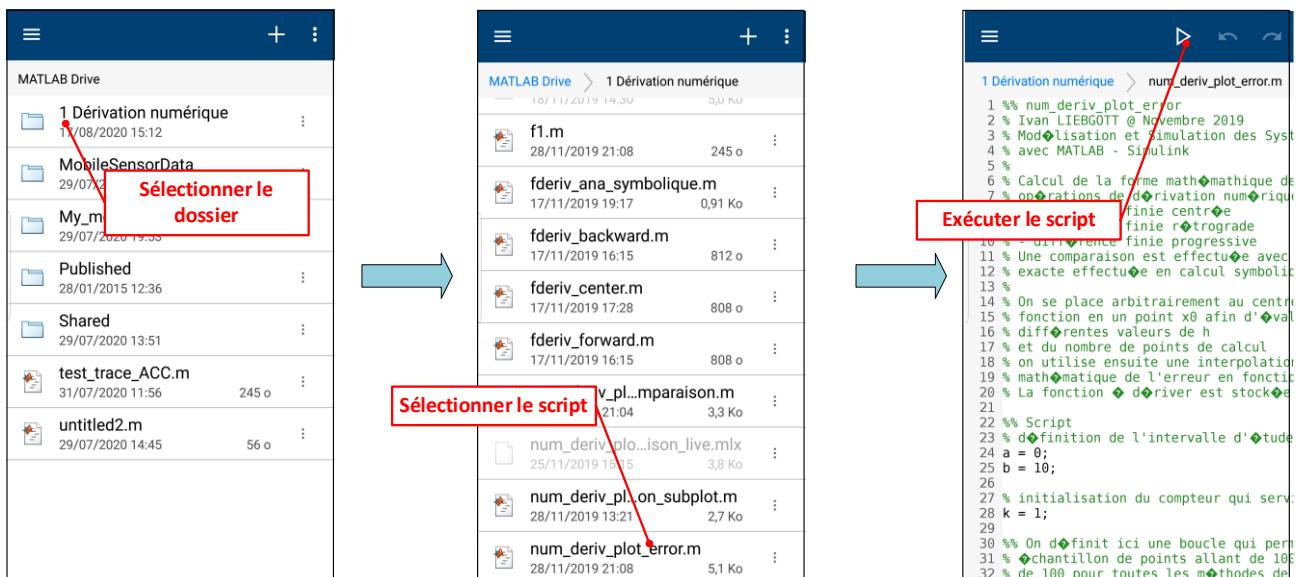


Figure 821 : exécution d'un script depuis MATLAB Mobile

5. Acquisition des données des capteurs du smartphone avec MATLAB Mobile

MATLAB Mobile permet de relever et d'exploiter très facilement les données issues des capteurs intégrés dans le smartphone :

- Accéléromètre 3 axes
- capteur de champ magnétique 3 axes
- capteur d'orientation (tangage, roulis, lacet)
- gyroscope 3 axes
- capteur de position terrestre (latitude, longitude, vitesse, altitude...)

Pour relever une série de mesure, suivez la procédure indiquée sur la Figure 822.



Figure 822 : configuration et démarrage d'une acquisition des mesures issues des capteurs du smartphone

En faisant défiler la fenêtre des capteurs, il est possible d'activer les autres capteurs (ce ne sera pas utile pour l'instant). Après avoir appuyé sur « START », les données des capteurs activés sont stockées dans des variables.

Réaliser alors une série de mouvements avec le smartphone puis appuyer sur « STOP » pour arrêter le processus d'acquisition. Indiquer alors le nom du fichier « **test_mesures_acc_3_axes** » qui sera automatiquement sauvégarde dans le dossier **MATLAB Drive /MobileSensorData/test_mesures_acc_3_axes.mat**. Le fichier porte l'extension **.mat** qui indique qu'il contient des variables que l'on pourra visualiser dans le Workspace de MATLAB en utilisant la commande `load('test_mesures_acc_3_axes.mat')`.

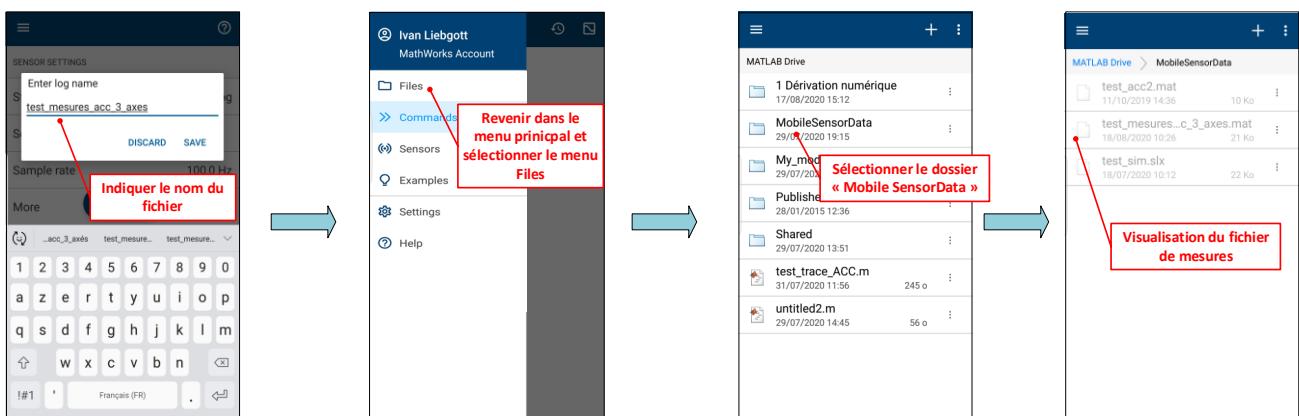


Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive

A partir du menu Files, il est possible de voir le fichier sauvegardé dans MATLAB Drive. Le fichier est alors instantanément accessible à partir de MATLAB online ou de la version de MATLAB installée sur votre PC

6. Exploitation des données des capteurs du smartphone avec MATLAB Mobile

L'exploitation des données doit se faire à partir de MATLAB online ou de la version de MATLAB installée sur votre PC. La condition nécessaire est de pouvoir accéder à MATLAB Drive.

Ouvrir MATLAB ou MATLAB online, sur votre PC et ouvrir le dossier **MATLAB Drive /MobileSensorData**. Observez l'apparition du fichier de mesure dans MATLAB Drive (Figure 824).

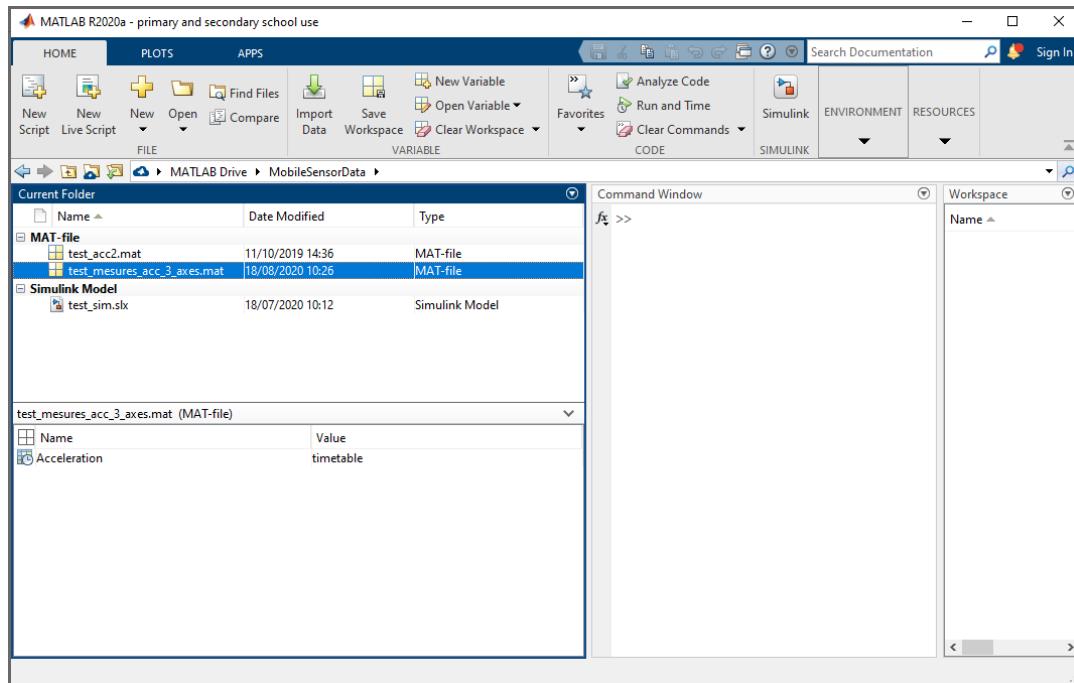


Figure 824 : visualisation du fichier de mesure sur le PC

Dans la fenêtre de commande, taper la commande suivante puis valider :

```
>> load test_mesures_acc_3_axes.mat
```

La variable « Acceleration » apparaît alors dans le Workspace et peut être exploité.

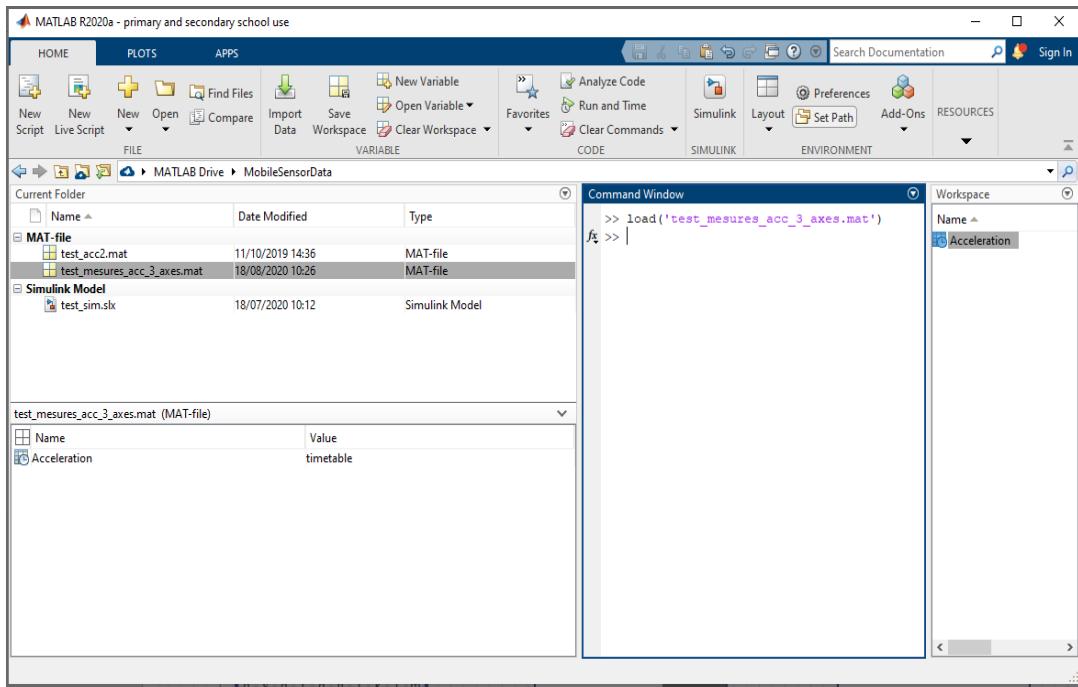


Figure 825 : visualisation du fichier de mesure dans le Workspace

La variable « **Acceleration** » porte par défaut le nom du capteur qui a permis de créer la variable. Si nous avions activé d'autres capteurs durant l'acquisition, chaque capteur activé aurait générer la création d'une variable associée. Dans la fenêtre de commande, taper « **Acceleration** » afin de visualiser la structure de la variable (Figure 826).

```
>> Acceleration
```

```
Acceleration =
844x3 timetable
    Timestamp          X          Y          Z
    _____
    18-Aug-2020 10:25:53.017 -0.17478   6.1509   8.2626
    18-Aug-2020 10:25:53.026 -0.3328    6.4861   8.3057
    18-Aug-2020 10:25:53.035 -0.4166    6.6297   7.9681
    18-Aug-2020 10:25:53.044 -0.64885   7.1278   7.5084
    18-Aug-2020 10:25:53.053 -1.4102    7.5563   6.9554
    18-Aug-2020 10:25:53.062 -1.3695    7.9131   6.3113
    18-Aug-2020 10:25:53.071 -1.4749    8.2555   5.8037
    18-Aug-2020 10:25:53.080 -1.0942    7.9634   5.7391
    18-Aug-2020 10:25:53.089 -0.38069   7.4606   5.8013
    18-Aug-2020 10:25:53.098  0.0838    7.2403   5.9426
    18-Aug-2020 10:25:53.107  0.22506   7.0368   6.1341
    18-Aug-2020 10:25:53.116  0.29689   6.9458   6.6106
    18-Aug-2020 10:25:53.125  0.38069   6.7782   7.0296
    18-Aug-2020 10:25:53.134  0.31126   6.7495   7.099
    18-Aug-2020 10:25:53.143  0.13887   6.692    6.941
    18-Aug-2020 10:25:53.152  0.079011  6.7878   6.8716
    18-Aug-2020 10:25:53.161 -0.10535   6.9266   6.8333
    18-Aug-2020 10:25:53.170 -0.61772   7.0272   6.6968
    18-Aug-2020 10:25:53.179 -0.99123   7.032    6.6896
    18-Aug-2020 10:25:53.188 -1.0702    7.0009   6.7854
```

Figure 826 : structure de la variable Acceleration

La variable **Acceleration** est de type « **timetable** » et contient les accélérations sur les 3 axes associées aux différents instants de l'acquisition.

Pour extraire la colonne des instants taper la commande suivante (Figure 827) :

```
>>t = Acceleration.Timestamp
```

```
t =  
  
844×1 datetime array  
  
18-Aug-2020 10:25:53.017  
18-Aug-2020 10:25:53.026  
18-Aug-2020 10:25:53.035  
18-Aug-2020 10:25:53.044  
18-Aug-2020 10:25:53.053  
18-Aug-2020 10:25:53.062  
18-Aug-2020 10:25:53.071
```

Figure 827 : extraction de la colonne des instants

La variable « **t** » de type **datetime** apparaît alors dans le Workspace.

Il est possible d'extraire la colonne correspondant uniquement aux secondes en tapant la commande suivante (Figure 828) :

```
>>t_sec = t.Second
```

```
t_sec =  
  
53.0170  
53.0260  
53.0350  
53.0440  
53.0530  
53.0620  
53.0710  
53.0800  
53.0890  
53.0980
```

Figure 828 : extraction de la colonne des instants correspondant aux secondes

La variable **t_sec** de type double apparaît alors dans le Workspace et pourra être utilisée pour tracer une courbe.

Pour extraire la colonne correspondant aux accélérations relevées selon l'axe X, taper la commande suivante (Figure 829):

```
>>AX = Acceleration.X
```

```

AX =
-0.1748
-0.3328
-0.4166
-0.6488
-1.4102
-1.3695
-1.4749
-1.0942
-0.3807
0.0838
0.2251
0.2969

```

Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X

Saisir le script de la Figure 830 et lancer son exécution. Le script est également disponible dans le dossier « Chapitre_12_MATLAB_Mobile ».

```

%% Exploitation des données issues de l'accéléromètre d'un smartphone
% Ivan LIEBGOTT @ aout 2020
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%%
load test_mesures_acc_3_axes.mat
close all;

% création des variables permettant d'effectuer les tracés
t = Acceleration.Timestamp;
AX = Acceleration.X;
AY = Acceleration.Y;
AZ = Acceleration.Z;

% tracés des courbes d'accélération
subplot(3,1,1);
plot(t,AX);grid on;

subplot(3,1,2);
plot(t,AY);grid on;

subplot(3,1,3);
plot(t,AZ);grid on;

```

Figure 830 : script permettant de faire le tracer les accélérations en fonction du temps

Le résultat de l'exécution du script peut être visualisé sur la Figure 831.

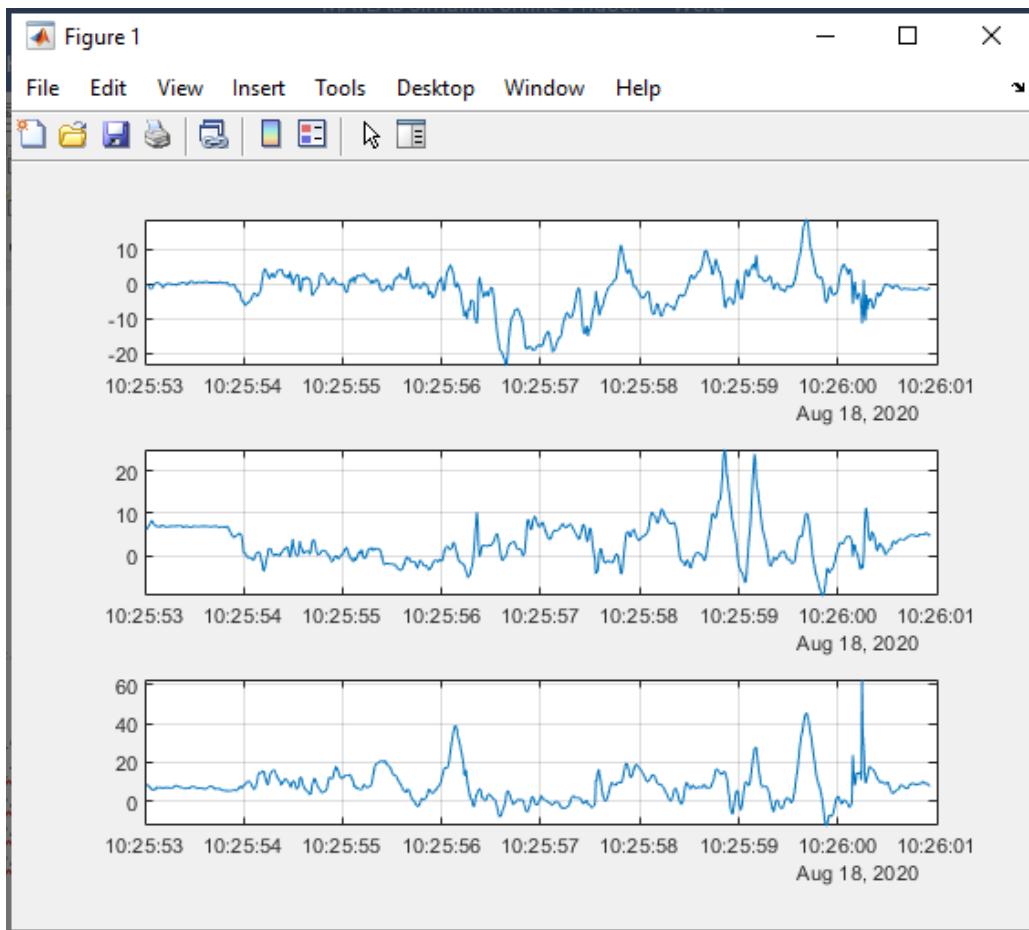


Figure 831 : accélérations selon les trois axes en fonction du temps

Il est maintenant possible d'utiliser cette série de données pour lui appliquer un ou plusieurs processus de traitement et d'analyse décrits dans le chapitre 10 « Ingénierie Numérique en langage MATLAB » (filtrage, intégration numérique...).

Annexe 1 : paramétrage des scopes

Ouvrir le fichier « **circuit_RL.slx** » et lancer la simulation.

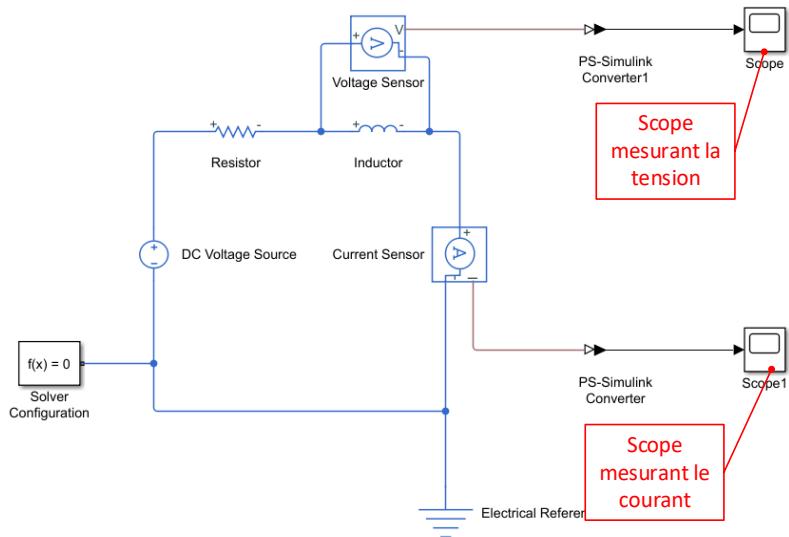


Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine

Ouvrir le scope mesurant la tension aux bornes de la bobine et visualiser l'allure et la mise en forme de la courbe.

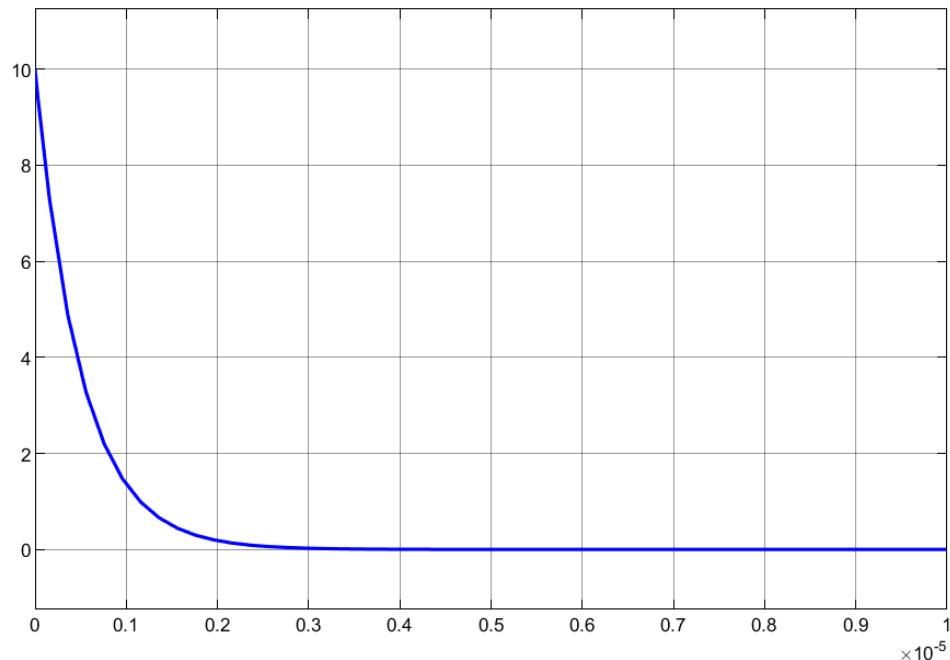


Figure 833 : mesure de la tension aux bornes de la bobine

Lors de l'ouverture du scope, la barre de commande est disponible sur la partie supérieure de la fenêtre (Figure 834).

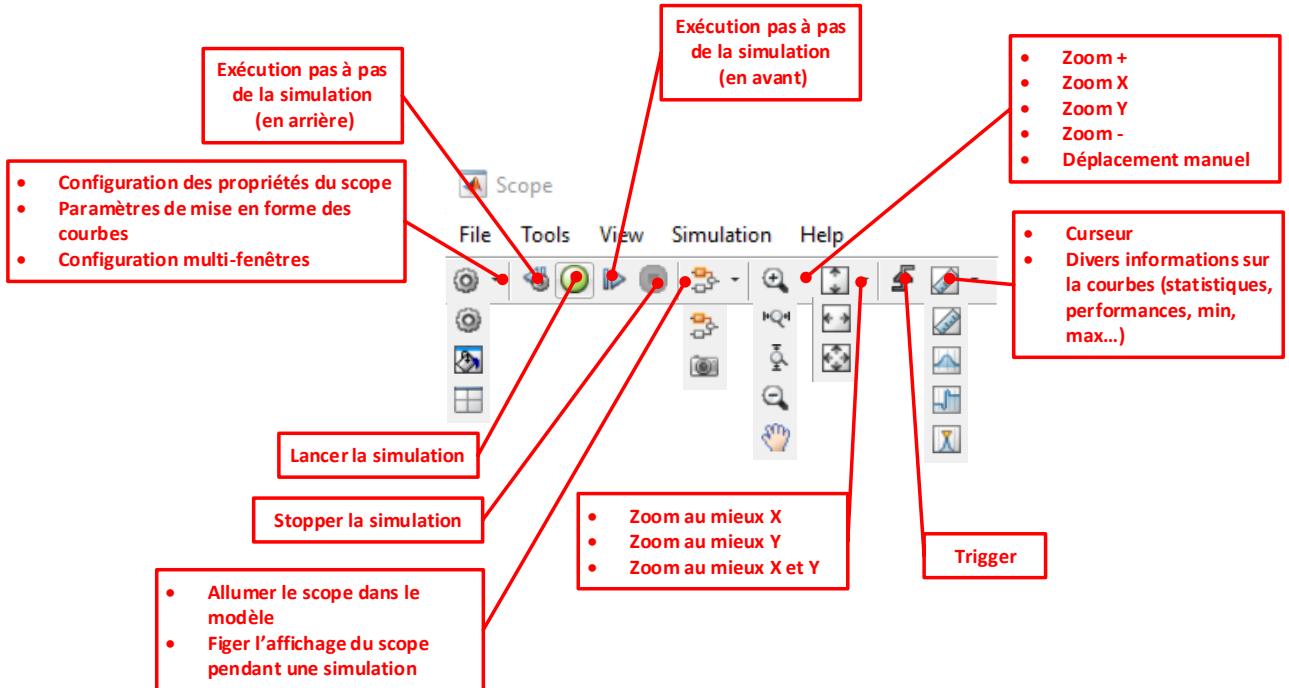


Figure 834 : barre de commande du scope

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 en forme des courbes en cliquant sur l'icône accessible depuis le menu déroulant **Configuration Properties** .

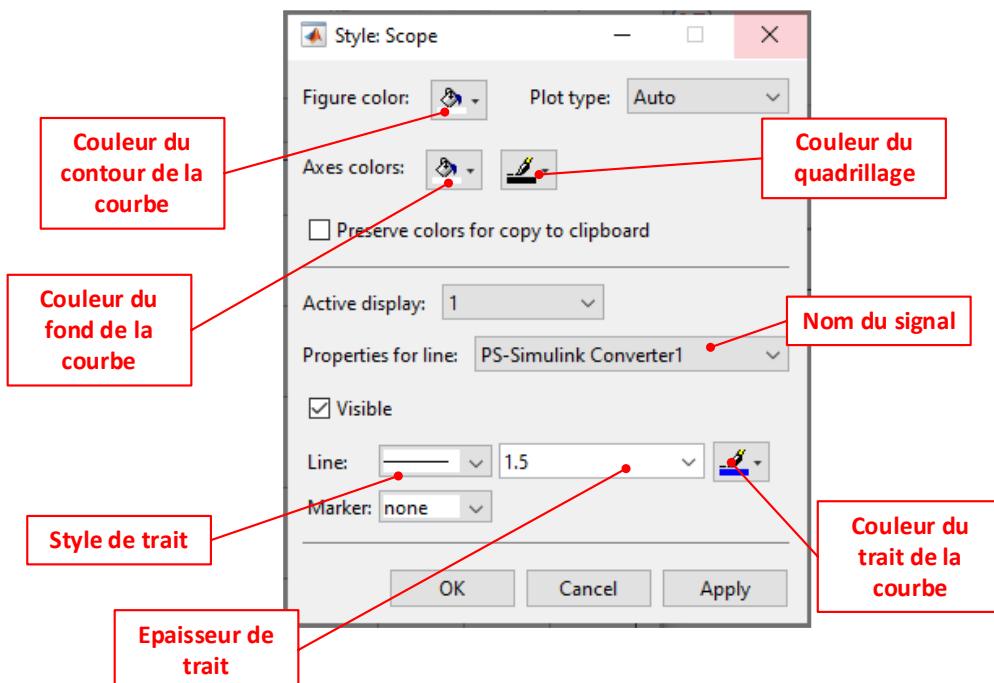


Figure 835 : mise en forme des courbes dans un scope

Modifier les paramètres de mise en forme comme indiqué sur la Figure 835 et observer le résultats obtenus (Figure 836).

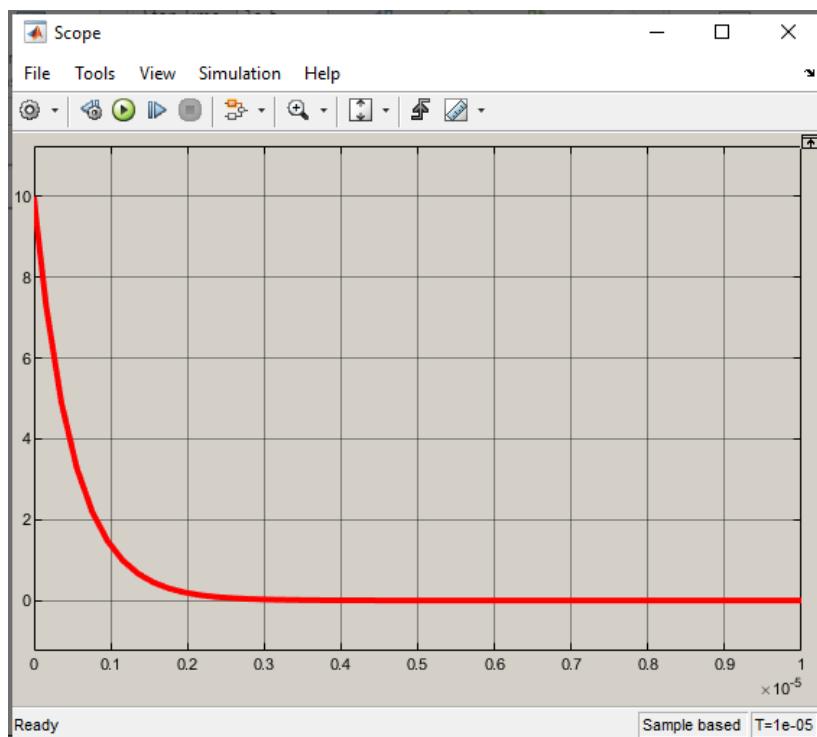


Figure 836 : modification de la mise en forme d'une courbe dans un scope

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 837).

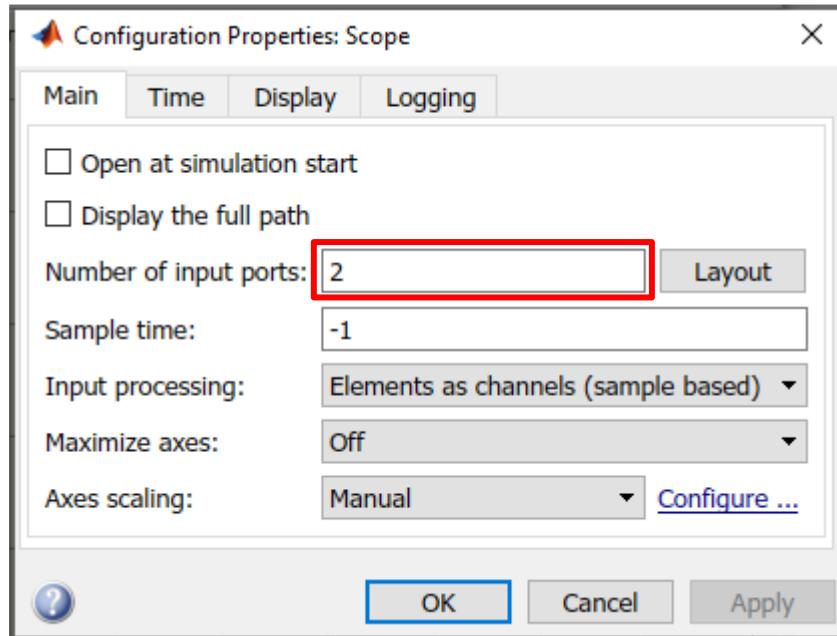


Figure 837 : modification du nombre d'entrées d'un scope

Le scope a maintenant 2 entrées. **Raccorder** la seconde entrée au signal mesurant le courant dans le circuit comme indiqué sur la Figure 838.

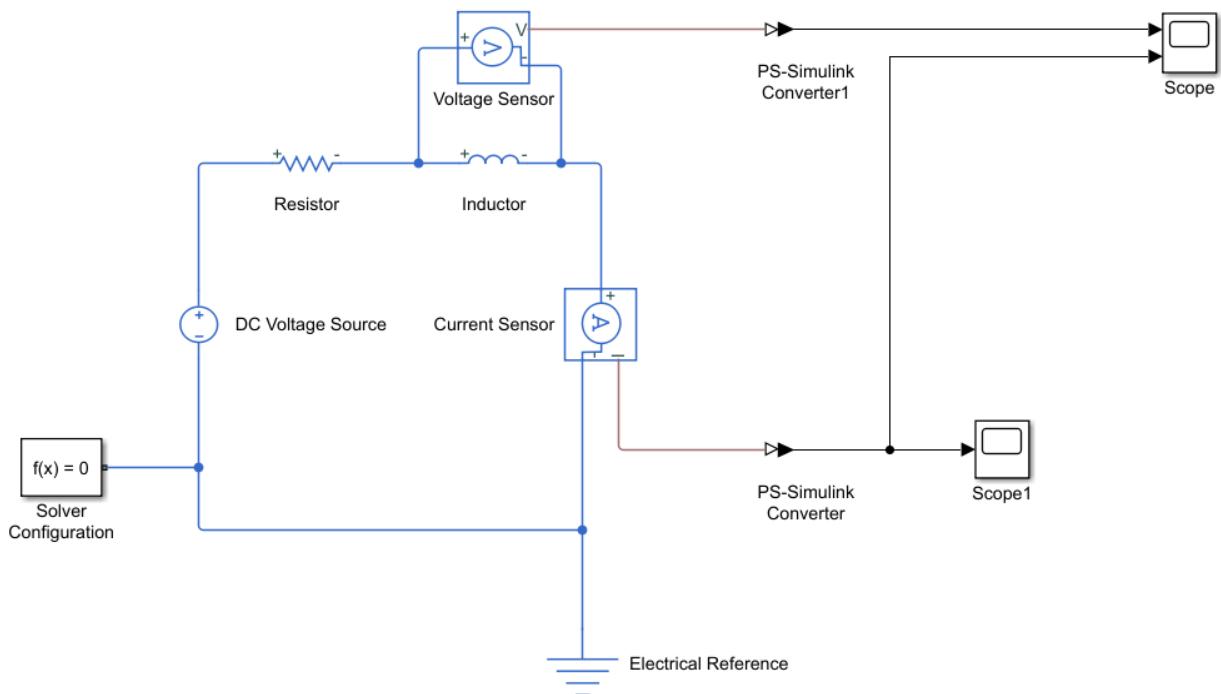


Figure 838 : raccordement d'un scope à deux entrées

Lancer la simulation et ouvrir le scope pour obtenir la courbe de la Figure 839.

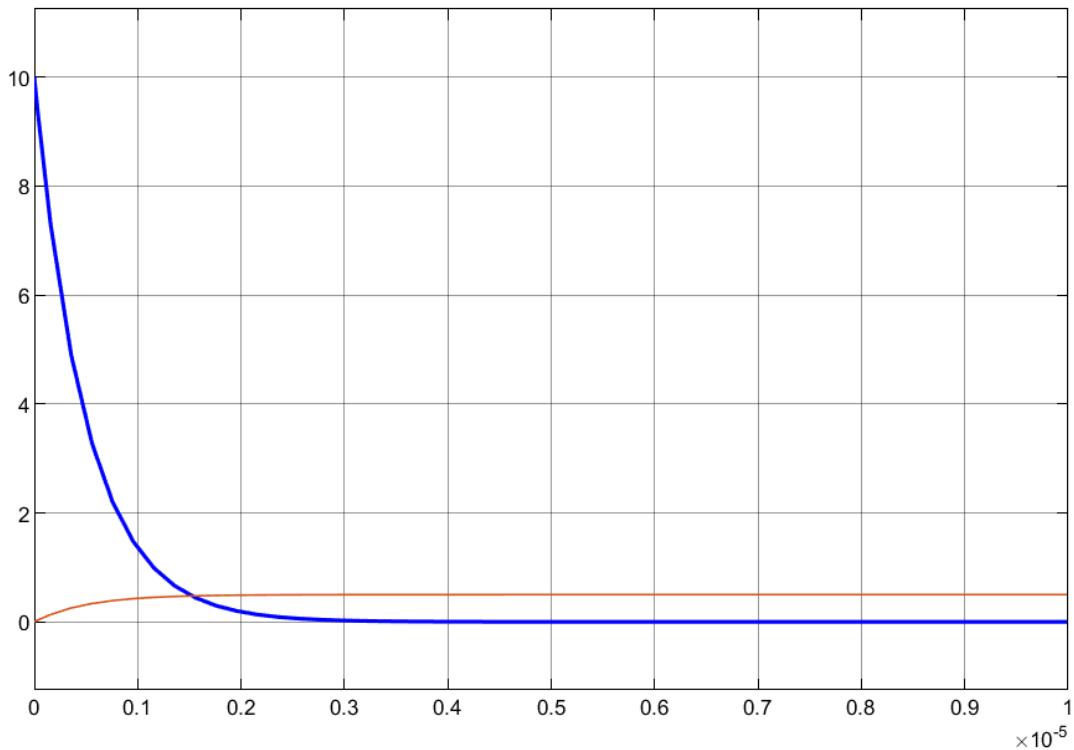


Figure 839 : visualisation des deux courbes dans le même scope

Modifier les caractéristiques de la seconde courbe en utilisant l'icône , comme indiqué sur la Figure 840.

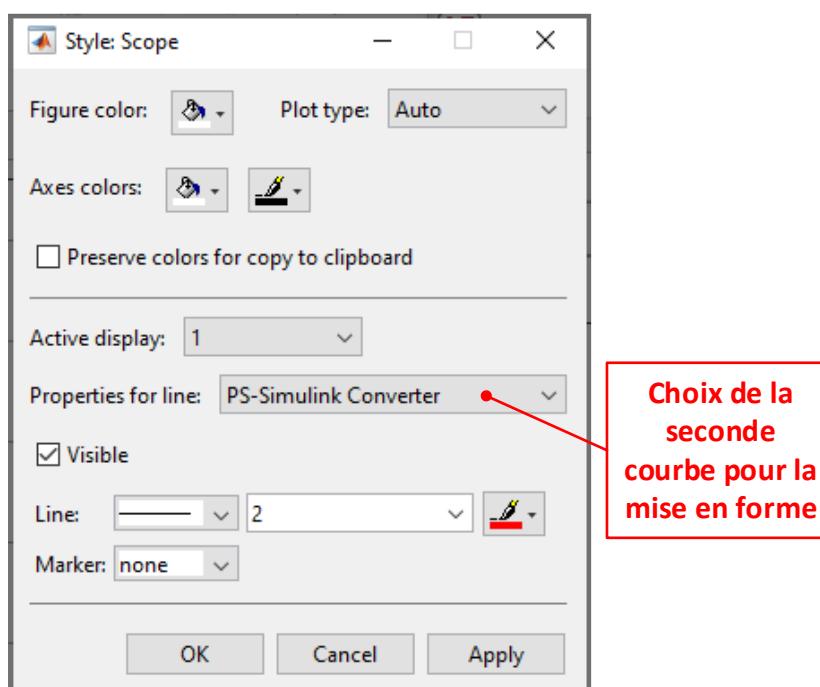


Figure 840 : modification des caractéristiques de la deuxième courbe

Vous devez obtenir la configuration de la Figure 841.

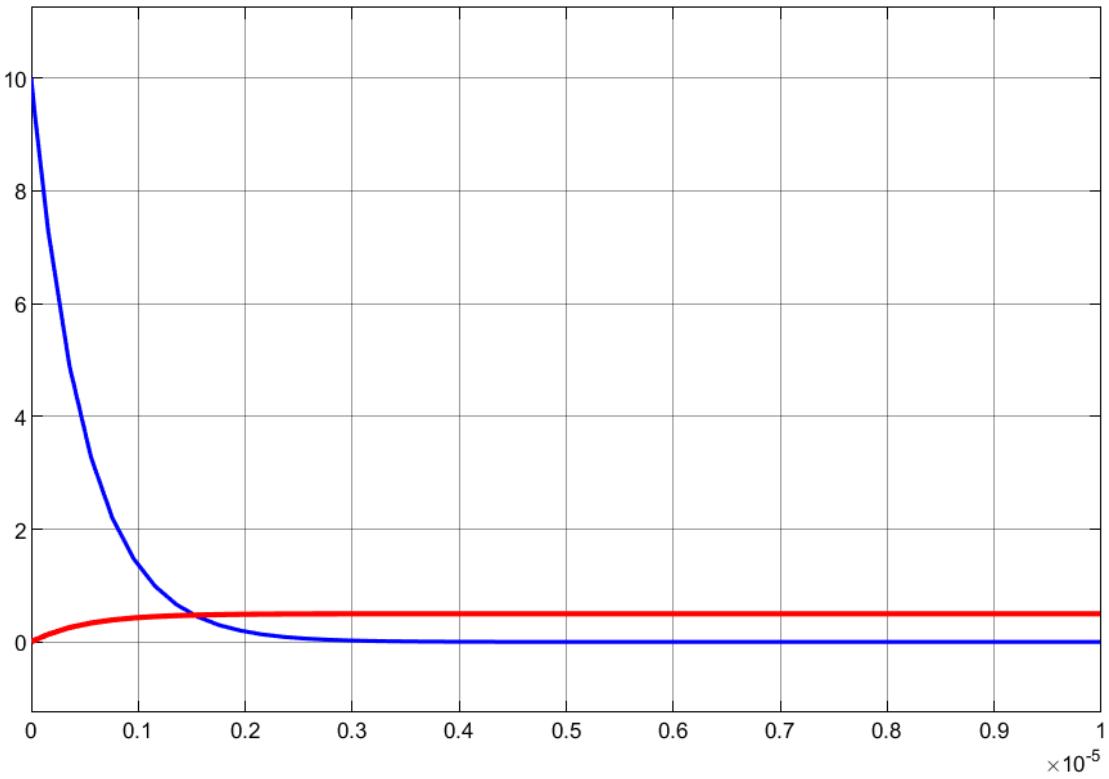


Figure 841 : modification des caractéristiques de la deuxième courbe

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 842).

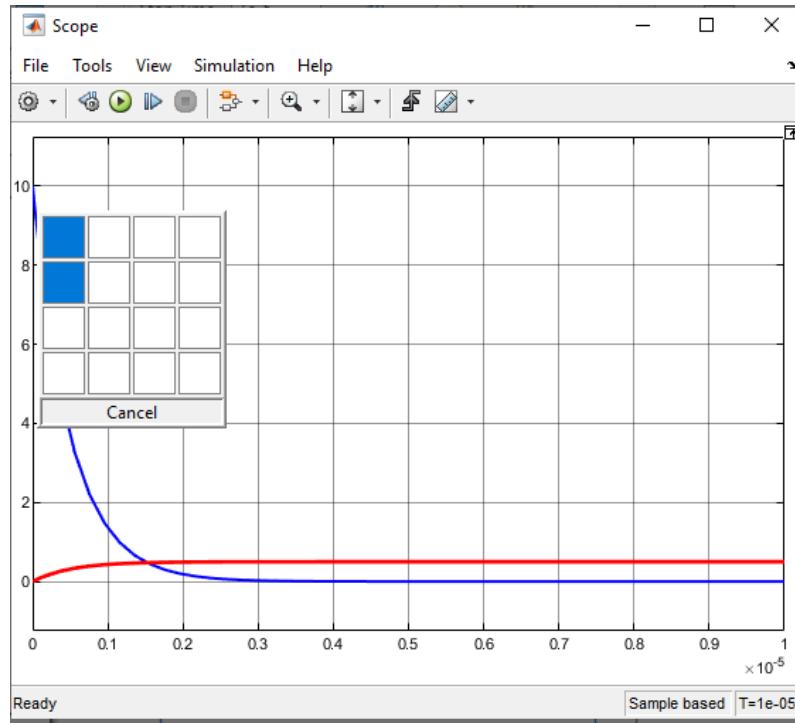


Figure 842 : modification du nombre de fenêtre dans le scope

Après mise à l'échelle des courbes à l'aide de la commande  , on obtient l'affichage représenté sur la Figure 843.

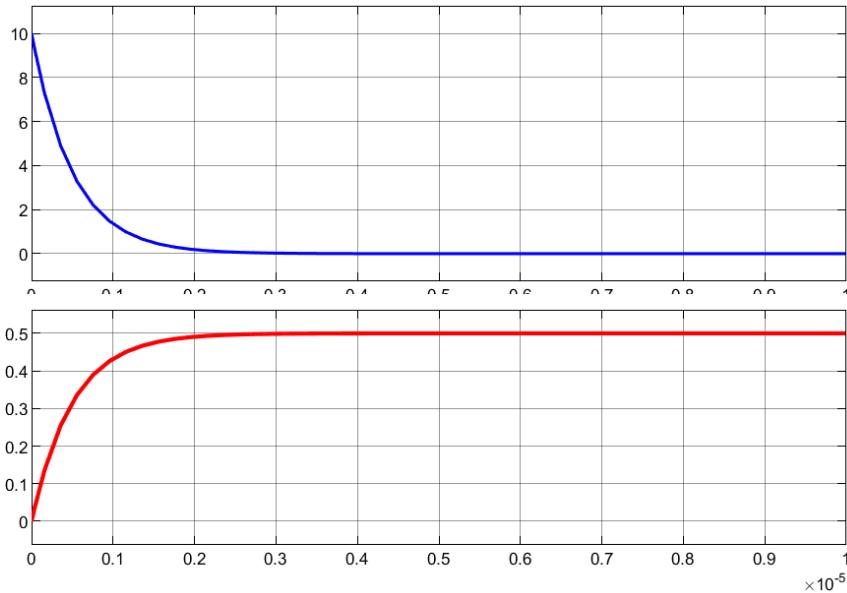


Figure 843 : affichage des courbes dans un scope sur plusieurs fenêtres

Un outil curseur est très pratique à utiliser dans les scopes, 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 844).

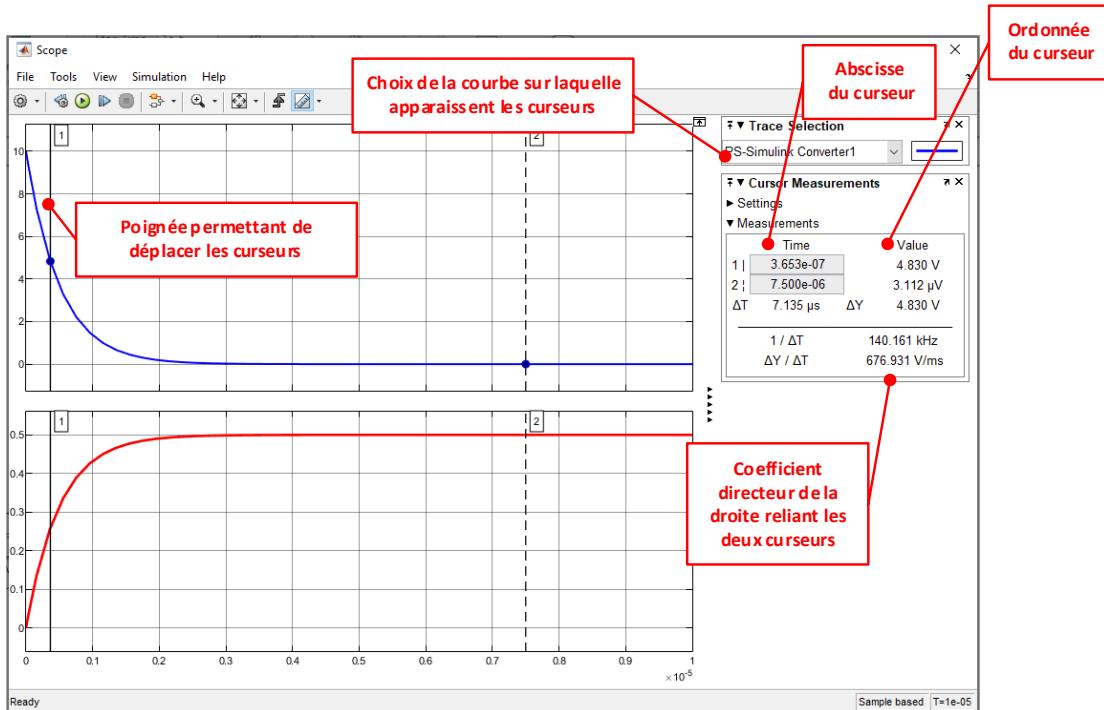
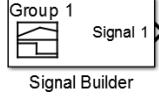


Figure 844 : utilisation des curseurs dans un scope

Pour plus d'informations sur l'utilisation des scopes vous pouvez utiliser l'aide de MATLAB.

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.

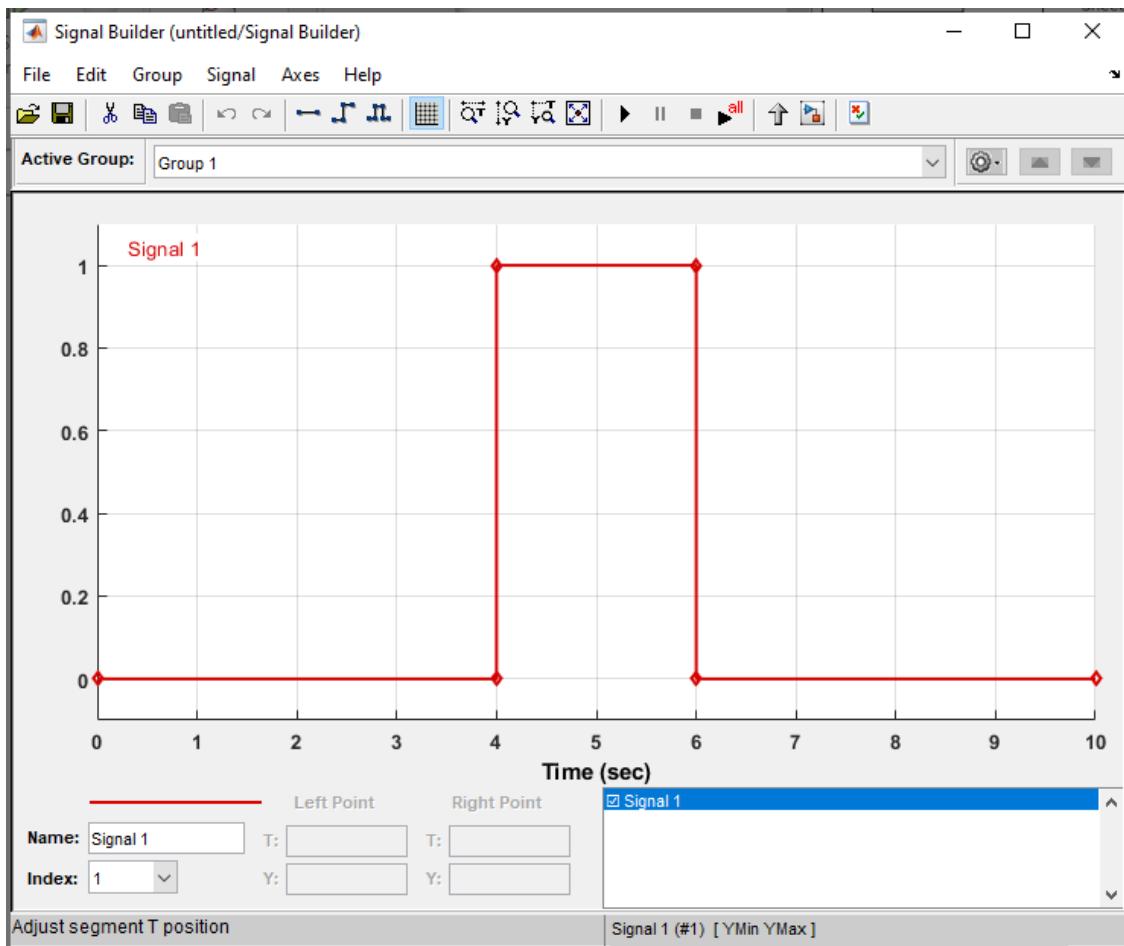


Figure 845 : fenêtre de paramétrage du Signal builder

La barre de commande du signal buider permet d'accéder à de nombreuses fonctions.

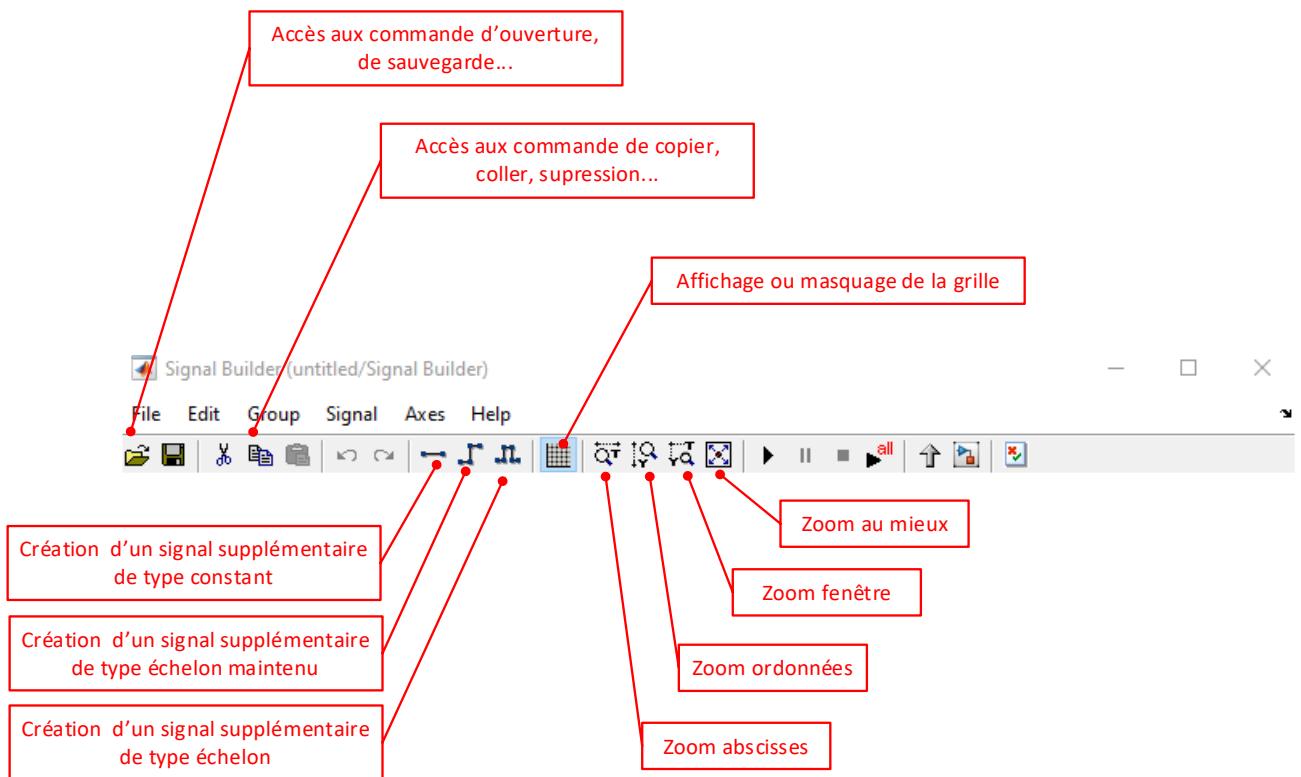


Figure 846 : barre de commande du signal builder

Dans un premier temps il faut indiquer la durée du signal. Pour cela sélectionner **Axes/Change Time Range** accessible depuis la barre de commande.

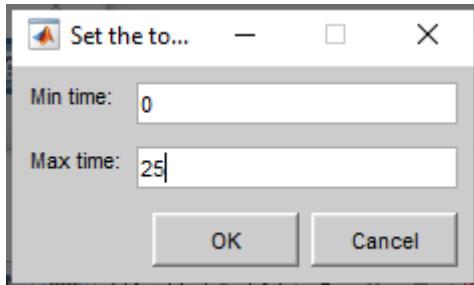


Figure 847 : choix de la durée du signal dans le « signal builder »

Choisir une durée de signal de 25 s (Figure 848).

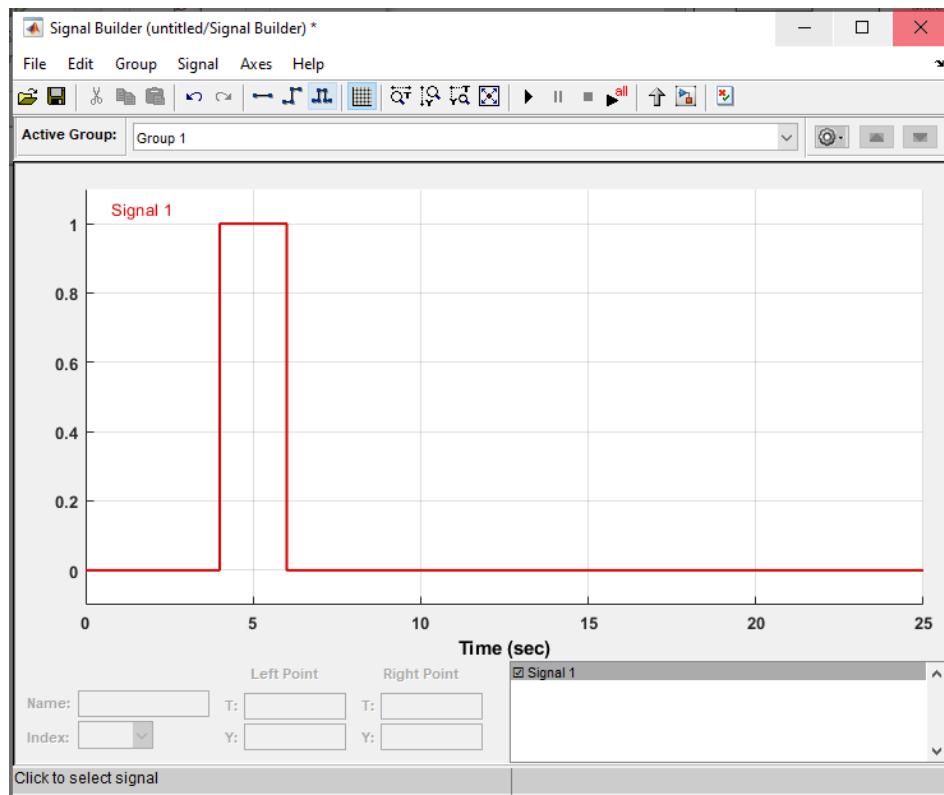


Figure 848 : modification de l'échelle de temps

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 (Figure 849).

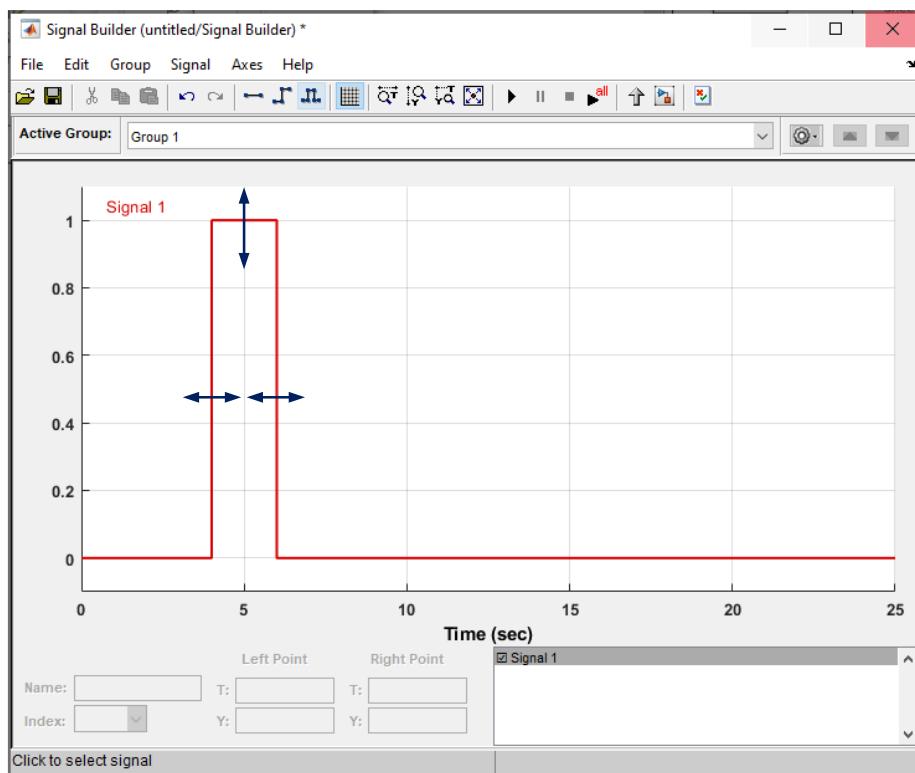


Figure 849 : déplacement manuel des segments dans le « signal builder »

Pendant le déplacement, il est possible de visualiser dans les cases « left point » et « right point » situées sous le signal la position du segment.

En sélectionnant un segment vertical (clique gauche avec la souris), il est possible de rentrer la valeur exacte de l'instant dans la case **T : « Left Point »**. Il est également possible de choisir exactement l'ordonnée du point d'origine et du point final dans les cases **Y : « Left Point » « Right Point »**. Les mêmes opérations peuvent être réalisées en sélectionnant un segment horizontal.

Il est également possible de déplacer uniquement un point pour obtenir des signaux de type rampe.

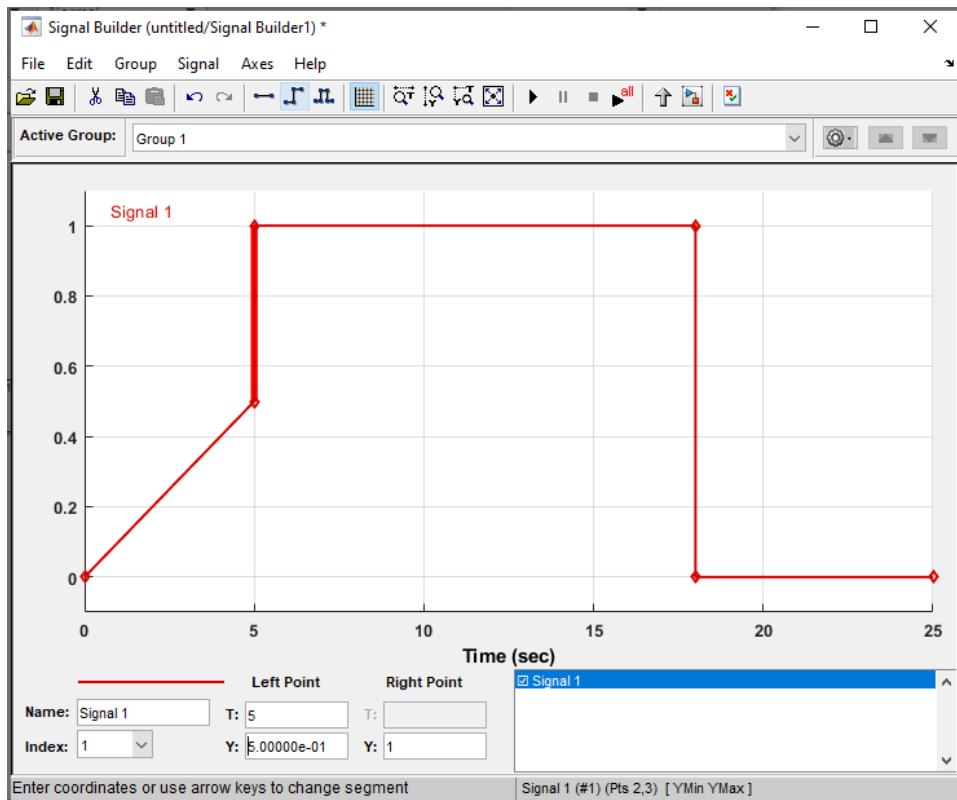


Figure 850 : déplacement d'un point dans le « signal builder »

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 en utilisant la méthode précédemment présentée.

Pour créer de nouveaux points, il faut utiliser la combinaison de touche **Maj+clique gauche** en plaçant les nouveaux points directement sur le signal.

Placer approximativement des nouveaux points conformément à la Figure 851.

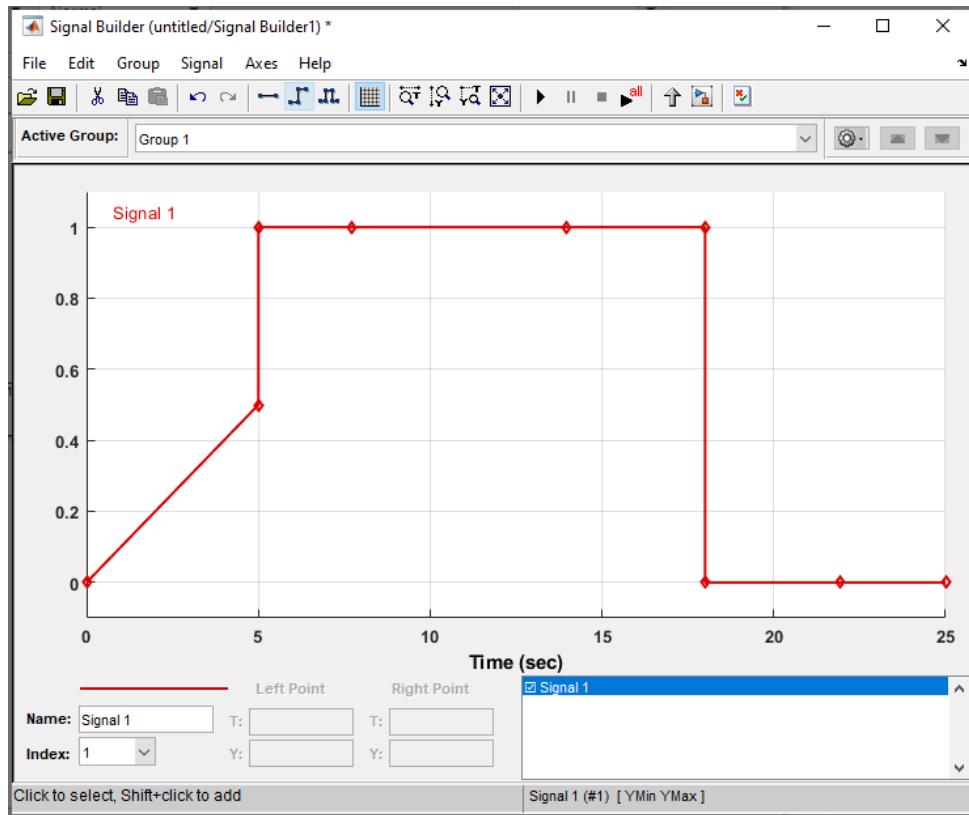


Figure 851 : placement de nouveaux points sur le signal

Déplacer les points et les segments pour obtenir approximativement le signal de la Figure 852.

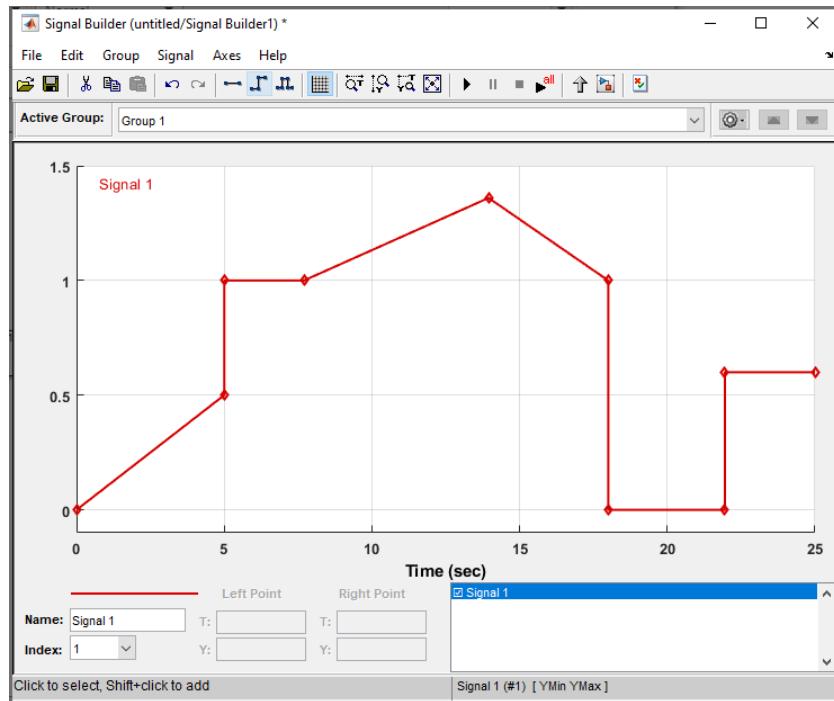


Figure 852 : mise en forme du signal par déplacement de segments et de points

Il est également possible de créer un bloc permettant de générer plusieurs signaux.

Cliquer sur les trois signaux  accessibles depuis la barre de commande et choisir d'introduire de nouveaux signaux.

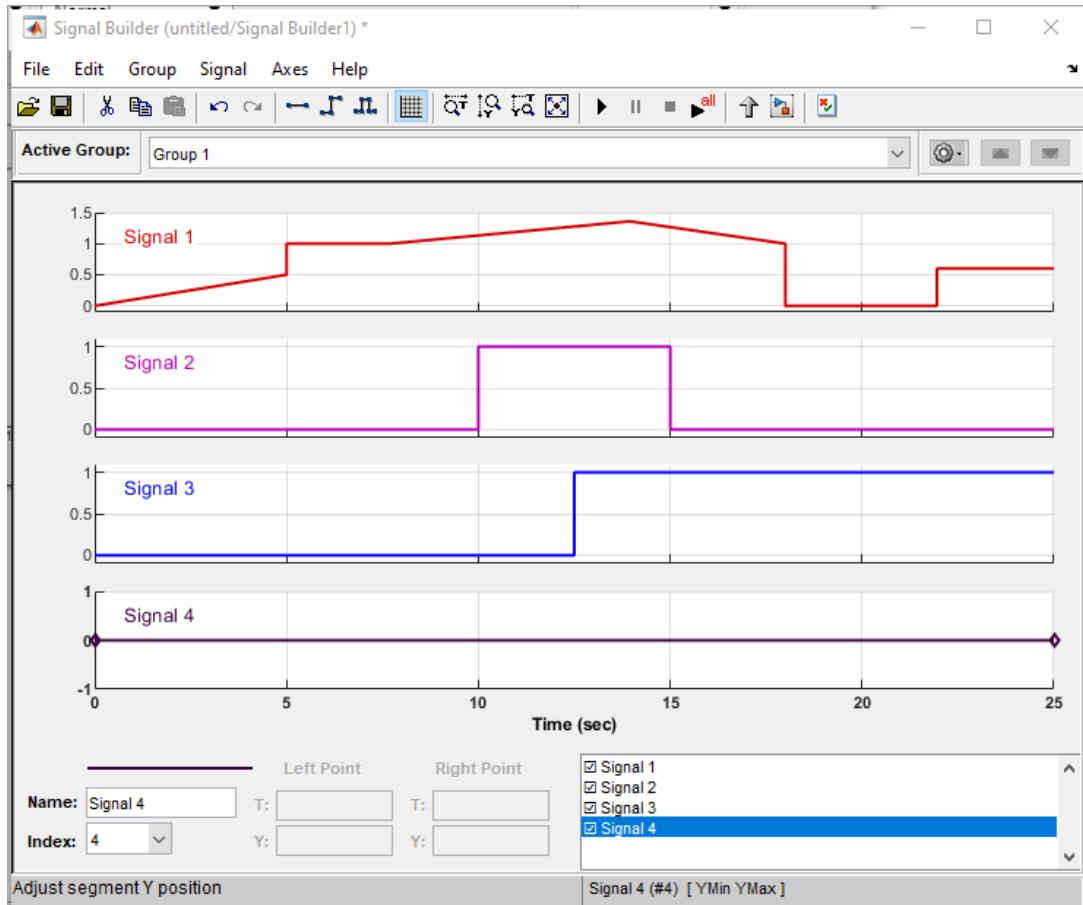


Figure 853 : génération de plusieurs signaux

Le bloc comporte maintenant 4 sorties.

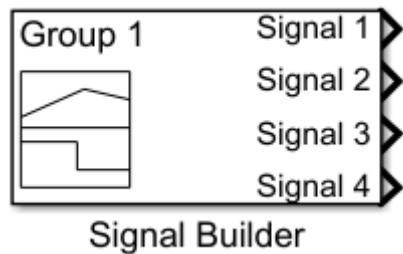


Figure 854 : visualisation des sorties multiples du « signal builder »

Table des figures

Figure 1 : cycle en V simplifié	20
Figure 2 : les compétences de l'ingénieur	21
Figure 3 : triptyque Cahier des charges/Système réel/Modèle	22
Figure 4 : architecture matérielle nécessaire à la mise en place de la démarche	23
Figure 5 : la phase d'expression et de spécification du besoin	24
Figure 6 : diagramme des exigences de l'asservissement de position	24
Figure 7 : la phase de Conception, Modélisation, Simulation	25
Figure 8 : relation entrée/sortie pour le moteur à courant continu	26
Figure 9 : équations de comportements du moteur à courant continu	26
Figure 10 : modélisation sous forme de schéma bloc du moteur à courant continu	27
Figure 11 : modélisation « white box » du moteur à courant continu	27
Figure 12 : modélisation par assemblage de composants du moteur à courant continu	28
Figure 13 : modélisation multi-phérique acausal du moteur à courant continu	28
Figure 14 : résolution des modèles et visualisation des résultats	29
Figure 15 : mesure de l'évolution de la vitesse de rotation du moteur en fonction du temps	29
Figure 16 : évaluation des écarts entre les performances simulée et les performances mesurées	30
Figure 17 : essai réalisé en vue de l'estimation des paramètres inconnus	30
Figure 18 : illustration du principe de l'estimation de paramètre	31
Figure 19 : les réponses du modèle avant et après estimation des paramètres inconnus	31
Figure 20 : modélisation de l'asservissement de position du moteur	32
Figure 21 : réponse de l'asservissement de position non corrigé	32
Figure 22 : illustration de l'utilisation d'un outil de contrôle commande pour régler un correcteur	33
Figure 23 : réponse de l'asservissement de position corrigée	33
Figure 24 : le Model-in-the-loop (MIL)	34
Figure 25 : la phase de Codage Implémentation	34
Figure 26 : structure du modèle à l'issu de la phase de Conception Modélisation Simulation	35
Figure 27 : le Software-in-the-loop(SIL)	36
Figure 28 : le Processor-in-the-loop	36
Figure 29 : la phase d'Intégration Vérification	37
Figure 30 : architecture matérielle du hardware-in-the-loop en mode externe	37
Figure 31 : le hardware-in-the-loop en mode externe: intégration du matériel dans la boucle	38
Figure 32 : architecture matérielle du Hardware-In-the-Loop en mode embarqué	38
Figure 33 : le Hardware-In-the-Loop mode embarqué: fonctionnement autonome du matériel	39
Figure 34 : visualisation de la vitesse de sortie du moteur à l'issu de la phase Hardware-in-the-loop	39
Figure 35 : la phase de Validation Recette	40
Figure 36 : exemple de script écrit en langage MATLAB	42
Figure 37 : le résultat obtenu après exécution du script	42
Figure 38 : exemple de modélisation Simulink	43
Figure 39 : la visualisation dans un scope des résultats obtenus	43
Figure 40 : exemple de modélisation Simscape	44
Figure 41 : la visualisation dans un scope des résultats obtenus	44
Figure 42 : Simscape et ses bibliothèques	46
Figure 43 : modélisation du comportement d'une boîte de vitesse automatique avec Stateflow	47
Figure 44 : les outils MATLAB pour la modélisation multi-phérique	48
Figure 45 : la fenêtre de l'environnement MATLAB	49
Figure 46 : la barre de commande de l'environnement MATLAB	50
Figure 47 : Simulink Start Page	51
Figure 48 : Blank Model Simulink	52
Figure 49 : la bibliothèque de Simulink	53
Figure 50 : la fenêtre de l'environnement Simulink	54
Figure 51 : le path de MATLAB	55
Figure 52 : ajout de dossier au « Path » pour la session courante	56
Figure 53 : le diagramme Chaîne d'énergie / Chaîne d'information	57
Figure 54 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB	58
Figure 55 : photo du pilote automatique de bateau	59
Figure 56 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB	60
Figure 57 : copie d'écran du modèle multi-phérique du pilote automatique de bateau réalisé avec MATLAB	61
Figure 58 : visualisation de la consigne de cap	62
Figure 59 : fenêtre de visualisation de Multibody (Mechanics Explorer)	63
Figure 60 : barre de visualisation de Multibody	63
Figure 61 : fenêtre de visualisation de Multibody (Mechanics Explorer)	64

Figure 62 : barre temporelle de Multibody	64
Figure 63 : visualisation de la consigne de cap et du cap effectif suivi par le bateau	65
Figure 64 : visualisation des grandeurs relatives au moteur	65
Figure 65 : visualisation de l'angle de rotation de la barre	66
Figure 66 : visualisation des pressions dans les chambres avant et arrière du vérin	66
Figure 67 : visualisation des débits dans les chambres avant et arrière du vérin	67
Figure 68 : modèle MATLAB – Simulink de la chaîne d'information du pilote automatique	68
Figure 69 : modélisation de la commande de cap avec des diagrammes d'états	68
Figure 70 : modèle Simscape de la partie électrique de la chaîne d'énergie du pilote automatique	69
Figure 71 : modèle réalisé avec Simscape Fluids de la partie hydraulique de la chaîne d'énergie du pilote automatique	70
Figure 72 : Modélisation à l'aide de Simscape Multibody de la structure du pilote automatique	71
Figure 73 : Modélisation Simulink du comportement dynamique du bateau	72
Figure 74 : modèle du pilote hydraulique avec pilotage interactif	73
Figure 75 : utilisation du pilotage interactif d'un modèle	74
Figure 76 : présentation du robot Maxpid	75
Figure 77 : modélisation multi-phérique du robot MAXPID	76
Figure 78 : position du bras du robot MAXPID	77
Figure 79 : visualisation de la tension de commande du moteur	77
Figure 80 : visualisation du courant dans l'induit du moteur	78
Figure 81 : l'axe linéaire Control'X	79
Figure 82 : modélisation multi-phérique du système Control'X	80
Figure 83 : visualisation de la vitesse et de la position de l'axe linéaire	81
Figure 84 : visualisation de la tension de commande du moteur et du courant dans l'induit	81
Figure 85 : évaluation des écarts entre les performances du modèle et celles de l'axe réel	82
Figure 86 : évaluation des écarts réel/modèle pour la vitesse et la position de l'axe	83
Figure 87 : évaluation des écarts réel/modèle pour le courant d'induit et la tension de commande	83
Figure 88 : circuit R-L	85
Figure 89 : les composants nécessaires à la modélisation du circuit R-L avec Simscape	86
Figure 90 : composants nécessaires à la modélisation du circuit RL	87
Figure 91 : commandes utiles	87
Figure 92 : Modèle Simscape du circuit R-L	88
Figure 93 : commandes utiles	88
Figure 94 : les différents types de ports de Simscape	88
Figure 95 : identification des connexions d'un modèle Simscape	89
Figure 96 : évolution de l'intensité du courant dans le circuit RL	93
Figure 97 : mise en forme des scopes	94
Figure 98 : modification de la mise en forme d'un scope	94
Figure 99 : modification de l'aspect des courbes d'un scope	95
Figure 100 : évolution de la tension aux bornes de la bobine pour le circuit RL	95
Figure 101 : évolution du courant avec une autre valeur de l'inductance	96
Figure 102 : définition du nom des signaux à analyser	97
Figure 103 : sélection des signaux à suivre avec le Data Inspector	97
Figure 104 : visualisation des signaux à suivre avec le Data Inspector	98
Figure 105 : fenêtre du Data Inspector	98
Figure 106 : visualisation d'une courbe à l'aide du Data Inspector	99
Figure 107 : visualisation de plusieurs courbes à l'aide du Data Inspector	99
Figure 108 : visualisation des résultats de simulations différentes à l'aide du Data Inspector	100
Figure 109 : affichage de plusieurs courbes de résultats à l'aide du Data Inspector	100
Figure 110 : modification de la couleur des courbes à l'aide du Data Inspector	101
Figure 111 : visualisation de la modification de la couleur des courbes à l'aide du Data Inspector	101
Figure 112 : sélection des simulations à comparer à l'aide du Data Inspector	101
Figure 113 : visualisation de la comparaison entre deux simulations à l'aide du Data Inspector	102
Figure 114 : les composants nécessaires à la modélisation du circuit R-L avec Simulink	103
Figure 115 : modèle du circuit R-L réalisé avec Simulink	104
Figure 116 : Commandes utiles	104
Figure 117 : évolution de l'intensité du courant dans le circuit RL	106
Figure 118 : avantages et inconvénients des approches causales et acausales	107
Figure 119 : les ports PCP des composants Simscape	108
Figure 120 : les références des principaux domaines physiques de Simscape	109
Figure 121 : les variables Across et through de Simscape	110
Figure 122 : source de force orientée positivement	112
Figure 123 : source de force orientée négativement	112
Figure 124 : orientation d'une source de force	113
Figure 125 : visualisation de l'influence de l'orientation de la source de force	113
Figure 126 : exemple de mesure de grandeurs électriques	114
Figure 127 : implantation des capteurs	115

Figure 128 : visualisation de la tension aux bornes de la bobine pour les deux orientations des capteurs	116
Figure 129 : visualisation de l'intensité du courant dans le circuit en fonction des deux orientations des capteurs	116
Figure 130 : fenêtre de paramétrage du solveur	118
Figure 131 : photo de l'axe linéaire	120
Figure 132 : les composants nécessaires à la modélisation de l'axe linéaire avec Simscape	121
Figure 133 : visualisation des domaines physiques intervenant dans la modélisation de l'axe linéaire avec Simscape	122
Figure 134 : modèle Simscape de l'axe linéaire	123
Figure 135 : commandes utiles	123
Figure 136 : spécification des variables Simscape à prendre en compte dans le logger	133
Figure 137 : fenêtre du Simscape Result Explorer	134
Figure 138 : visualisation des résultats de la simulation	135
Figure 139 : création d'un sous-système	137
Figure 140 : commandes utiles	137
Figure 141 : création d'un sous-système	138
Figure 142 : visualisation du contenu du sous-système	138
Figure 143 : renommer les ports d'un sous-système	139
Figure 144 : naviguer dans les sous-systèmes d'un modèle	139
Figure 145 : modèle Simscape de l'axe linéaire avec sous-système « moteur »	140
Figure 146 : modèle Simscape de l'axe linéaire sans le capteur de vitesse de rotation	140
Figure 147 : modèle Simscape de l'axe linéaire avec le nom des signaux	141
Figure 148 : modèle Simscape de l'axe linéaire, création du sous-système « Axe »	141
Figure 149 : modèle Simscape de l'axe linéaire avec sous-système « moteur » et « Axe »	141
Figure 150 : suppression du port d'un sous-système	142
Figure 151 : modèle Simscape terminé de l'axe linéaire	142
Figure 152 : mettre une image sur un sous-système	142
Figure 153 : configuration de la fenêtre « Add mask icon image »	143
Figure 154 : affichage d'une image sur un sous-système	143
Figure 155 : réponse en position de l'axe linéaire non asservi	144
Figure 156 : modèle de l'asservissement en position linéaire de l'axe	144
Figure 157 : composants à ajouter au modèle	145
Figure 158 : paramétrage du bloc sommateur	146
Figure 159 : réponse en position de l'axe linéaire asservi	147
Figure 160 : modèle de l'asservissement en position de l'axe linéaire	147
Figure 161 : paramétrage du bloc PID	148
Figure 162 : réponse corrigée de l'asservissement en position de l'axe linéaire	149
Figure 163 : les composants nécessaires à la modélisation de la commande d'un vérin hydraulique	150
Figure 164 : visualisation des domaines physiques intervenant dans la modélisation de la commande du vérin hydraulique	151
Figure 165 : modèle Simscape de la commande du vérin hydraulique	152
Figure 166 : évolution de la vitesse et de la position de la tige du vérin	160
Figure 167 : modèle Simscape de la commande du vérin hydraulique avec routage des signaux	161
Figure 168 : visualisation des résultats de la simulation	163
Figure 169 : modèle Simscape de la commande d'un vérin hydraulique avec source de débit	164
Figure 170 : paramétrage d'un bloc limiteur de pression	165
Figure 171 : variation de l'ouverture de soupape en fonction de pression	166
Figure 172 : évolution de la vitesse et de la position de la tige du vérin	166
Figure 173 : évolution de la vitesse et de la position de la tige du vérin	167
Figure 174 : principe de commande PWM	168
Figure 175 : Illustration du fonctionnement du bloc « Controlled PWM Voltage »	169
Figure 176 : tension de commande du bloc Controlled PWM Voltage	169
Figure 177 : signal PWM en fonction de la tension de commande	170
Figure 178 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage	171
Figure 179 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage	172
Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage	172
Figure 181 : commande PWM d'un moteur à courant continu	173
Figure 182 : vitesse de rotation du moteur	173
Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM	174
Figure 184 : commande PWM d'un moteur à courant continu avec pont en H	175
Figure 185 : vitesse de rotation du moteur commandé par le pont en H	176
Figure 186 : tension moyenne d'alimentation du moteur	176
Figure 187 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage	177
Figure 188 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage	178
Figure 189 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage	178
Figure 190 : paramétrage du bloc H-Bridge	181
Figure 191 : vitesse de rotation du moteur commandé par le pont en H	182
Figure 192 : tension moyenne d'alimentation du moteur	182
Figure 193 : les éléments de la bibliothèque de Simulink « Dashboard »	183

Figure 194 : pilotage interactif d'un modèle	184
Figure 195 : visualisation en temps réel de la position de l'axe linéaire	184
Figure 196 : modèle « axe_lineaire_dashboard_start.slx	185
Figure 197 : sélection de la fonction Simulation Pacing	185
Figure 198 : paramétrage de la fenêtre du Simulation Pacing.....	185
Figure 199 : insertion des blocs de la bibliothèque Dashboard.....	186
Figure 200 : fenêtre de paramétrage du bloc knob	187
Figure 201 : allure du bouton knob après paramétrage et connexion avec le modèle	187
Figure 202 : fenêtre de paramétrage du bloc Dashboard Scope	188
Figure 203 : simulation d'un modèle interactif avec les composants de la bibliothèque dashboard.....	189
Figure 204 : transfert de chaleur par conduction	190
Figure 205 : bloc Conductive Heat Transfer de Simscape.....	191
Figure 206 : Paramétrage du bloc Conductive Heat Transfer	191
Figure 207 : transfert de chaleur par convection.....	192
Figure 208 : bloc Convective Heat Transfer de Simscape	192
Figure 209 : Paramétrage du bloc Convective Heat Transfer	193
Figure 210 : transfert de chaleur par rayonnement.....	193
Figure 211 : bloc Radiative Heat Transfer de Simscape	194
Figure 212 : Paramétrage du bloc Radiative Heat Transfer.....	194
Figure 213 : bloc Thermal Mass de Simscape	195
Figure 214 : Paramétrage du bloc Thermal Mass	195
Figure 215 : résistance thermique R_{th}	196
Figure 216 : bloc Thermal Resistance de Simscape	197
Figure 217 : Paramétrage du bloc Thermal Resistance	198
Figure 218 : les éléments de la bibliothèque Simscape Thermal	199
Figure 219 : les capteurs de la bibliothèque Simscape Thermal	199
Figure 220 : les sources de la bibliothèque Simscape Thermal	199
Figure 221 : chauffage d'une barre par conduction	200
Figure 222 : ouverture d'un «Blank Model » Simulink.....	201
Figure 223 : bibliothèque Simscape Thermal	201
Figure 224 : les composants nécessaires à la modélisation du chauffage d'une barre par conduction	202
Figure 225 : positionnement des composants dans la fenêtre de travail.....	202
Figure 226 : les commandes de rotation et d'inversion des composants	203
Figure 227 : modèle Simscape du chauffage d'une barre par conduction	203
Figure 228 : les ports transmetteurs de puissance (PCP) du domaine thermique	204
Figure 229 : paramétrage du bloc Temperature Source.....	205
Figure 230 : paramétrage du bloc Conductive Heat Transfer	205
Figure 231 : paramétrage du bloc Thermal Mass	206
Figure 232 : les ports du capteur de température du domaine thermique	207
Figure 233 : implantation du capteur de température	208
Figure 234 : paramétrage du bloc Ideal Temperature Source	208
Figure 235 : les composants associés au capteur de température	209
Figure 236 : implantation du capteur de température	209
Figure 237 : paramétrage du bloc PS – Simulink Converter	210
Figure 238 : les ports du capteur de flux thermique	210
Figure 239 : mesure d'une grandeur physique de type flux thermique	211
Figure 240 : paramétrage du bloc Ideal Heat Flow Sensor	211
Figure 241 : les composants associés au capteur de flux thermique	212
Figure 242 : implantation des capteurs de flux thermique dans le modèle	212
Figure 243 : paramétrage du bloc PS – Simulink Converter	213
Figure 244 : spécification de la température initiale de la barre	213
Figure 245 : configuration des paramètres du solveur	214
Figure 246 : température au milieu de la barre en fonction du temps	214
Figure 247 : flux thermiques mesurés sur le modèle	215
Figure 248 : circulation des flux thermiques dans les différentes parties du modèle	216
Figure 249 : vérification de la conservation du flux thermique dans le modèle	216
Figure 250 : visualisation de la conservation du flux thermique dans le scope	217
Figure 251 : chauffage d'une barre par conduction non isolée	217
Figure 252 : modélisation de l'échange de chaleur par convection	218
Figure 253 : paramétrage du bloc Temperature Source.....	218
Figure 254 : paramétrage du bloc Convective Heat Transfer	219
Figure 255 : température au milieu de la barre en fonction du temps avec pertes par convection	220
Figure 256 : flux thermiques mesurés sur le modèle avec pertes par convection	220
Figure 257 : ajout d'un capteur pour mesure le flux thermique de convection	221
Figure 258 : visualisation de la conservation du flux thermique	221
Figure 259 : les échanges thermiques à travers une paroi	222

Figure 260 : coefficient de transfert thermique par convection.....	222
Figure 261 : coefficient de transfert thermique par convection.....	223
Figure 262 : conductivité thermique des matériaux usuels.....	223
Figure 263 : modélisation d'une couche de la paroi.....	223
Figure 264 : modélisation Simscape d'une couche de la paroi.....	224
Figure 265 : paramétrage du bloc Convective Heat Transfer	225
Figure 266 : paramétrage du bloc Conductive Heat Transfer	226
Figure 267 : paramétrage du bloc Thermal Mass	226
Figure 268 : modélisation Simscape d'une couche de la paroi.....	227
Figure 269 : Paramétrage du bloc Thermal Resistance pour la convection.....	228
Figure 270 : Paramétrage du bloc Thermal Resistance pour la conduction	229
Figure 271 : modélisation d'une couche de paroi en utilisant.....	229
Figure 272 : modélisation d'une paroi d'un bâtiment.....	230
Figure 273 : caractéristiques physiques des différentes couches de matériau.....	230
Figure 274 : coefficients de transfert thermique par convection.....	230
Figure 275 : paramètres nécessaires à la construction du modèle de la paroi multicouche	231
Figure 276 : modélisation d'une paroi multicouche, démarrage de la modélisation	231
Figure 277 : paramétrage du bloc Température intérieure.....	232
Figure 278 : paramétrage du bloc Convection Air intérieur/paroi	232
Figure 279 : paramétrage du bloc Capteur de flux thermique.....	232
Figure 280 : paramétrage du bloc PS – Simulink Converter	233
Figure 281 : paramétrage du bloc Conduction laine de verre.....	233
Figure 282 : paramétrage du bloc Masse thermique de la laine de verre	234
Figure 283 : paramétrage du bloc Conduction béton plein.....	234
Figure 284 : paramétrage du bloc Masse thermique du béton plein.....	235
Figure 285 : paramétrage du bloc Conduction enduit.....	236
Figure 286 : paramétrage du bloc Masse thermique de l'enduit	236
Figure 287 : paramétrage du bloc Convection Air extérieur/paroi	237
Figure 288 : paramétrage du bloc Température extérieure	237
Figure 289 : réglage du temps de simulation.....	237
Figure 290 : visualisation du flux thermique en intérieur de paroi en régime permanent.....	238
Figure 291 : modification d'un paramètre dans le script	238
Figure 292 : visualisation de la nouvelle valeur du flux thermique	238
Figure 293 : modélisation équivalente avec des résistance thermiques	239
Figure 294 : paramétrage du bloc Résistance thermique de convection Air intérieur/paroi	239
Figure 295 : paramétrage du bloc Résistance thermique de conduction de la laine de verre	240
Figure 296 : paramétrage du bloc Résistance thermique de conduction du béton plein	240
Figure 297 : paramétrage du bloc Résistance thermique de conduction de l'enduit	240
Figure 298 : paramétrage du bloc Résistance thermique de convection Air extérieur/paroi	241
Figure 299 : double vitrage simple.....	241
Figure 300 : double vitrage avec couche d'argon.....	241
Figure 301 : modèle retenu pour la pièce intérieure et pour la paroi vitrée	242
Figure 302 : caractéristiques des matériaux	242
Figure 303 : paramètres nécessaires à la construction du modèle d'un double vitrage	243
Figure 304 : modélisation d'une paroi multicouche, démarrage de la modélisation	243
Figure 305 : sous-système « Double vitrage + argon »	244
Figure 306 : sous-système « Double vitrage »	244
Figure 307 : paramétrage du bloc Masse thermique de l'air de la pièce intérieure	245
Figure 308 : paramétrage du bloc Convection Air intérieur/paroi	245
Figure 309 : paramétrage du bloc Convection paroi vitrée/air extérieur	246
Figure 310 : paramétrage du bloc Température extérieure	246
Figure 311 : paramétrage du bloc PS – Simulink Converter	247
Figure 312 : paramétrage du bloc Conduction de la vitre intérieure	247
Figure 313 : paramétrage du bloc Masse thermique de la vitre intérieure	248
Figure 314 : paramétrage du bloc Conduction de la couche d'argon	248
Figure 315 : paramétrage du bloc Masse thermique de l'argon	249
Figure 316 : paramétrage du bloc Conduction de la vitre extérieure	249
Figure 317 : paramétrage du bloc Masse thermique de la vitre extérieure	250
Figure 318 : paramétrage du bloc Conduction de la vitre intérieure	251
Figure 319 : paramétrage du bloc Masse thermique de la vitre intérieure	251
Figure 320 : réglage du temps de simulation	252
Figure 321 : évolution de la température dans la pièce en fonction du temps pour les deux types de vitrage	252
Figure 322 : modèles Simscape Thermal	253
Figure 323 : modèle House Heating System , Mathworks ©	253
Figure 324 : modèle thermique de la maison réalisé avec Simscape	254
Figure 325 : évolution des températures intérieures et extérieures et du coût du chauffage	255

Figure 326 : température des différentes parois de la maison.....	255
Figure 327 : flux de chaleur au travers des différentes parois de la maison	256
Figure 328 : évolution de la température sans système de chauffage	256
Figure 329 : présentation du hacheur série.....	257
Figure 330 : principe de fonctionnement du hacheur série.....	258
Figure 331 : schéma de principe du hacheur série.....	258
Figure 332 : schéma de commande d'un moteur à courant par un hacheur série.....	259
Figure 333 : circulation du courant dans le circuit en phase active et en phase de roue libre	259
Figure 334 : visualisation de l'influence du rapport cyclique sur l'ondulation de courant	260
Figure 335 : visualisation de l'influence de la fréquence de hachage sur l'ondulation de courant	260
Figure 336 : visualisation de l'influence de la valeur de l'inductance sur l'ondulation de courant	261
Figure 337 : analogie entre la forme du modèle Simscape et le schéma électrique de principe.....	262
Figure 338 : modèle Simscape du hacheur série	263
Figure 339 : les composants nécessaires à la modélisation du hacheur série	263
Figure 340 : paramétrage du bloc Diode.....	264
Figure 341 : paramétrage du bloc MOSFET	265
Figure 342 : paramétrage du bloc Pulse Generator	266
Figure 343 : évolution de la vitesse de rotation du moteur commandé par un hacheur série	267
Figure 344 : création d'un sous-système pour didactiser le modèle	268
Figure 345 : mise en place des capteurs de courant dans le circuit.....	268
Figure 346 : ajout d'un capteur sur le modèle	269
Figure 347 : modèle instrumenté et didactisé	269
Figure 348 : sous-système « capteur de courant »	270
Figure 349 : évolution du courant dans les trois branches du circuit.....	271
Figure 350 : le modèle non didactisé	273
Figure 351 : le modèle didactisé.....	273
Figure 352 : amélioration de la didactisation du modèle.....	274
Figure 353 : le modèle didactisé et optimisé.....	275
Figure 354 : réglage de la période de commutation du transistor	276
Figure 355 : visualisation de la valeur moyenne et instantanée du courant moteur	277
Figure 356 : ajout d'une inductance de lissage en série avec le moteur	278
Figure 357 : exploitation et visualisation de la circulation du courant dans le hacheur série.....	279
Figure 358 : réglages des paramètres de la simulation.....	280
Figure 359 : exploitation et visualisation de l'influence du rapport cyclique sur le courant moteur.....	281
Figure 360 : réglage des paramètres de la simulation.....	282
Figure 361 : exploitation et visualisation de l'influence de la fréquence de hachage sur le courant moteur	283
Figure 362 : paramétrage du bloc Pulse Generator	284
Figure 363 : réglage des paramètres de la simulation	285
Figure 364 : visualisation et exploitation de l'influence de l'inductance de la charge sur le courant moteur	286
Figure 365 : tableau de données	289
Figure 366 : tracé de deux vecteurs.....	290
Figure 367 : tracé de deux courbes dans la même fenêtre graphique	291
Figure 368 : affichage de la grille sur un graphique.....	291
Figure 369 : ouverture d'une nouvelle fenêtre graphique.....	292
Figure 370 : codes de mise en forme des courbes	292
Figure 371 : mise en forme d'une courbe : modification de la couleur et du style de ligne	293
Figure 372 : mise en forme d'une courbe : ajout de marqueurs	293
Figure 373 : mise en forme d'une courbe : choix de l'épaisseur de trait	294
Figure 374 : mise en forme d'une courbe : légende et annotation des axes	295
Figure 375 : mise en forme d'une courbe : modification des échelles des axes	295
Figure 376 : indexation des lignes et des colonnes d'une matrice.....	296
Figure 377 : indexation des éléments d'une matrice avec un indice unique	297
Figure 378 : extraction d'une matrice.....	298
Figure 379 : extraction d'une ligne entière ou d'une colonne entière	299
Figure 380 : tracé de la fonction $f(t)$	302
Figure 381 : fenêtre Editor d'édition des scripts	302
Figure 382 : exemple de script	303
Figure 383 : fenêtre d'ajout automatique de dossier au « path » de MATLAB.....	303
Figure 384 : Tracé de plusieurs Sinus sur le même graphique.....	304
Figure 385 : résultats du script de tracé de plusieurs sinus	305
Figure 386 : les opérateurs logiques et de comparaison de MATLAB	305
Figure 387 : répartition d'une série de points	307
Figure 388 : script d'interpolation d'une série de données	308
Figure 389 : graphique de l'interpolation d'une série de données	308
Figure 390 : script de résolution d'une équation du second degré	309
Figure 391 : résolution d'une équation du second degré avec racines complexes.....	309

Figure 392 : développer et factoriser une expression	311
Figure 393 : script permettant de dériver une fonction	312
Figure 394 : script permettant d'intégrer une fonction et de calculer une intégrale définie	312
Figure 395 : script permettant de calculer la transformée de Laplace de fonctions.....	313
Figure 396 : script permettant d'obtenir la transformée inverse de Laplace d'une fonction.....	314
Figure 397 : script permettant de décomposer en éléments simples une fraction rationnelle.....	314
Figure 398 : script permettant la résolution d'une équation différentielle du second ordre	316
Figure 399 : représentation de la solution 1 de l'équation différentielle d'ordre 2	317
Figure 400 : représentation de la solution 2 de l'équation différentielle d'ordre 2	318
Figure 401 : fonctions de transfert en série.....	319
Figure 402 : fonction de transfert en parallèle.....	320
Figure 403 : fonction de transfert en boucle fermée	320
Figure 404 : fonction de transfert d'un système quelconque.....	321
Figure 405 : réponse indicielle d'un système.....	322
Figure 406 : réponse impulsionale d'un système	322
Figure 407 : diagramme de Bode d'un système	323
Figure 408 : diagramme de Black-Nichols d'un système.....	323
Figure 409 : diagramme de Nyquist d'un système	324
Figure 410 : tracé de deux diagrammes de Bode sur le même graphique.....	324
Figure 411 : diagramme de Bode avec indication des marges de gain et de phase.....	325
Figure 412 : ouverture du modèle Simulink du four	329
Figure 413 : modélisation de l'asservissement en température d'un four	329
Figure 414 : script contenant les paramètres de la simulation	330
Figure 415 : visualisation des variables créées dans le Workspace.....	330
Figure 416 : réponse en température du four	331
Figure 417 : nommer un signal Simulink	332
Figure 418 : placement des points de linéarisation pour tracer un diagramme de Bode en boucle ouverte	332
Figure 419 : placement d'un point de linéarisation de type « Open loop input ».....	333
Figure 420 : représentation des points de linéarisation.....	333
Figure 421 : insertion d'un bloc Bode Plot	334
Figure 422 : fenêtre de paramétrage du bloc Bode Plot	334
Figure 423 : fenêtre de tracé du diagramme de Bode.....	335
Figure 424 : diagramme de Bode de la boucle ouverte	335
Figure 425: représentation des points de linéarisation.....	336
Figure 426 : insertion d'un second bloc Bode plot.....	336
Figure 427: fenêtre de paramétrage du bloc Bode Plot	337
Figure 428 : paramétrage des points de linéarisation pour la boucle fermée	338
Figure 429 : diagramme de Bode de la boucle fermée	339
Figure 430 : asservissement en température d'un four avec saturation	340
Figure 431 : réponse en température de four avec saturation	341
Figure 432 : mise en place d'un scope sur un signal.....	342
Figure 433 : visualisation de la tension en sortie de saturateur	342
Figure 434 : fenêtre de paramétrage de l'exportation des données vers le Workspace	344
Figure 435 : script pour tracer une série de courbes	345
Figure 436 : influence de la correction proportionnelle sur la température du four	346
Figure 437 : modèle du four pour la linéarisation.....	346
Figure 438 : script permettant de linéariser un modèle Simulink	347
Figure 439 : diagramme de Bode obtenu en linéarisant le modèle Simulink	347
Figure 440 : script permettant de tracer une série de diagrammes de Bode en faisant varier le gain proportionnel du correcteur.....	347
Figure 441 : diagrammes de Bode obtenus en faisant varier le gain du correcteur proportionnel	348
Figure 442 : boucle de cap sans le diagramme d'état.....	350
Figure 443 : bibliothèque Statflow	350
Figure 444 : positionnement d'un chart dans un modèle Simulink	351
Figure 445 : commande de création d'un état et d'une « transition par défaut ».....	351
Figure 446 : ouverture de la fenêtre Symbols.....	352
Figure 447 : environnement de travail de Stateflow	352
Figure 448 : création des états du système	353
Figure 449 : création d'une transition par défaut.....	353
Figure 450 : création des transitions entre les états	353
Figure 451 : affectation des actions dans les états	354
Figure 452 : nature des informations d'une étiquette de transition	355
Figure 453 : écriture des étiquettes de transitions dans un diagramme d'états	355
Figure 454 : chart non connecté avec le modèle	356
Figure 455 : fenêtre Symbols avec paramétrage incomplet des entrées/sorties.....	356
Figure 456 : affectation des variables du diagramme à l'aide du Symbol Wizard	357

Figure 457 : la fenêtre Symbols	357
Figure 458 : visualisation des variables dans le Model Explorer	358
Figure 459 : chart avant connexion avec le système	358
Figure 460 : inversions de la position des ports.....	359
Figure 461 : modification des numéros de port d'un chart.....	359
Figure 462 : connexion du chart avec le schéma bloc Simulink.....	359
Figure 463 : résultat de la simulation	360
Figure 464 : représentation de super-état et de sous-état	361
Figure 465 : représentation de sous-états parallèles.....	362
Figure 466 : création de super-états	363
Figure 467 : créer des sous-états parallèles	364
Figure 468 : représentation des états parallèles dans Stateflow.....	364
Figure 469 : utilisation de variables internes pour évaluer l'activation d'un état	365
Figure 470 : paramétrage de la variable voyant_rouge.....	365
Figure 471 : chart complet de commande de cap avec super-états et états parallèles.....	366
Figure 472 : connexion de la variable de sortie du chart à un scope.....	366
Figure 473 : visualisation de l'état d'activation du voyant	367
Figure 474 : ajout d'un voyant de la bibliothèque Dashboard dans le modèle	367
Figure 475 : rappel des principales règles de syntaxe de Stateflow	369
Figure 476 : modèle Multibody de la partie mécanique du pilote hydraulique	370
Figure 477 : modèle Multibody du pilote avec masques sur les solides	371
Figure 478 : fenêtre Mechanics Explorers de la partie mécanique du pilote hydraulique	371
Figure 479 : options de la barre de visualisation de Multibody.....	372
Figure 480 : paramétrage de la pesanteur dans Multibody	372
Figure 481 : modification de l'action de la pesanteur.....	373
Figure 482 : modèle du pilote hydraulique avec modèle Multibody à intégrer	374
Figure 483 : visualisation du système Multibody sans connexion avec le reste du modèle	374
Figure 484 : placement des ports de connexion dans le modèle	375
Figure 485 : connexions des ports du sous-système avec le modèle	376
Figure 486 : interface de translation entre Simscape et Multibody	376
Figure 487 : interfaçage de rotation entre Simscape et Multibody	377
Figure 488 : les interfaces entre Simscape et Multibody.....	377
Figure 489 : ajout du bloc Simscape Multibody Interface.....	378
Figure 490 : paramétrage des ports d'une liaison	379
Figure 491 : connexion entre Simscape et Multibody.....	379
Figure 492 : ajout d'un port pour imposer un couple à la liaison	380
Figure 493 : conversion du signal physique en signal Simulink.....	380
Figure 494 : ajout de port sur une liaison dans Multibody.....	381
Figure 495 : visualisation de l'offset de l'angle de barre	381
Figure 496 : compensation de l'offset de l'angle de la barre	382
Figure 497 : ajout d'un effort sur la liaison, utilisation d'une Lookup Table	383
Figure 498 : paramétrage d'une Lookup Table	383
Figure 499 : visualisation d'une Lookup Table	384
Figure 500 : visualisation de la courbe représentative de la loi entrée sortie d'une Lookup Table.....	384
Figure 501 : résultat de la simulation du cap suivi par le bateau.....	385
Figure 502 : choix de la version de Multibody Link	386
Figure 503 : fenêtre d'accueil de Solidworks	387
Figure 504 : ouverture de la fenêtre des compléments de Solidworks	388
Figure 505 : sélection des compléments de Solidworks	388
Figure 506 : ouverture du modèle Solidworks de la partie mécanique du pilote	389
Figure 507 : modèle Multibody obtenu.....	390
Figure 508 : fenêtre Mechanics Explorer	390
Figure 509 : la modélisation black box.....	391
Figure 510 : les conditions de l'essai.....	392
Figure 511 : visualisation des données utilisées pour l'identification	392
Figure 512 : visualisation dans le Workspace des données utilisées pour l'identification	393
Figure 513 : menu déroulant des applications de MATLAB.....	393
Figure 514 : description de la fenêtre de la toolbox identification.....	394
Figure 515 : importation de données temporelles.....	394
Figure 516 : sélection des données temporelles pour l'identification	395
Figure 517 : visualisation des données importées dans la fenêtre de la « System Identification » toolbox.....	395
Figure 518 : visualisation des données importées dans la « System Identification » toolbox	396
Figure 519 : choix de la forme du modèle dans la « System Identification » toolbox.....	396
Figure 520 : choix de la forme de la fonction de transfert dans la « System Identification » toolbox	397
Figure 521 : la fenêtre Plant Identification Progress de la « System Identification » toolbox	397
Figure 522 : visualisation de résultats de l'identification	398

Figure 523 : fonction de transfert obtenue par identification	398
Figure 524 : superposition des données issues de l'expérimentation et du modèle issu de l'identification	399
Figure 525 : diagramme de Bode de la fonction de transfert tf1	399
Figure 526 : modélisation du moteur à courant continu	402
Figure 527 : asservissement en vitesse d'un moteur à courant continu	403
Figure 528 : modélisation de l'asservissement en vitesse du moteur	403
Figure 529 : réponse temporelle du système non corrigé à un échelon unitaire	404
Figure 530 : ajout d'un bloc PID Controller dans l'asservissement	404
Figure 531 : fenêtre de paramétrage du PID Controller	405
Figure 532 : fenêtre de réglage des performances du système	406
Figure 533 : les critères de performances du système et les valeurs de gain du correcteur	407
Figure 534 : ajout du diagramme de Bode de la fonction de transfert en boucle ouverte	407
Figure 535 : visualisation du diagramme de Bode de la fonction de transfert en boucle ouverte pour le réglage du PID	408
Figure 536 : réglage des options des critères de performances	408
Figure 537 : visualisation des critères de performances	409
Figure 538 : réponse corrigée	410
Figure 539 : actualisation automatique des paramètres du PID	411
Figure 540 : réponse du système après réglage du PID	411
Figure 541 : modèle du moteur à courant continu asservi en vitesse avec perturbation	412
Figure 542 : réponse temporelle du système non corrigé	413
Figure 543 : lancement de l'application « Control System Designer »	414
Figure 544 : Control System Designer	414
Figure 545 : choix du bloc à régler	415
Figure 546 : fenêtre Edit Architecture	415
Figure 547 : choix des points de linéarisation pour la FTBO	416
Figure 548 : diagramme de Bode de la FTBO	416
Figure 549 : diagramme de Black de la FTBO	417
Figure 550 : choix de la configuration des fenêtres graphiques	417
Figure 551 : modification de la configuration des fenêtres graphiques	418
Figure 552 : choix des points de linéarisation	418
Figure 553 : choix du point d'entrée de la fonction de transfert	419
Figure 554 : fenêtre New Step to Plot	419
Figure 555 : choix du point de sortie de la fonction de transfert	420
Figure 556 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la consigne	420
Figure 557 : point d'entrée et de sortie pour la réponse temporelle vis-à-vis de la perturbation	421
Figure 558 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la perturbation	422
Figure 559 : visualisation des fenêtres utiles pour effectuer le réglage	422
Figure 560 : les fonctions de la fenêtre de réglage du PID	423
Figure 561 : définition des critères de rapidité	424
Figure 562 : visualisation du temps de réponse à 5%	424
Figure 563 : définition des critères de performances	425
Figure 564 : visualisation des critères de performances dans la fenêtre graphique	425
Figure 565 : réglage satisfaisant le cahier des charges du PID	426
Figure 566 : résultat du réglage manuel du PID sur la réponse temporelle	426
Figure 567 : réponse indicielle du système vis-à-vis d'un échelon de perturbation	427
Figure 568 : influence des réglages effectués sur les diagrammes fréquentiels de la FTBO	428
Figure 569 : réponse temporelle après réglage du PID	429
Figure 570 : modèle de l'asservissement du moteur à continu avec correction par fonction de transfert à régler	430
Figure 571 : réponse du système non corrigé	431
Figure 572 : lancement de l'application « Control System Designer »	432
Figure 573 : Control System Designer	432
Figure 574 : choix du bloc à régler	433
Figure 575 : fenêtre Edit Architecture	433
Figure 576 : choix des points de linéarisation pour la FTBO	434
Figure 577 : diagramme de Bode de la FTBO	434
Figure 578 : diagramme de Black de la FTBO	435
Figure 579 : choix de la configuration des fenêtres graphiques	435
Figure 580 : modification de la configuration des fenêtres graphiques	436
Figure 581 : choix des points de linéarisation	437
Figure 582 : choix du point d'entrée de la fonction de transfert	437
Figure 583 : fenêtre New Step to Plot	438
Figure 584 : choix du point de sortie de la fonction de transfert	438
Figure 585 : définition des points de linéarisation de FTBF_consigne	439
Figure 586 : ajout de la réponse indicielle de la FTBF	439
Figure 587 : tracé du diagramme de Bode de la FTBF	440
Figure 588 : configuration des diagrammes pour effectuer le réglage	440

Figure 589 : la barre de commande du Bode Editor.....	441
Figure 590 : ajouts d'éléments au correcteur	442
Figure 591 : variation du gain de la fonction de transfert en boucle ouverte	443
Figure 592 : Edition de la fonction de transfert su correcteur	444
Figure 593 : ajout d'un intégrateur dans le correcteur	445
Figure 594 : visualisation de l'instabilité de la réponse indicielle	445
Figure 595 : visualisation de la fonction de transfert du correcteur	446
Figure 596 : visualisation des pôles et des zéros sur les courbes	447
Figure 597 : stabilisation du système par ajout d'un correcteur à avance de phase	448
Figure 598 : visualisation de la fonction de transfert du correcteur	449
Figure 599 : ajout d'un filtre rejecteur au correcteur	450
Figure 600 : visualisation des paramètres de réglage du filtre rejecteur.....	451
Figure 601 : réglage du filtre rejecteur	452
Figure 602 : performances de la boucle fermée après réglage du filtre.....	453
Figure 603 : visualisation de la fonction de transfert du correcteur	454
Figure 604 : influence du filtre sur le comportement de la boucle ouverte.....	455
Figure 605 : réponse indicielle corrigée.....	456
Figure 606 : comparaison des réponses indicielles corrigée et non corrigée	456
Figure 607 : organisation des scripts et des fonctions, méthode 1	458
Figure 608 : organisation des scripts et des fonctions, méthode 2	459
Figure 609 : my_0_function.....	459
Figure 610 : codage de la fonction my_0_function.....	460
Figure 611 : codage de la fonction my_0_function.....	460
Figure 612 : visualisation de la fonction my_0_function dans le répertoire courant.....	460
Figure 613 : my_1_function.....	461
Figure 614 : codage de my_1_function	461
Figure 615 : codage de la fonction my_1_function.....	462
Figure 616 : visualisation de la fonction my_1_function dans le répertoire courant.....	462
Figure 617 : my_2_function.....	463
Figure 618 : codage de my_2_function	463
Figure 619 : codage de la fonction my_2_function	463
Figure 620 : visualisation de la fonction my_2_function dans le répertoire courant.....	464
Figure 621 : script_0.m.....	464
Figure 622 : résultat obtenu suite à l'exécution de script_0.m	465
Figure 623 : ajout d'une légende automatiquement au graphique.....	466
Figure 624 : représentation graphique avec légende automatique	467
Figure 625 : script_0_legend_all_in_one	468
Figure 626 : my_plot_function	469
Figure 627 : my_plot_function	470
Figure 628 : my_3_function.....	470
Figure 629 : my_3_function.....	471
Figure 630 : appel de la fonction my_plot_function	471
Figure 631 : modification de my_3_function	471
Figure 632 : appel de la fonction my_plot_function	472
Figure 633 : my_4_function.....	472
Figure 634 : codage de my_4_function	472
Figure 635 : my_5_function.....	473
Figure 636 : codage de my_5_function	473
Figure 637 : la fonction fzero	473
Figure 638 : calcul de la dérivée numérique au point x0	475
Figure 639 : différence finie progressive	476
Figure 640 : formule de différence finie rétrograde	476
Figure 641 : formule de différence finie centrée	477
Figure 642 : codage de la fonction f1 qui contient l'expression de la fonction à dériver	478
Figure 643 : fonction fderiv_forward (a,b,nb_point,f).....	478
Figure 644 : codage de la fonction fderiv_forward (a,b,nb_point,f)	479
Figure 645 : fonction fderiv_backward (a,b,nb_point,f)	480
Figure 646 : codage de la fonction fderiv_backward (a,b,nb_point,f).....	481
Figure 647 : fonction fderiv_center (a,b,nb_point,f)	481
Figure 648 : codage de la fonction fderiv_center (a,b,nb_point,f)	482
Figure 649 : fonction fderiv_ana_symbolique (a,b,nb_point,f).....	482
Figure 650 : codage de la fonction fderiv_ana_symbolique (a,b,nb_point,f)	483
Figure 651 : comparaison des différentes méthodes de dérivation numérique	485
Figure 652 : comparaison des différentes méthodes de dérivation numérique	485
Figure 653 : zoom sur la courbe	486
Figure 654 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique	486

Figure 655 : script permettant de visualiser l'influence du pas.....	488
Figure 656 : influence du pas sur l'allure des courbes	489
Figure 657 : script permettant de connaître l'ordre de variation des fonctions erreurs	492
Figure 658 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée	492
Figure 659 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde.....	493
Figure 660 : modification de la fonction f1.m	494
Figure 661 : comparaison des différentes méthodes de dérivation numérique	494
Figure 662 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique	495
Figure 663 : influence du pas sur l'allure des courbes	495
Figure 664 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée	496
Figure 665 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde.....	496
Figure 666 : mesure de la position à partir du signal d'un codeur incrémental	497
Figure 667 : zoom sur la courbe	497
Figure 668 : influence du pas d'échantillonnage sur le calcul de la dérivée numérique.....	498
Figure 669 : dérivation numérique du signal brut	498
Figure 670 : influence de l'augmentation du pas de dérivation.....	499
Figure 671 : fonction fderiv_num_pas_variable(t,Y,step_size).....	499
Figure 672 : codage de la fonction fderiv_num_pas_variable(t,Y,step_size)	500
Figure 673 : script permettant de faire varier le pas de dérivation	501
Figure 674 : influence du pas de dérivation numérique	502
Figure 675 : illustration de la recherche des solutions de l'équation $f(x)=0$ par la méthode de dichotomie	503
Figure 676 : fonction fdichotomie(f,a,b,epsilon)	504
Figure 677 : codage de la fonction fdichotomie(f,a,b,epsilon).....	504
Figure 678: fonction f2.m.....	505
Figure 679 : recherche de solution par dichotomie	506
Figure 680 : courbe représentative de la fonction $f(x)$	506
Figure 681 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle [-4 ; 2].....	506
Figure 682 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle [0 ; 2].....	506
Figure 683 : illustration de la méthode de Newton.....	507
Figure 684 : fonction fNewton(f,x0,epsilon)	508
Figure 685 : codage de la fonction fNewton(f,x0,epsilon)	509
Figure 686: fonction f2.m.....	509
Figure 687 : recherche de solution par la méthode de Newton	510
Figure 688 : courbe représentative de la fonction $f(x)$	510
Figure 689 : exécution du script Newton.m pour rechercher la solution autour de $x=3$	510
Figure 690 : exécution du script Newton.m pour rechercher la solution autour de $x=-2$	511
Figure 691 : calcul de l'intégrale d'une fonction sur un intervalle [a ; b]	511
Figure 692 : discréttisation du domaine d'intégration	512
Figure 693 : méthode des rectangles	513
Figure 694 : méthode des trapèzes	513
Figure 695 : codage de la fonction f3.m qui contient l'expression de la fonction à intégrer.....	513
Figure 696 : fonction fintégration_rec (a,b,nb_point,f).....	514
Figure 697 : codage de la fonction fintegration_rec (a,b,nb_point,f)	514
Figure 698 : fonction fintégration_trap (a,b,nb_point,f)	515
Figure 699 : codage de la fonction fintegration_trap (a,b,nb_point,f)	515
Figure 700 : fonction fintegration_ana_symbolique (a,b,nb_point,f)	516
Figure 701 : codage de la fonction fintegration_ana_symbolique (a,b,nb_point,f)	516
Figure 702 : comparaison des différentes méthodes de dérivation numérique	518
Figure 703 : comparaison des différentes méthodes d'intégration numérique.....	519
Figure 704 : zoom sur la courbe	520
Figure 705 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique	520
Figure 706 : script permettant de visualiser l'influence du pas.....	522
Figure 707 : influence du pas sur l'allure des courbes	522
Figure 708 : script permettant de connaître l'ordre de variation des fonctions erreurs	525
Figure 709 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles	525
Figure 710 : variation de l'erreur de d'intégration en fonction de h pour la méthode des trapèzes	526
Figure 711 : modification de la fonction f1.m	526
Figure 712 : comparaison des différentes méthodes d'intégration numérique.....	527
Figure 713 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique	527
Figure 714 : influence du pas sur l'allure des courbes	528
Figure 715 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles	528
Figure 716 : variation de l'erreur d'intégration en fonction de h pour la méthode des trapèzes	529
Figure 717 : relevé de l'accélération en fonction du temps	529
Figure 718 : intégration numérique de l'accélération	531
Figure 719 : loi en trapèze de vitesse.....	532
Figure 720 : exemple de signal bruité.....	533

Figure 721 : filtre numérique.....	533
Figure 722 : mesure de la position à partir du signal d'un codeur incrémental	534
Figure 723 : zoom sur la courbe	534
Figure 724 : fonction filtre_mgl(Y,n)	535
Figure 725 : codage de la fonction filtre_mgl(Y,n).....	536
Figure 726 : script permettant de voir l'influence de n pour un filtre à moyenne glissante.....	537
Figure 727 : filtrage par moyenne glissante d'un signal.....	538
Figure 728 : zoom sur la courbe	538
Figure 729 : filtre passe-bas du premier ordre dans le domaine de Laplace	538
Figure 730 : filtre passe-bas du premier ordre dans le domaine temporel	539
Figure 731 : fonction filtre_pb_ordre_1(t,Y,tau)	539
Figure 732 : codage de la fonction filtre_pb_ordre_1(t,Y,tau)	540
Figure 733 : script permettant de voir l'influence de tau pour un filtre passe-bas du premier ordre.....	541
Figure 734 : filtrage d'un signal par un filtre passe-bas du premier ordre avec différentes valeurs de t.....	541
Figure 735 : processus de traitement du signal.....	542
Figure 736 : dérivation numérique en utilisant Simulink.....	542
Figure 737 : signal brut à dériver	543
Figure 738 : dérivation d'un signal numérique discontinu sans filtrage	543
Figure 739 : filtrage réalisé avec une fonction de transfert	544
Figure 740 : paramétrage du bloc Transfer Fcn	544
Figure 741 : dérivation numérique d'un signal filtré en utilisant Simulink	545
Figure 742 : filtrage réalisé avec le bloc Analog Filter Design	546
Figure 743 : paramétrage du bloc Analog Filter Design	546
Figure 744 : dérivation numérique d'un signal filtré en utilisant Simulink	547
Figure 745 : filtrage réalisé avec le bloc Moving Average	548
Figure 746 : paramétrage du bloc Moving Average	548
Figure 747 : dérivation numérique d'un signal filtré en utilisant Simulink	549
Figure 748 : illustration de la méthode d'Euler de résolution numérique d'une équation différentielle.....	551
Figure 749 : codage de la fonction F1_m qui contient l'expression de la fonction f(t,y)	552
Figure 750 : fonction fEuler (f, a, b, CI, nb_points).....	553
Figure 751 : codage de la fonction fEuler(f, a, b, CI, nb_points)	553
Figure 752 : résolution d'une équation différentielle du premier ordre	554
Figure 753 : solution numérique calculée avec 10 points	554
Figure 754 : solution numérique calculée avec 100 points.....	555
Figure 755 : fonction my_ode (f, a, b, CI, nb_points)	558
Figure 756 : structure de la matrice sol	558
Figure 757 : codage de la fonction my_ode (f,a,b,CI,nb_points)	559
Figure 758 : codage de la fonction F_2.m qui contient l'expression de la fonction f(t,y)	560
Figure 759 : comparaison des différentes méthodes de résolution d'une équation différentielle.....	561
Figure 760 : comparaison des différentes méthodes de résolution d'une équation différentielle (100 points calculés)	562
Figure 761 : comparaison des différentes méthodes de résolution d'une équation différentielle (1000 points calculés).....	563
Figure 762 : zoom sur la courbe	563
Figure 763 : comparaison des différentes méthodes de résolution d'une équation différentielle(10000 points calculés)	564
Figure 764 : zoom sur la courbe	564
Figure 765 : codage de la fonction F_3.m	566
Figure 766 : résolution d'une équation différentielle d'ordre 3 avec my_ode et ode23	567
Figure 767 : solution d'une équation différentielle du troisième ordre	567
Figure 768 : résolution de l'équation différentielle de Van der Pol	568
Figure 769 : solution de l'équation différentielle de Van der Pol, 10000 points calculés	569
Figure 770 : oscillateur mécanique avec prise en compte de la pesanteur	570
Figure 771 : codage de la fonction F_mass_spring_damper.m	571
Figure 772 : Calcul et tracé de la solution de l'équation différentielle en utilisant 3 méthodes de résolution	573
Figure 773 : résolution de l'équation différentielle avec 3 méthodes.....	573
Figure 774 : modèle permettant la résolution de l'équation différentielle avec Simulink	574
Figure 775 : spécification de la condition initiale $x(0)=0.1$ dans l'intégrateur de la position.....	574
Figure 776 : visualisation de la solution $x(t)$ déterminée avec Simulink	575
Figure 777 : modèle multi_physique du système masse-ressort-amortisseur.....	575
Figure 778 : spécification de la condition initiale dans Simscape	576
Figure 779 : visualisation de la solution $x(t)$ calculée avec Simscape	576
Figure 780 : recherche du numéro de ligne du plus grand pivot	581
Figure 781 : fonction num_ligne_pivot.m	581
Figure 782 : codage de la fonction num_ligne_pivot(A,num_col)	582
Figure 783 : fonction echange_lignes(A, ligne_i, ligne_j).....	583
Figure 784 : codage de la fonction echange_ligne(A, ligne_i, ligne_k)	584
Figure 785 : fonction transvection(A, num_ligne_pivot)	585
Figure 786 : codage de la fonction transvection(A, num_ligne_pivot)	585

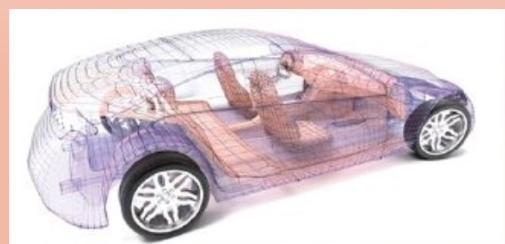
Figure 787 : fonction triangulation(A)	586
Figure 788 : codage de la fonction triangulation(A)	587
Figure 789 : fonction résolution.m	587
Figure 790 : codage de la fonction resolution(M)	589
Figure 791 : fonction Pivot_de_Gauss(A,B)	589
Figure 792 : hiérarchisation des fonctions utilisées dans l'algorithme du pivot de Gauss	590
Figure 793 : codage de la fonction Pivot_de_Gauss(A,B)	590
Figure 794 : MATLAB/Simulink online et MATLAB Drive	593
Figure 795 : Dossier MATLAB Drive dans la version installée de MATLAB/Simulink	594
Figure 796 : visualisation de MATLAB Drive Connector dans la barre de tâche de Windows	594
Figure 797 : interface de MATLAB Drive Connector	595
Figure 798 : fenêtre de MATLAB online	596
Figure 799 : script à saisir online	596
Figure 800 : création d'un Blank Model dans Simulink online	598
Figure 801 : librairie de Simulink online	598
Figure 802 : modèle créer à l'aide de Simulink online	598
Figure 803 : résultat de la simulation	599
Figure 804 : ajout du dossier Maxpid dans le Path de MATLAB online	600
Figure 805 : modèle du robot Maxpid ouvert dans MATLAB/Simulink online	600
Figure 806 : résultats de la simulation	601
Figure 807 : visualisation des résultats dans le scope	601
Figure 808 : partage d'un dossier du Drive de MATLAB	602
Figure 809 : définition des membres et des droits d'accès au dossier	603
Figure 810 : gestion des droits d'accès et des membres	603
Figure 811 : création d'un lien pour partager un dossier de MATLAB Drive	604
Figure 812 : partage d'un dossier en utilisant MATLAB on line	604
Figure 813 : ouverture de l'interface MATLAB Drive Connector	605
Figure 814 : lancement de MATLAB Drive online	605
Figure 815 : partage d'un dossier à partir de MATLAB Drive online	606
Figure 816 : MATLAB Mobile	607
Figure 817 : ouverture de MATLAB Mobile	608
Figure 818 : utilisation des lignes de commandes dans MATLAB Mobile	608
Figure 819 : configuration MATLAB Keyboard	609
Figure 820 : accès à MATLAB Drive depuis MATLAB Mobile	609
Figure 821 : exécution d'un script depuis MATLAB Mobile	610
Figure 822 : configuration et démarrage d'une acquisition des mesures issues des capteurs du smartphone	611
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	611
Figure 824 : visualisation du fichier de mesure sur le PC	612
Figure 825 : visualisation du fichier de mesure dans le Workspace	613
Figure 826 : structure de la variable Acceleration	613
Figure 827 : extraction de la colonne des instants	614
Figure 828 : extraction de la colonne des instants correspondant aux secondes	614
Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X	615
Figure 830 : script permettant de faire tracer les accélérations en fonction du temps	615
Figure 831 : accélérations selon les trois axes en fonction du temps	616
Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine	617
Figure 833 : mesure de la tension aux bornes de la bobine	617
Figure 834 : barre de commande du scope	618
Figure 835 : mise en forme des courbes dans un scope	619
Figure 836 : modification de la mise en forme d'une courbe dans un scope	619
Figure 837 : modification du nombre d'entrées d'un scope	620
Figure 838 : raccordement d'un scope à deux entrées	620
Figure 839 : visualisation des deux courbes dans le même scope	621
Figure 840 : modification des caractéristiques de la deuxième courbe	621
Figure 841 : modification des caractéristiques de la deuxième courbe	622
Figure 842 : modification du nombre de fenêtres dans le scope	622
Figure 843 : affichage des courbes dans un scope sur plusieurs fenêtres	623
Figure 844 : utilisation des curseurs dans un scope	623
Figure 845 : fenêtre de paramétrage du Signal builder	624
Figure 846 : barre de commande du signal builder	625
Figure 847 : choix de la durée du signal dans le « signal builder »	625
Figure 848 : modification de l'échelle de temps	626
Figure 849 : déplacement manuel des segments dans le « signal builder »	626
Figure 850 : déplacement d'un point dans le « signal builder »	627
Figure 851 : placement de nouveaux points sur le signal	628
Figure 852 : mise en forme du signal par déplacement de segments et de points	628

Figure 853 : génération de plusieurs signaux.....	629
Figure 854 : visualisation des sorties multiples du « signal builder »	629

Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB – Simulink (R2020a)

pour l'étudiant et l'ingénieur

Quatrième édition



The MathWorks

Ivan LIEBGOTT