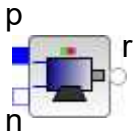
Motorisations et modélisation multiphysique

Cet article a pour objectif de présenter les différentes motorisations et leurs commandes proposées dans le module SIMM sous Scilab. Nous montrerons également comment réaliser son propre bloc de modélisation acausale en utilisant le langage Modelica.

Modélisation sous Scilab de différentes motorisations

Moteur à courant continu parfait

a) Modélisation

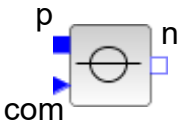


Le bloc DC_Motor disponible dans la palette Composants/Actionneurs de SIMM permet de simuler le comportement d'un moteur à courant continu sans frottement. Les paramètres demandés lorsque l'on clique sur le bloc sont : R résistance du moteur (Ohm), L inductance (H), k constante de couple ou de fcem (Nm), J inertie du rotor (kg.m^2).

En notant p et n les connecteurs de gauche du bloc (avec v_p potentiel électrique, i_p courant électrique) et r celui de droite (avec θ l'angle de rotation du moteur et C le couple disponible en sortie), les équations qui caractérisent le moteur sont :

$$v_p - v_n = Ri + L\frac{di}{dt} + k\dot{\theta}\ ,\quad i = i_p = i_n\ ,\quad ki - C = J\ddot{\theta}$$

b) Commande idéale



Pour faire varier la vitesse, il faut imposer une différence de potentiels entre les bornes p et n. La commande idéale consiste à utiliser un bloc de variateur de tension (palette Electrique/Sources/MEAS_SignalVoltage).

Les bornes p et n sont reliées aux bornes p et n du moteur et on impose au niveau de la borne com un signal quelconque (constant, sinusoïdal...). La tension aux bornes du moteur sera alors continue.

c) Commande par hacheur

Pour être plus proche de la commande réelle d'un MCC, on peut utiliser les différents drivers disponibles pour ce type de moteur (palette Composants/Préactionneurs) : Q1driver (hacheur 1 quadrant), Q2driver (hacheur 2 quadrants), Q4driver (hacheur 4 quadrants).

Les connecteurs p_s et n_s sont reliés aux connecteurs p et n d'une source de tension (alimentation) tandis que les connecteurs p et n sont reliés aux bornes du moteur. Le signal de commande de fermeture des interrupteurs du hacheur est envoyé sur le connecteur com. On utilise un bloc PWM pour générer un signal périodique de fréquence constante et de rapport cyclique variable.

Le modèle utilisé dans chaque hacheur est le modèle théorique avec diodes et interrupteurs idéaux.

Les figures suivantes montrent les modèles complets de pilotage de MCC avec chacun des hacheurs détaillés.

Pour obtenir des résultats pertinents au niveau de la tension du moteur, il faut utiliser un pas de calcul petit (le PWM est à une fréquence de 500 Hz). La simulation utilisée ici utilise un pas de 15000 pour une durée de simulation de 0.1s (on ne voit que le début du régime transitoire sur la vitesse moteur).

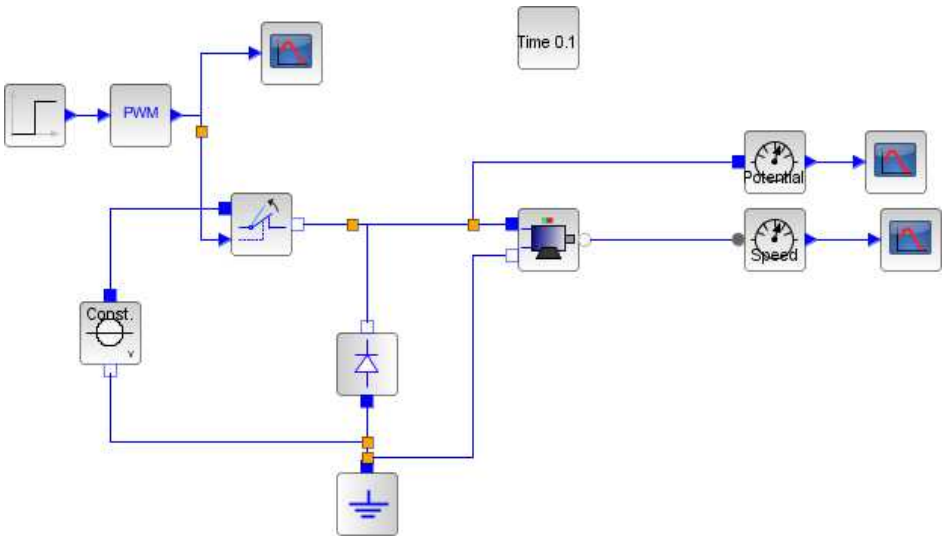


Figure 1: Pilotage d'un MCC avec hacheur 1 quadrant

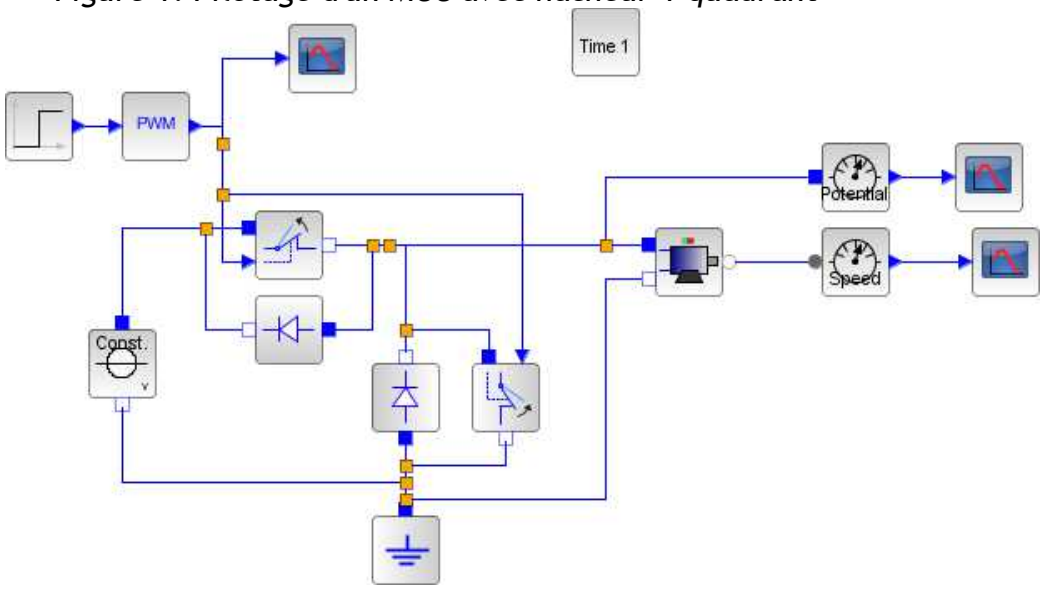


Figure 2: Pilotage d'un MCC avec hacheur 2 quadrants

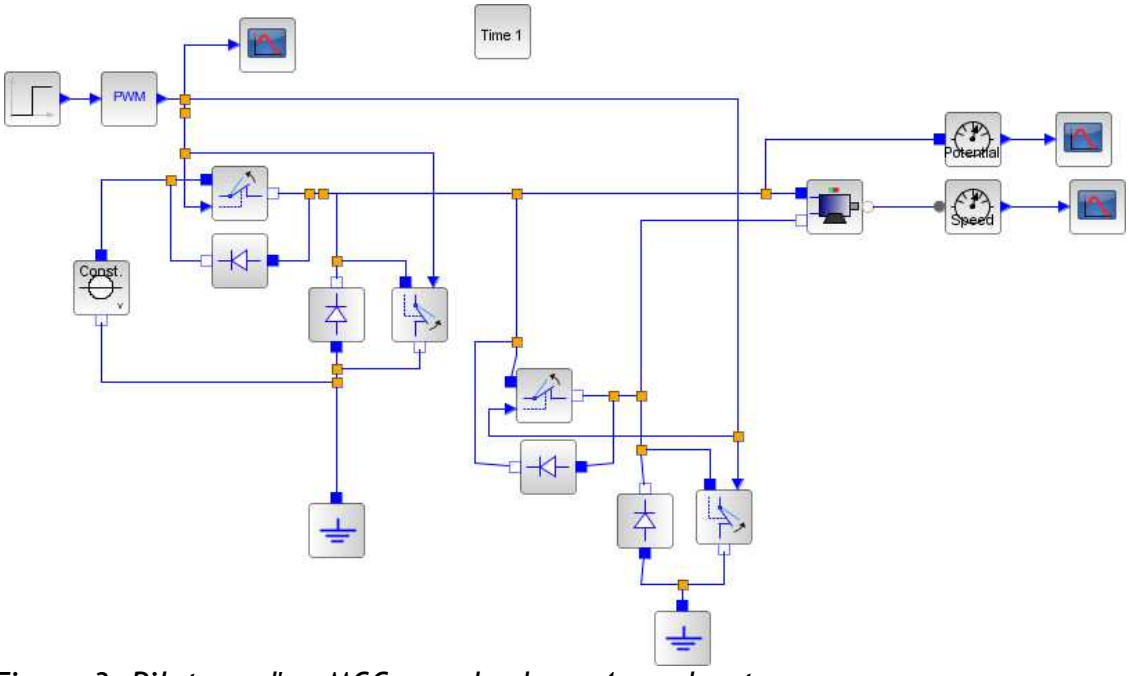


Figure 3: Pilotage d'un MCC avec hacheur 4 quadrants

d) Ajout de frottements

Le bloc proposé ici est valable pour tous les moteurs. Il consiste à ajouter des frottements visqueux et des frottements secs (selon la loi de Coulomb) : Mécanique/Rotation 1D/Basique/RotationalFriction.

Le modèle utilisé dans le bloc est le suivant : $C = f_v \dot{\theta} + C_r$ avec C la différence de couple entre le connecteur gauche et le connecteur droit). Dans l'équation mécanique du MCC, ce sera ce couple qui sera utilisé.

Le couple C_r vaut C_{r0} lorsque la vitesse est positive et $-C_{r0}$ lorsqu'elle est négative.

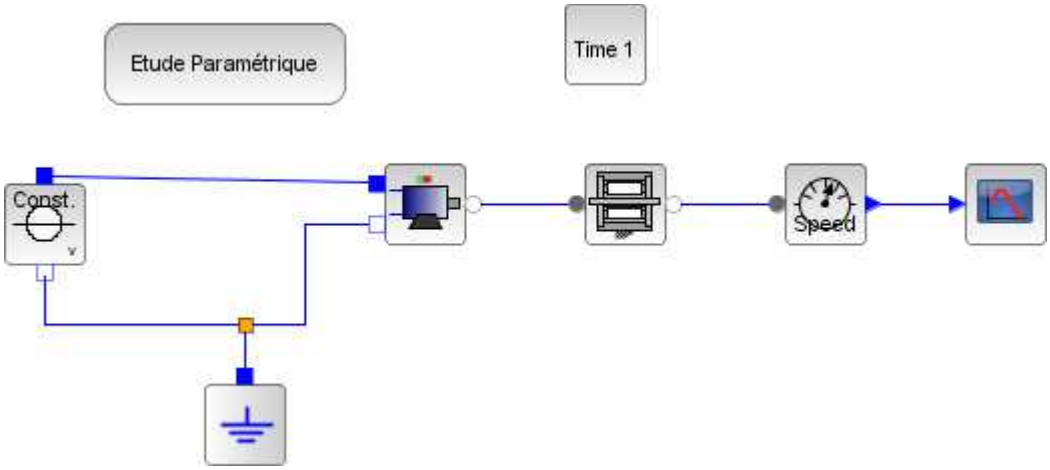


Figure 4: Modélisation d'un MCC avec frottements

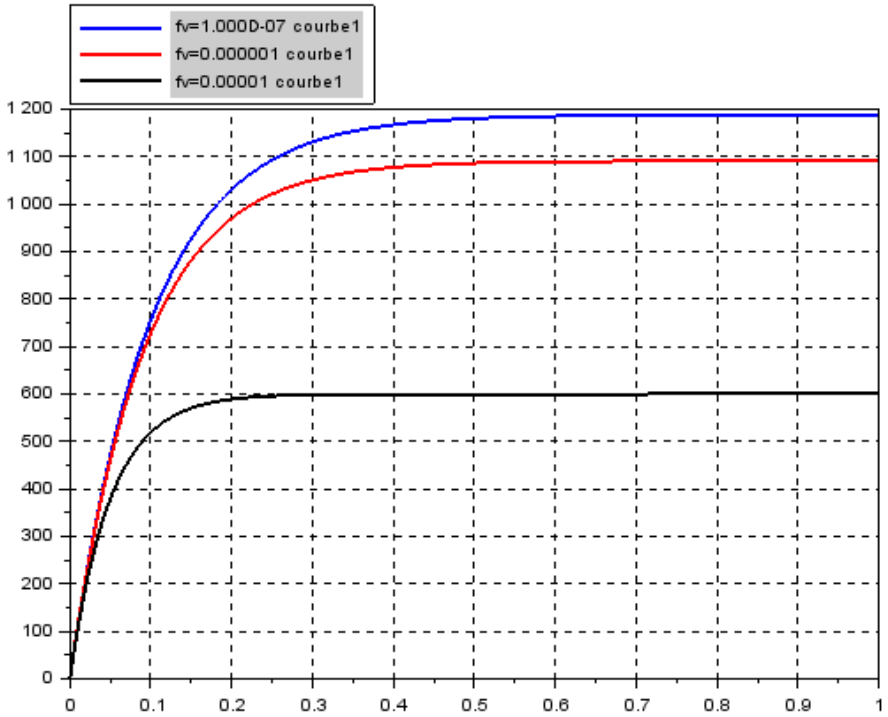


Figure 5: Vitesse du moteur en rad/s pour différentes valeurs de coefficient visqueux

Il est possible de mettre un couple de frottement sec nul, par contre on ne peut pas mettre un coefficient de frottement visqueux nul pour des raisons de convergence numérique. Il faut dans ces conditions prendre un coefficient très petit !

Remarque :un autre bloc est disponible mais il est plus complexe à utiliser (modèle de Stribeck inclus).

Servo-moteur

a) Modélisation

Un servo-moteur n'est autre qu'un moteur à courant continu asservi en position. Il contient en plus du MCC un réducteur de rapport de réduction ρ et un capteur qui donne une image de la position angulaire du moteur ou en sortie de réducteur. Un asservissement de type proportionnel est généralement inclus dans le servo-moteur.

Ainsi les équations simplifiées du servo-moteur sont les suivantes :

$u = Ri + k\dot{\theta}_m$, $ki - \frac{C}{\rho} = J\ddot{\theta}_m$, $\theta_r = \rho\theta_m$, $u = K_p(\theta_c - \theta_r)$.

Lorsqu'on achète un servo-moteur, les fabricants indiquent généralement :

- les dimensions ;
- les performances de vitesses : 0.21 s pour 60° à 4.8 V, 0.16 sec pour 60° à 6V ;
- les performances de couple : 4.6 kg.cm à 4.8 V, 5.7 kg.cm à 6V.

En se donnant une allure souhaitée du régime transitoire (sans dépassement), les valeurs de R, k, K_p ,J et ρ peuvent être choisies pour respecter les performances annoncées par le constructeur.

Ainsi le bloc servo-moteur disponible sous SIMM permet de simuler le comportement d'un servomoteur dont on donne les caractéristiques obtenues dans la documentation du servomoteur.

Le connecteur d'entrée correspond à l'angle désiré en degré (0 à 180 ° en général) et le connecteur de sortie est la rotation du servomoteur.

Les paramètres demandés sont :

- la tension d'alimentation en Volt,
- le couple nominal donné dans la documentation sous la tension U en N.m,
- le temps T que met le moteur à atteindre un angle donné en ° sous la tension U,
- l'angle atteint en une durée T.

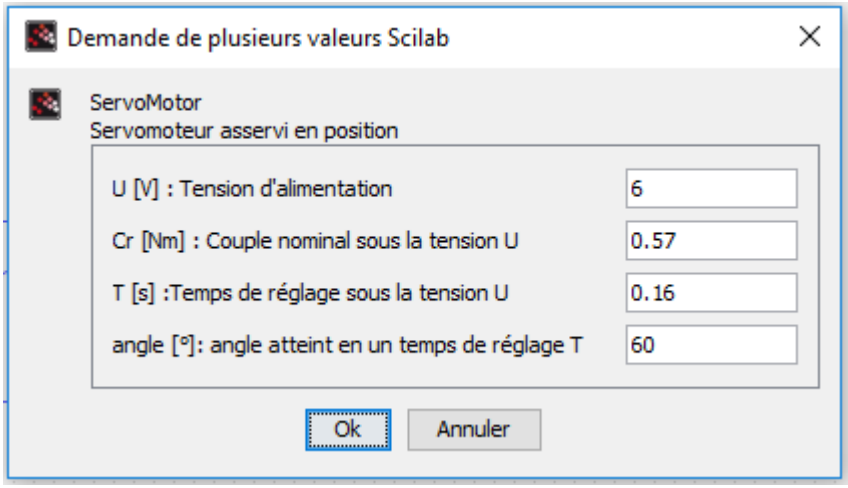


Figure 6: Paramètres de réglage d'un servomoteur

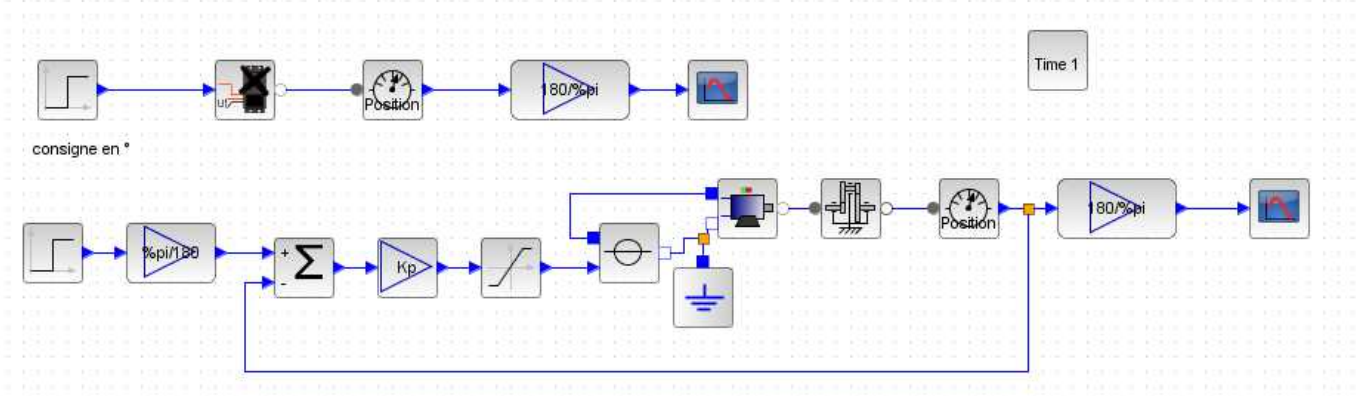


Figure 7: Simulation d'un servomoteur et équivalence avec le modèle détaillé du servomoteur

La simulation suivante

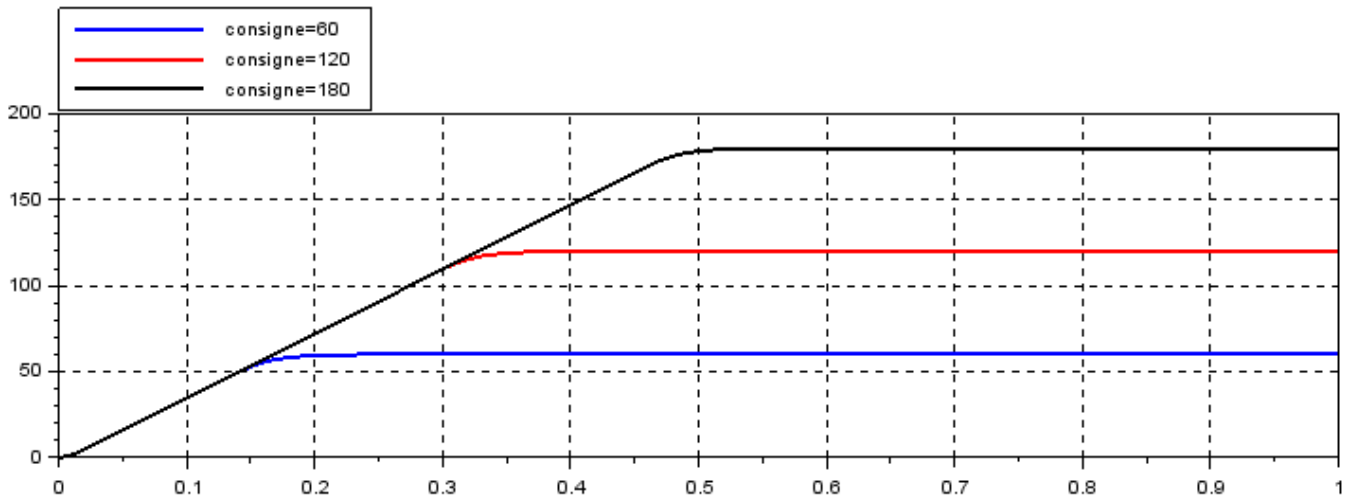
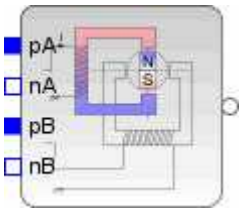


Figure 8: Position du servomoteur en degré

La simulation suivante représente l'angle en sortie du servomoteur lorsqu'on demande une consigne de 60, 120 ou 180°. On constate que les 60° sont atteints en 0.16 s comme indiqué dans la documentation du servomoteur utilisé. On observe une saturation de la tension de consigne du servomoteur qui se traduit par une vitesse constante pendant toute la phase de déplacement du servomoteur (l'asservissement n'apparaît que lorsque la position est proche de la consigne).

Moteur pas à pas

a) Modèle



La modélisation du moteur pas à pas a déjà été étudiée en détail dans un article précédent. On rappelle simplement ici les équations et simulations obtenues. Le moteur pas à pas bipolaire est constitué de deux enroulements indépendants appelés phases. Un bloc appelé StepperMotor est disponible dans la palette Composants/Actionneurs de SIMM.

Chaque phase est modélisée par l'association d'une inductance, d'une résistance et d'une fem sinusoïdale. L'induction engendre deux relations entre, d'une part, la vitesse du rotor (notée ω) et la tension induite e et d'autre part entre le courant i dans la phase et le couple magnétique exercé C, ceci en fonction du décalage entre l'angle du champs magnétique et l'angle du rotor. L'angle magnétique est donné par la relation $\phi_e = n\phi$ où ϕ est l'angle de rotation du rotor et n le nombre de paires de pôles aimantés (50 pour un moteur à 200 pas par tour).
On note p_A et n_A les deux bornes de la phase A et p_B, n_B celles de la phase B.
On a donc pour la phase A les relations suivantes :

$$v_{p_A} - v_{n_A} = R\,i_A + L\frac{di_A}{dt} + e_A\,,\quad e_A = k\,\omega\sin(n\phi)\,,\quad C_A = n\,k\,i_A\sin(n\,\phi)$$

Pour la phase B, les équations sont les mêmes mais tournées de 90° électriquement.

$$v_{p_B} - v_{n_B} = R\,i_B + L\frac{di_B}{dt} + e_B\,,\quad e_B = k\,\omega\sin\left(n\phi + \frac{\pi}{2}\right)\,,\quad C_B = n\,k\,i_B\sin\left(n\phi + \frac{\pi}{2}\right)$$

L'équation mécanique du rotor soumis aux deux actions magnétiques et à un couple extérieur C_r est la suivante : $C_B + C_A - C_r - f_v\,\omega = J\,\dot{\omega}$ avec J moment d'inertie du rotor et f_v coefficient de frottement visqueux.

Le modèle qui est ainsi décrit et qui est implanté dans le bloc StepperMotor est équivalent au schéma-bloc suivant

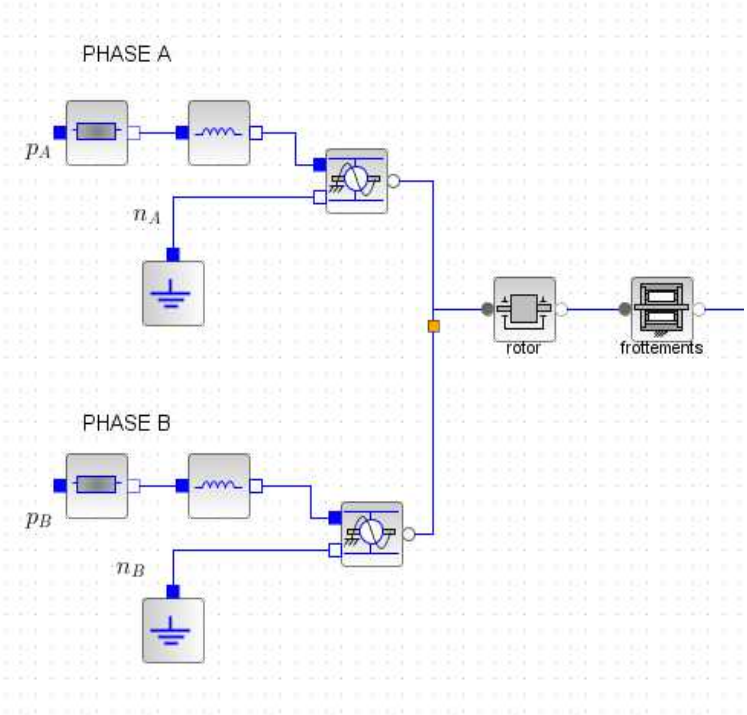
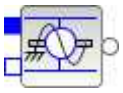


Figure 9: Modèle d'un moteur pas à pas bipolaire.

Le bloc  (palette Electrique/Composants basiques/Passif/EMF_SINUS) fournit les relations de couplage électro-mécaniques. Il faut renseigner dans ce bloc le nombre de paires de pôles n, la constante de couple et le déphasage entre les champs (en radian).

Remarque : il est indispensable de mettre un coefficient de frottement visqueux assez important pour assurer la convergence des calculs lors des simulations. De même si le schéma numérique ne réussit pas à converger, il est possible de mettre 0 pour la valeur de l'inductance ce qui simplifie la résolution.

b) Pilotage du moteur

Pour piloter le moteur pas à pas, il est nécessaire de générer des courants pas à pas sur chaque bobine.
Plusieurs modes sont possibles et décrits sur les figures suivantes où on représente l'évolution du courant entre -1 et 1 A avec une période de commutation identique T=0.001s :

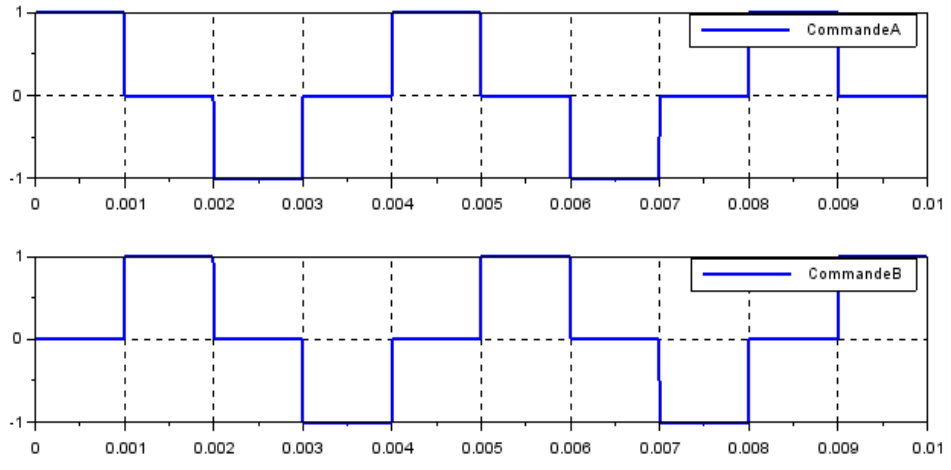


Figure 10: Commande par pas entier

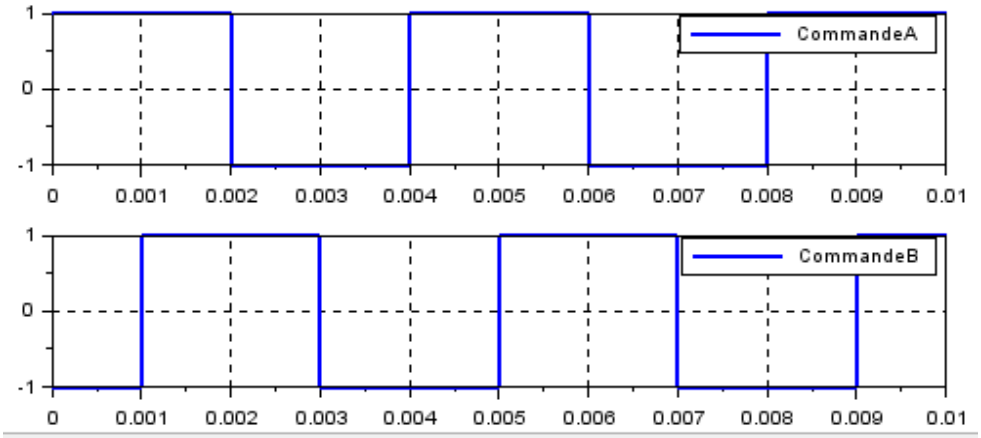


Figure 11: Commande par pas entier à couple maximal

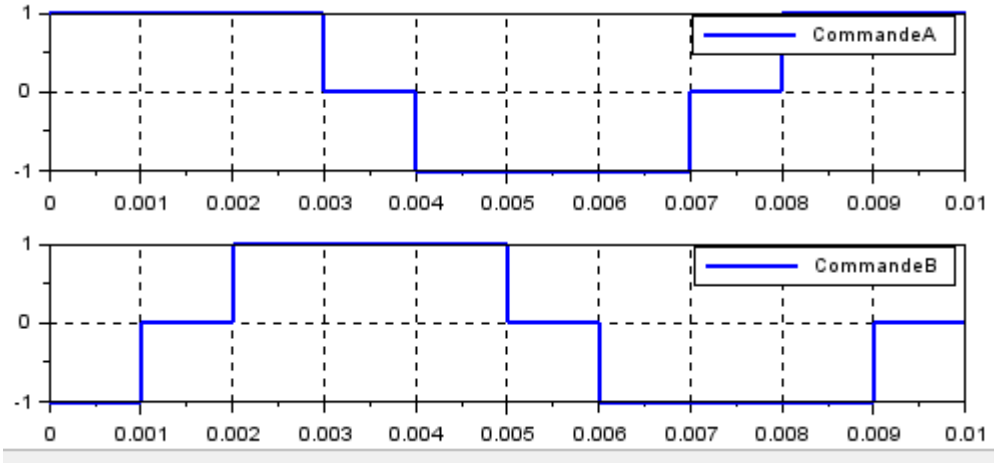
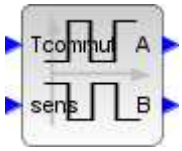


Figure 12: Commande par demi-pas



Si on inverse les commandes A et B, on change le sens de rotation du moteur.

Le bloc SteppingSequence disponible dans la palette Composants/Préactionneur permet de générer automatiquement ces signaux.

Il faut renseigner, comme paramètres du bloc, l'amplitude souhaitée du signal, le décalage temporel initial si nécessaire (0 par défaut) et le mode (0 pour pas entier, 1 pour pas entier à couple maximal et 2 pour demi-pas). Deux entrées sont nécessaires pour faire fonctionner ce bloc : la durée de commutation en seconde et le sens noté 0 ou 1.

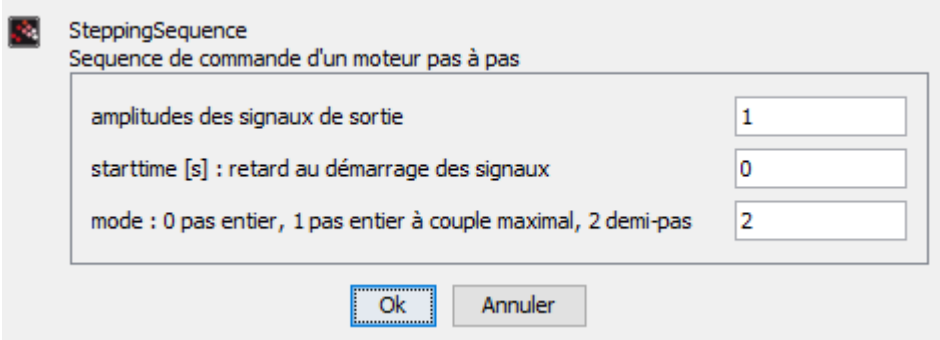
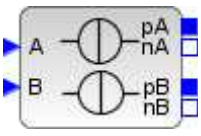


Figure 13: Paramétrage du bloc de commande



Le signal ainsi émis peut être envoyé comme consigne d'asservissement de courant sur chaque phase. Pour remplacer l'asservissement de courant, un bloc représentant 2 sources de courant idéales est disponible dans la palette Composants/Préactionneurs/IdealCurSourcesStepper

On peut également piloter les deux phases en imposant la différence de potentiel (source de tension contrôlée par le signal périodique) mais le pilotage sera moins performant.

Le modèle suivant permet alors de tester le pilotage de moteurs pas à pas.

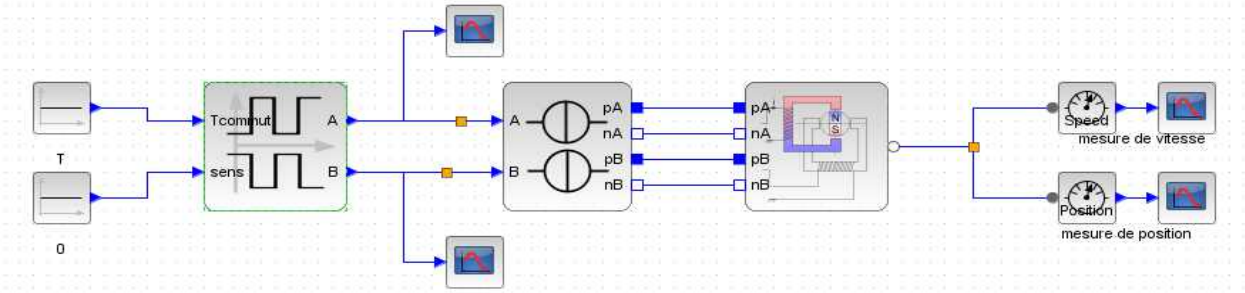


Figure 14: Modélisation de la commande d'un moteur pas à pas

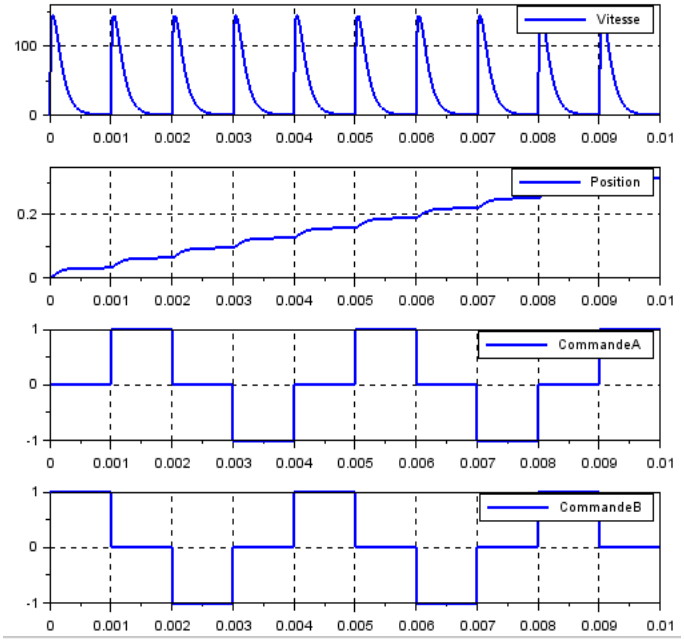


Figure 15: Résultats de simulation pour un mode de fonctionnement en pas entier

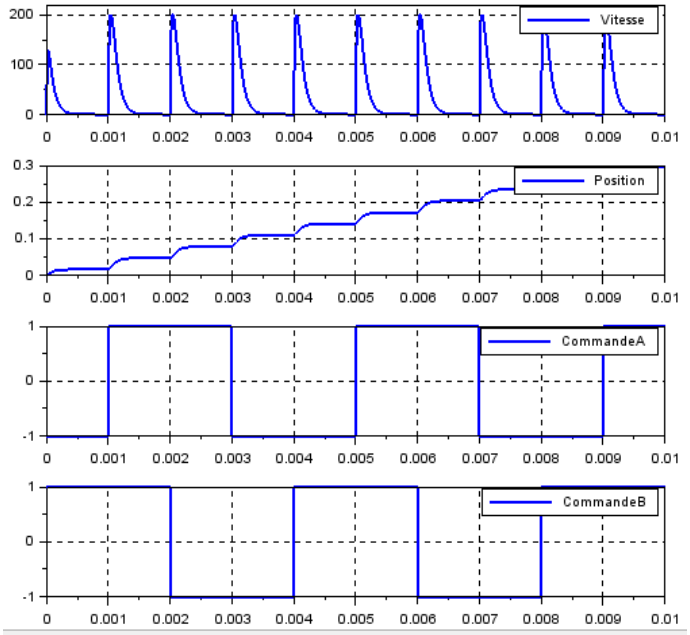


Figure 16: Résultats de simulation pour un mode de fonctionnement en pas entier à couple maximal

On observe bien que le mode demi-pas est plus lent mais par contre plus précis.

La vitesse s'annule entre chaque pas et assure ainsi que la distance souhaitée à chaque pas est correcte. Ceci montre les limites que l'on peut prendre pour le temps de commutation T (ici environ 0.0005 s.

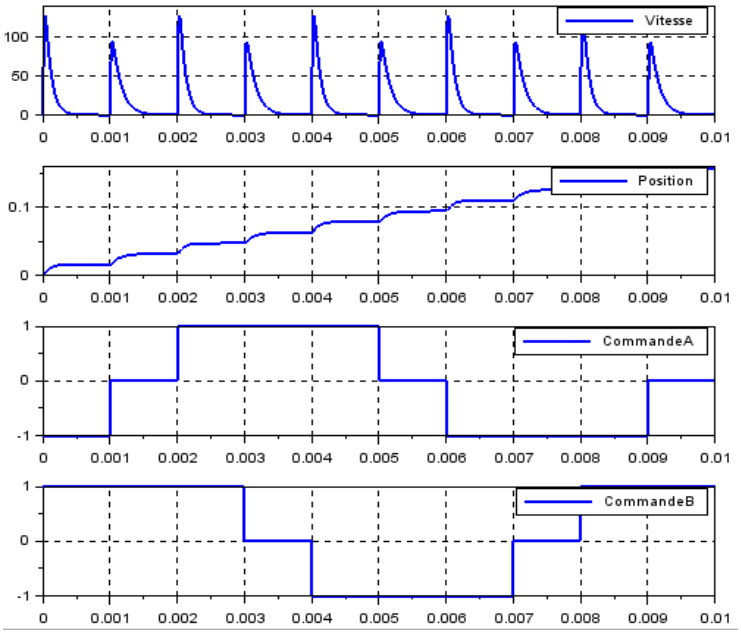
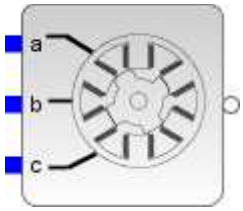


Figure 17: Résultats de simulation pour un mode de fonctionnement en demi-pas

Moteur synchrone

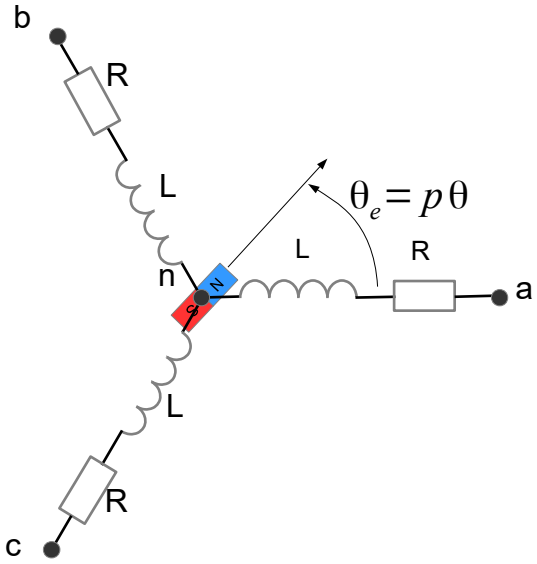
a) Modèle

Un modèle parfait de moteur synchrone est disponible dans la palette SIMM/Composants/Actionneurs PermanentMagnetSynchronousMotor (Moteur synchrone à aimants permanents) ou WoundedRotorSynchronousMotor (moteur synchrone à rotor bobiné). Ces deux modèles sont identiques à la différence près que le moteur à aimants permanents possède un flux magnétique constant ϕ_s alors que le flux dans le cas d'un rotor bobiné dépend du courant imposé à l'extérieur du bloc et peut donc être variable.



Nous nous intéressons au cas du moteur synchrone à aimants permanents. On retrouve à gauche 3 connecteurs qui correspondent aux 3 fils de chaque phase du moteur. En sortie un connecteur de type rotation 1D est défini et permet d'avoir accès à la position angulaire et au couple.

On note θ l'angle de rotation du moteur et v_a, v_b, v_c les potentiels aux bornes de chaque phase ainsi que i_a , i_b , i_c les courants circulant dans chaque phase.
Le montage électrique qui correspond au modèle implanté dans le bloc est le suivant avec n nœud central.



D'un point de vue électrique, le champ magnétique tourne d'un angle électrique θ_e fonction du nombre de paires de pôles p de l'aimant, ainsi $\theta_e = p\theta$.
En notant n le nœud central du montage étoile, les équations électriques qui caractérisent le modèle du moteur synchrone sont :

$$v_a - v_n = Ri_a + \frac{d\phi_a}{dt} \quad \text{avec} \quad \phi_a = L i_a + M(i_b + i_c) + \phi_s \cos\theta_e$$

$$v_b - v_n = Ri_b + \frac{d\phi_b}{dt} \quad \text{avec} \quad \phi_b = L i_b + M(i_a + i_c) + \phi_s \cos\left(\theta_e - \frac{2\pi}{3}\right)$$

$$v_c - v_n = Ri_c + \frac{d\phi_c}{dt} \quad \text{avec} \quad \phi_c = L i_c + M(i_a + i_b) + \phi_s \cos\left(\theta_e + \frac{2\pi}{3}\right)$$

et $i_a + i_b + i_c = 0$
où L et R sont les inductances et résistances d'une phase et M l'inductance mutuelle entre les phases (on a généralement $M = L\cos(120°)$)

D'un point de vue mécanique, l'équation qui relie l'angle du rotor au couple magnétique est la suivante :
$J\ddot{\theta} = p\phi_s\left(\sin(\theta_e) i_a + \sin\left(\theta_e - 2\frac{\pi}{3}\right) i_b + \sin\left(\theta_e + 2\frac{\pi}{3}\right) i_c\right) - f_v\dot{\theta} + C_s$ avec C_s le couple disponible en sortie, f_v coefficient de frottement visqueux et J inertie du rotor.

b) Pilotage par onduleur

Pour valider le modèle de moteur synchrone, on impose sur les trois phases des tensions sinusoïdales décalées de 120° d'amplitude donnée 12 V avec une petite fréquence pour éviter le décrochage du moteur f=1Hz.

Les paramètres du moteur sont les suivants

SynchronousMotor
Moteur synchrone

R [Ohm]: Résistance statorique	1
L [H] : Inductance propre des enroulements statoriques	0.001
M [H] : Inductance mutuelle entre 2 enroulements statoriques	-0.0005
p : nombre de paires de pôles du rotor	2
J [kg.m^2]: moment d'inertie du rotor	0.000001
fv [Nm.rads^-1]: coefficient de frottement visqueux	0.001
phif [W]: flux permanent constant créé par l'aimant permanent	0.001

Figure 18: Paramètres du moteur synchrone

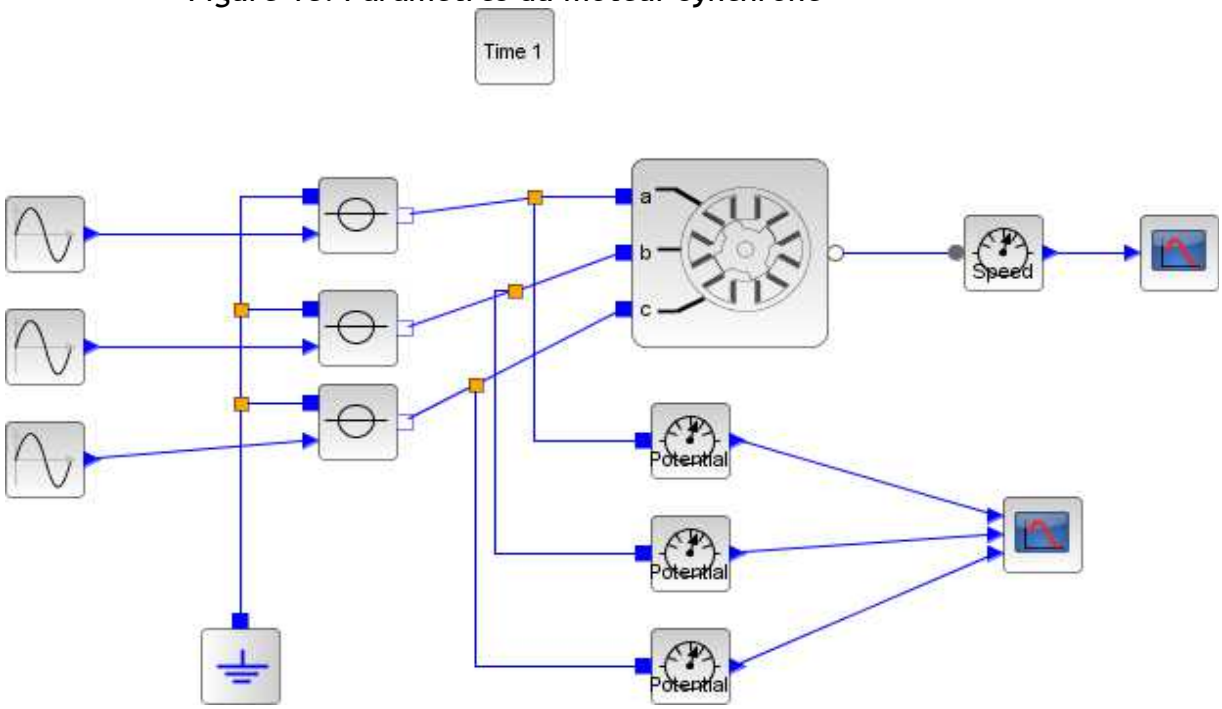


Figure 19: Schéma-bloc du pilotage en boucle ouverte du moteur synchrone

On trace la vitesse angulaire du rotor et la tension aux bornes de chaque phase moteur.

On constate que la vitesse du rotor après un régime transitoire est de π rad/s ce qui correspond à la fréquence des tensions imposées en entrée.

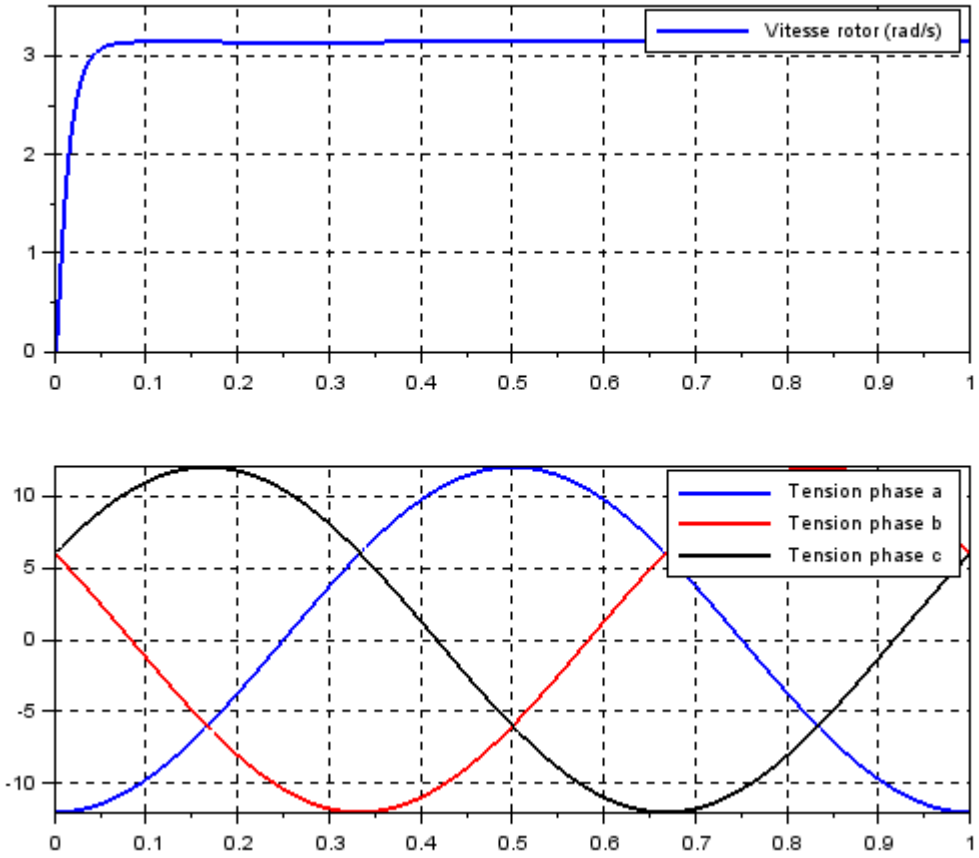


Figure 20: Résultats de simulation pour des tensions d'entrée de fréquence 1 Hz

Il est intéressant de faire varier la fréquence des tensions d'entrée pour visualiser le décrochage du moteur. A partir d'une certaine fréquence, soit le calcul ne se fait pas, soit on observe que la vitesse en régime permanent n'est plus constante ce qui montre que la simulation n'est pas réaliste, pas physique (il n'y a pas de frottement sec).

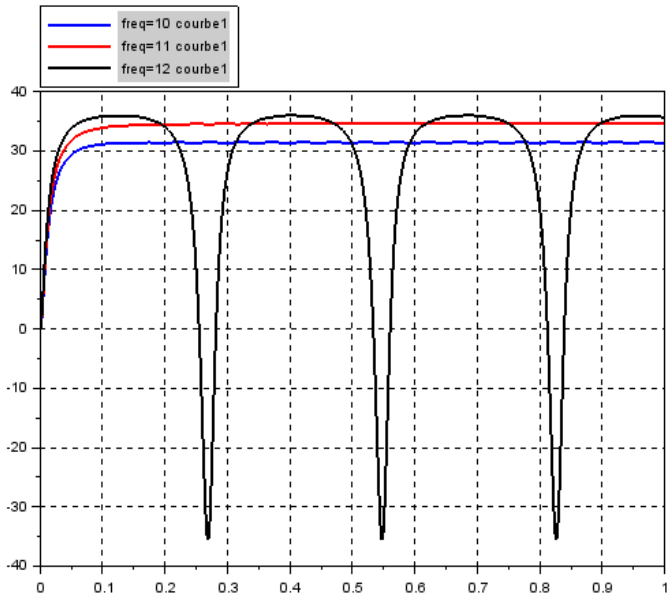
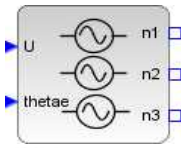


Figure 21: Courbes de vitesse du rotor pour des fréquences de 10,11 et 12 Hz

La génération des trois sinus déphasés de fréquence et d'amplitude données est disponible dans un bloc Onduleur parfait (SIMM/Composants/Préactionneurs/IdealInverter). Ce bloc demande en entrée l'amplitude de la tension de commande ainsi que l'évolution de l'angle électrique au cours du temps. On peut donner en paramètre un déphasage initial entre le champ magnétique et le rotor d'angle nul initialement.



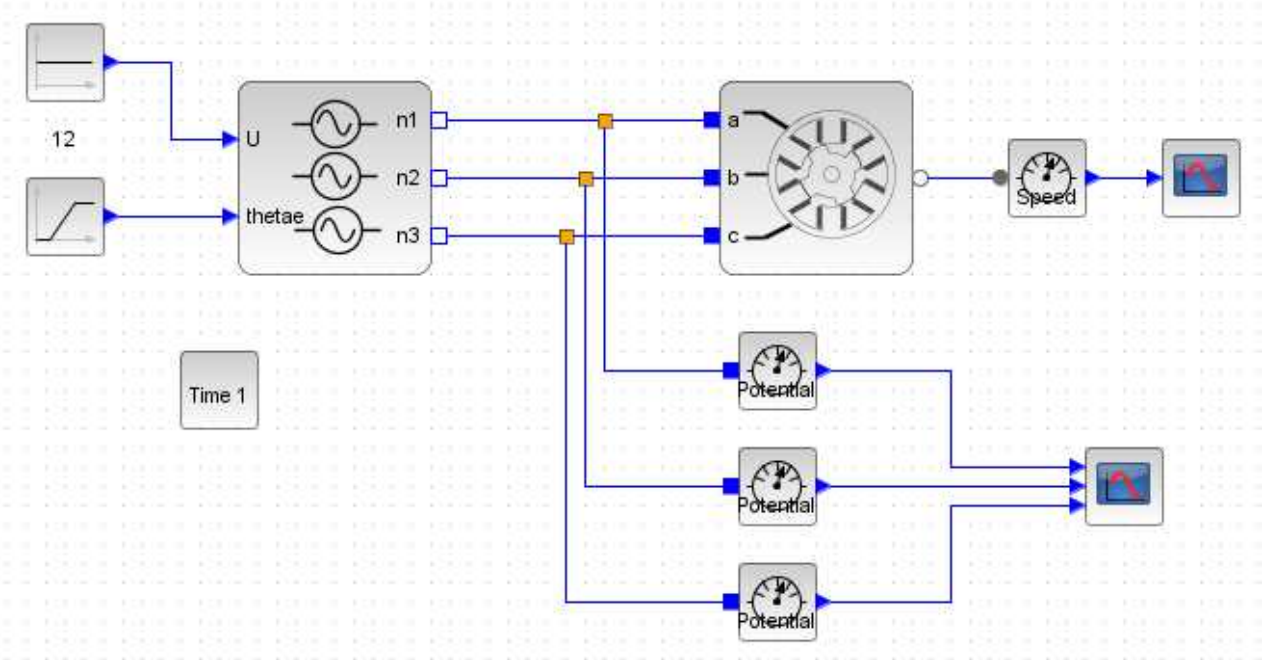


Figure 22: Schéma-bloc du pilotage en boucle ouverte par onduleur parfait

c) Autopilotage par onduleur et capteur de position

Pour que le moteur tourne à une vitesse imposée, on mesure précisément l'angle du rotor et on en déduit l'angle électrique de consigne nécessaire puis on l'envoie en entrée du bloc onduleur.
Pour voir l'influence de l'autopilotage, on modifie l'inertie du moteur J=0.00001 kg.m^2 et le frottement visqueux fv=0.0001 Nm.s.

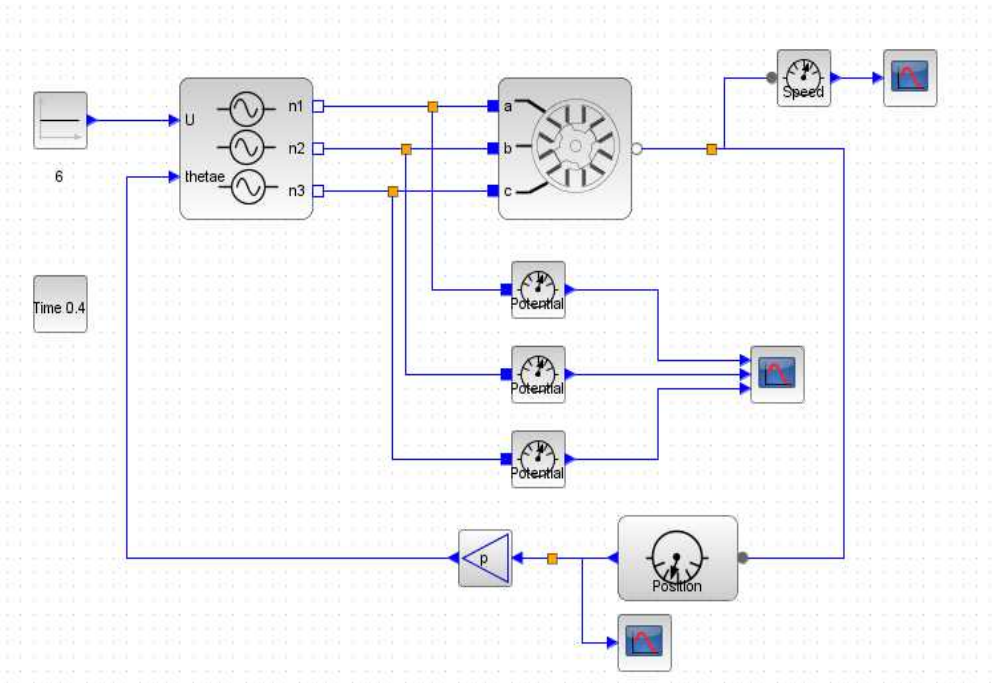


Figure 23: Schéma-bloc du moteur synchrone autopiloté

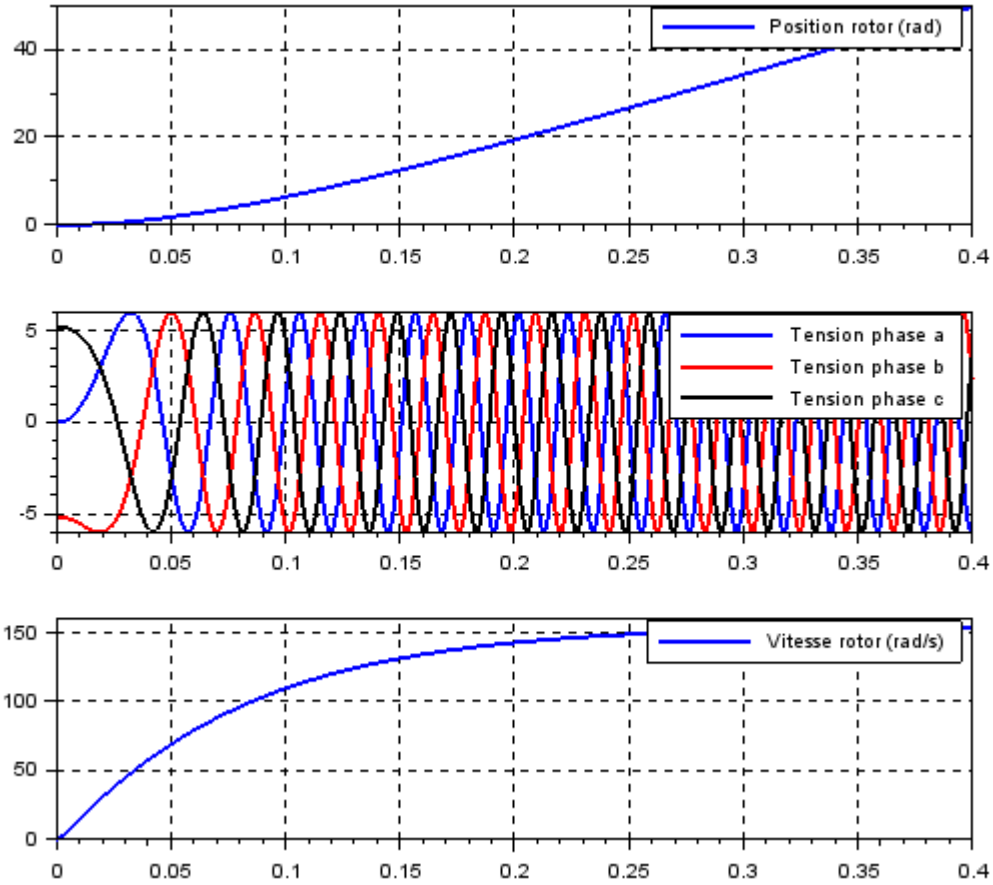
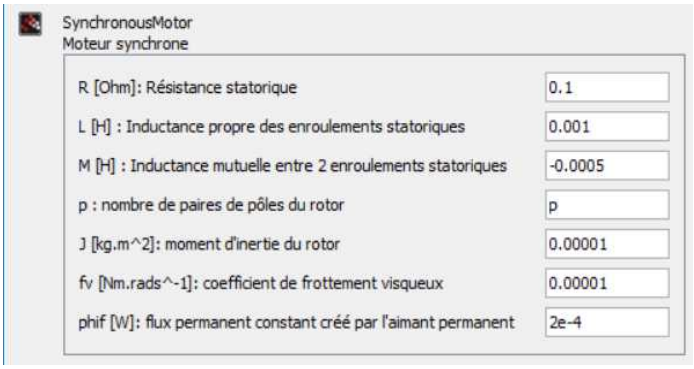


Figure 24: Evolution des grandeurs pour un fonctionnement autopiloté

On constate la variation automatique de fréquence des sinusoïdes pendant le régime transitoire. On remarque que la courbe de vitesse de la machine synchrone autopilotée ressemble à celle d'un moteur à courant continu.

d) BLDC

On utilise les caractéristiques suivantes pour le moteur synchrone utilisé comme moteur brushless mais on change la commande qui est cette fois-ci discrète.



Trois capteurs à effet hall permettent de déterminer quand le champ magnétique sur chaque phase est positif ou négatif et renvoie donc 1 ou 0 en fonction de l'angle électrique.
A partir de ces trois informations binaires des capteurs (Ca, Cb, Cc), on reconstruit 6 états de fonctionnement de l'onduleur qui imposera les potentiels au + ou au moins -, ou bien le courant à 0 (phase transitoire (+) ou (-)).
Les valeurs des potentiels sont rappelées sur le schéma de la figure 25.

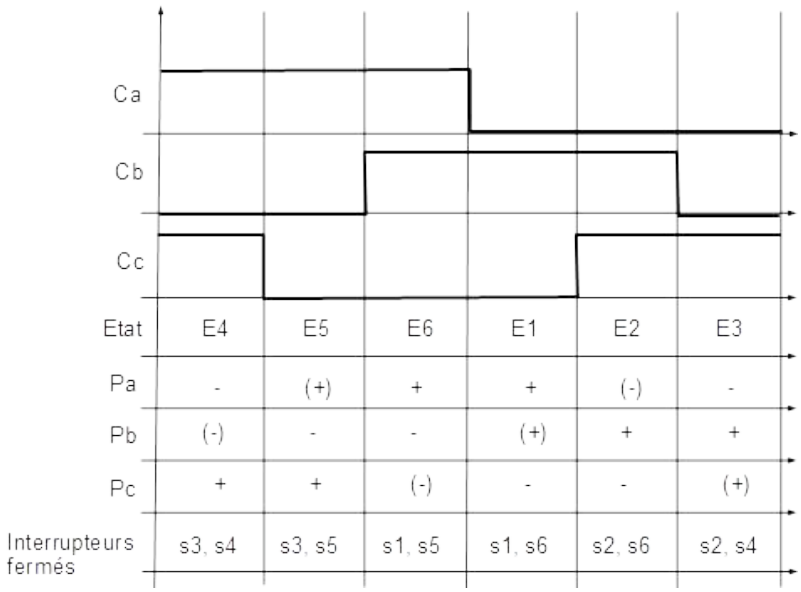
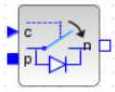


Figure 25: Etat des capteurs et potentiels associées



L'onduleur est constitué de 6 interrupteurs numérotés s1 à s6 avec une diode en parallèle pour permettre la continuité de courant lorsque les interrupteurs passent d'un état fermé à un état ouvert (cette association d'un interrupteur et d'une diode en parallèle est disponible dans la palette SIMM/Electrique/Basique/PowerTransistor

Le schéma de la figure 26 représente le modèle utilisé pour l'onduleur BLDC disponible dans la palette SIMM/Composants/Préactionneur/OnduleurBLDC.

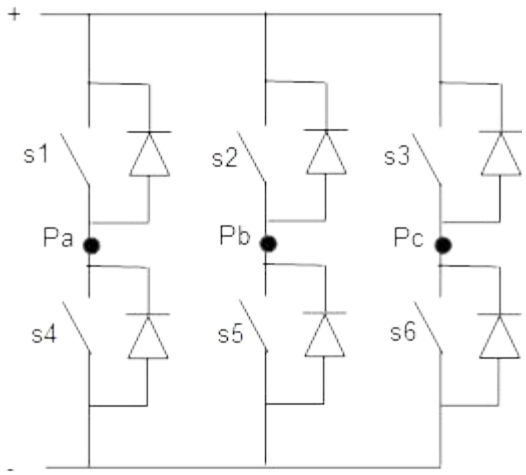


Figure 26: Onduleur pour BLDC

On obtient alors le modèle acausal suivant pour le pilotage par capteur à effet Hall d'un moteur brushless :

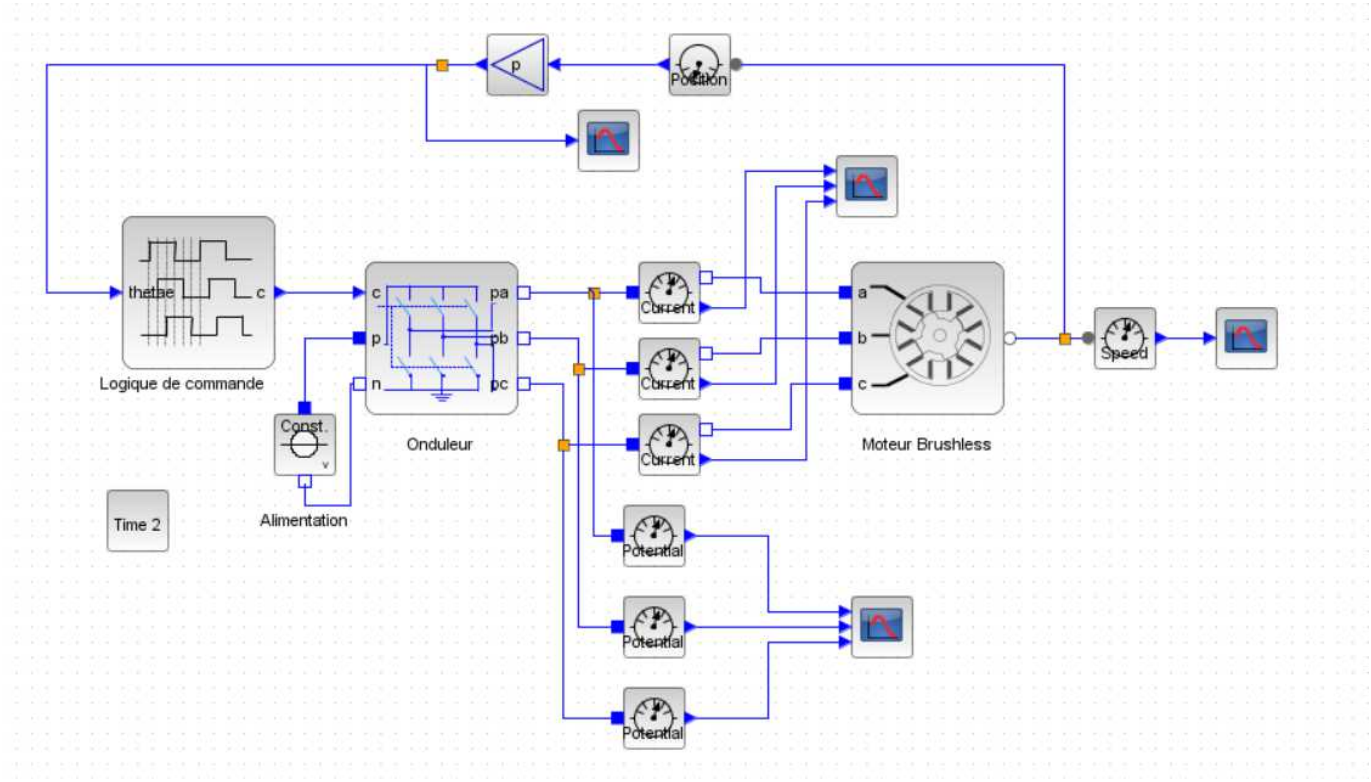


Figure 27: Modèle acausal de la commande d'un moteur brushless

On relève les tensions et courants dans chaque phase ainsi que la vitesse du moteur et la position de celui-ci.

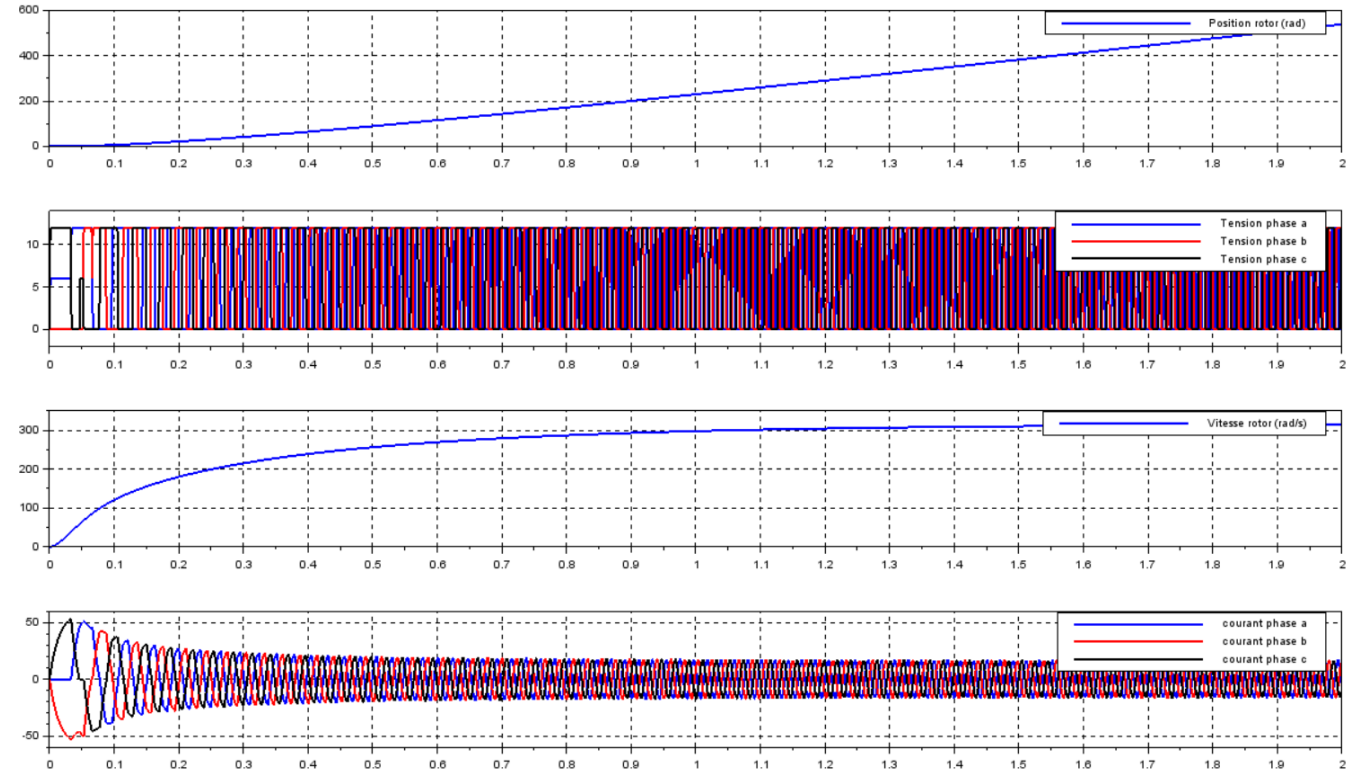


Figure 28: Grandeurs observées lors du fonctionnement du moteur BLDC

On observe à nouveau que la vitesse du rotor croit selon la caractéristique d'un modèle du premier ordre. Après un régime transitoire où les courants sont très importants, on observe une phase de stabilisation sinusoïdale avec des amplitudes de l'ordre de 20 A.

Les signaux électriques peuvent être exploités (directement dans Scilab) pour tracer notamment le diagramme

de Fresnel.

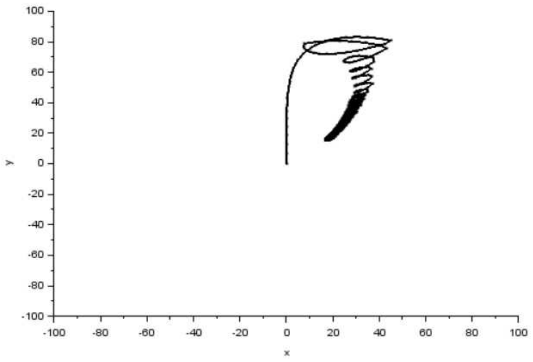


Figure 29: Diagramme de Fresnel

Modélisation multiphysique en Modelica sous Scilab

La librairie SIMM offre différents blocs spécifiques à des domaines multiphysiques variés. Cependant, un utilisateur peut vouloir développer son propre comportement multiphysique. Un nouveau bloc a été implanté dans SIMM et permet de définir tout type de comportement dans Scilab. Il est nécessaire pour bien comprendre son utilisation de connaître les connecteurs physiques utilisés dans SIMM et également la syntaxe basique de Modelica.

Connecteurs physiques

Les blocs sous SIMM possèdent des connecteurs à gauche et à droite qui représentent une connexion physique possible. Chaque connecteur fait transiter des grandeurs de type flux ou de type potentiel. Lorsque deux blocs sont reliés, il faut s'assurer que les connecteurs sont de même type.

Le tableau suivant liste les symboles utilisés dans SIMM ainsi que les noms des grandeurs physiques qui transitent dans chaque connecteur.

Symboles	Type de connecteur	Raccourcis pour les types	Grandeurs accessibles dans le connecteur noté c
	Electrique	El+, El-	c.v : potentiel en V c.i : courant en A
	Hydraulique	Hy+, Hy-	c.p pression en Pa c.q débit en m^3/s
	Aéraulique	Ae+, Ae-	c.p pression en Pa c.q débit m^3/s" c.h taux d'humidité en %
	Rotation 1D	Ro+, Ro-	c.phi angle en rad c.tau couple en Nm
	Translation 1D	Tr+, Tr-	c.f force en N c.s position en m
	Mécanique plane	Pl+, Pl-	c.f force 2D en N c.t couple en Nm c.r_0 position absolue en m c.phi angle en rad
	Thermique	Th+, Th-	c.T température en Kelvin c.Q_flow flux thermique en W
	Signaux	Si+ (Si-)	c.signal (sans unité)

Les connecteurs peuvent être pleins ou vides. Ceci permet de fixer une convention pour les équations utilisés dans les blocs.

Exemple : bloc alimentation 

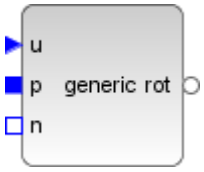
On note p le connecteur de gauche et n le connecteur de droite. Le bloc demande le paramètre correspondant à la tension constante d'alimentation U.

Les équations qui sont utilisées dans le bloc sont les suivantes : p.v-n.v=U et p.i=n.i

Il y a donc bien une convention qui indique que le potentiel positif est le p.

De la même manière pour la résistance : p.v-n.v=R*p.i et p.i=n.i

Bloc pour définir un modèle personnel

Le bloc de la palette SIMM/Utilitaires/Analyse est appelé SIMMUserModel 

En cliquant sur le bloc, on accède à la première fenêtre de paramétrage.

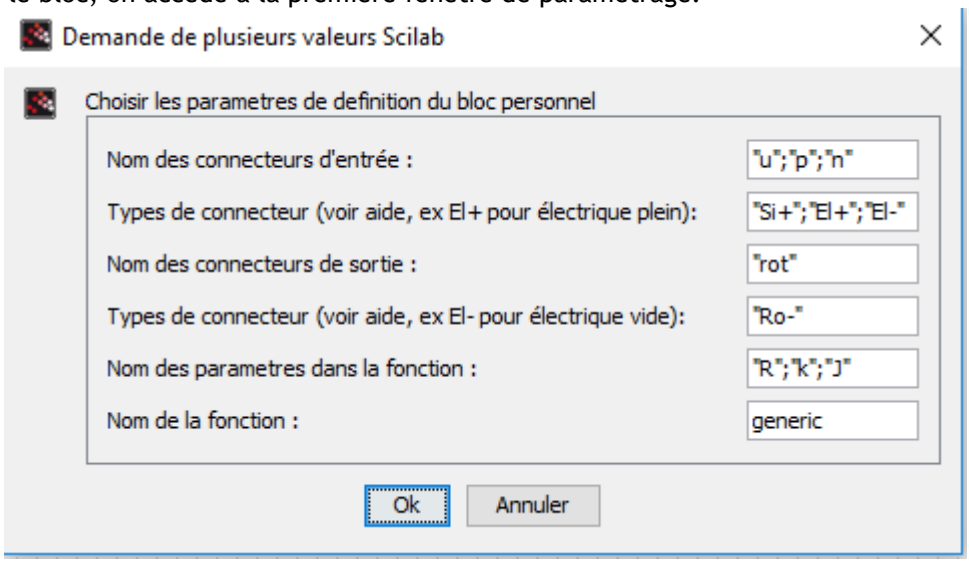
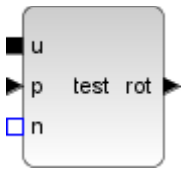


Figure 30: Fenêtre de paramétrage du bloc SIMMUserModel

La première entrée correspond aux noms des connecteurs que l'on souhaite utiliser en entrée (connecteurs de gauche du bloc). Les noms doivent être donnés sous forme de chaînes de caractères séparées par des virgules : "u" ;"p"

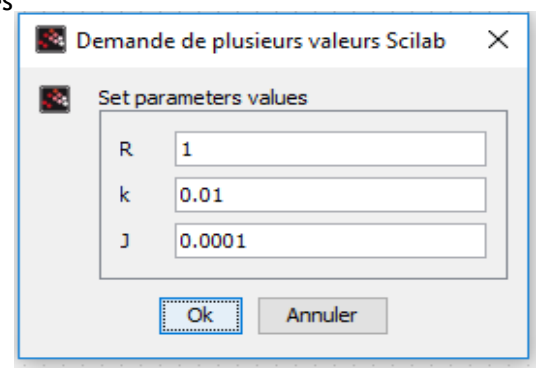
Il faut ensuite spécifier le type de connecteur correspondant à chaque entrée. Scilab mettra automatiquement la bonne forme et bonne couleur du connecteur une fois la génération réalisée. Les noms des types sont à choisir parmi ceux rappelés dans le tableau du paragraphe précédent et doivent être renseignés sous forme de chaînes de caractères séparées par des virgules. Exemple : "Si+" pour un connecteur de type signal plein (triangle bleu plein), "El-" pour un connecteur de type électrique vide.

Il est possible également d'utiliser les connecteurs génériques de Scilab, c'est à dire le triangle noir pour représenter des grandeurs de type symbolique (type explicite "E"), ou le carré noir pour représenter des grandeurs basiques réelles Modelica (type implicite "I"). 

Les deux entrées suivantes permettent de définir les noms et les types des grandeurs de sortie (du côté droit du bloc) en suivant le même principe que pour les grandeurs de gauche.

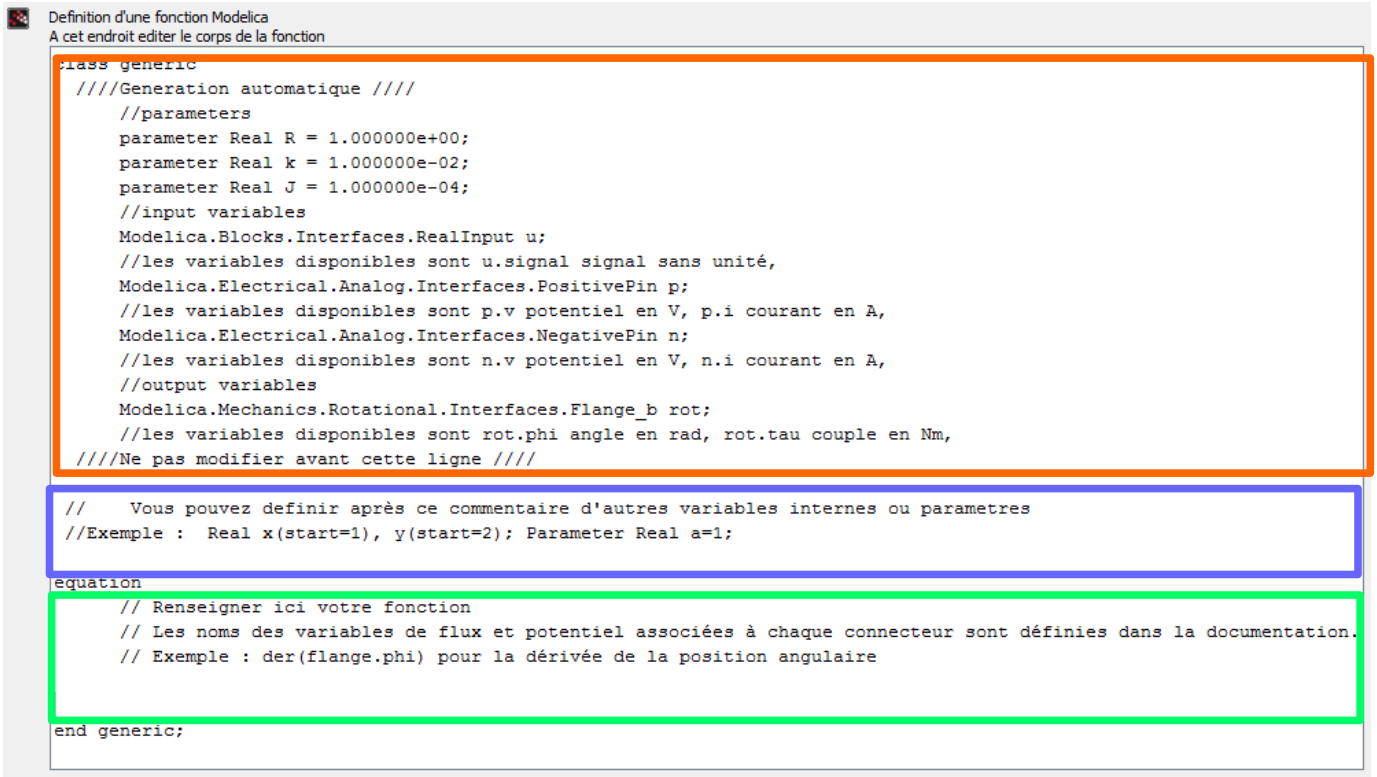
Il est ensuite possible de définir des paramètres constants passés dans la fonction et qui seront utilisés (ou non) dans le modèle de la fonction. Si le champ est laissé vide, aucun paramètre constant ne sera utilisé.

Pour terminer il faut renseigner le nom souhaité pour la fonction. Il ne peut pas y avoir plusieurs fonctions qui portent le même nom ! Ainsi si deux blocs SIMMUserModel sont utilisés pour représenter le même comportement mais avec des paramètres différents, il faudra renommer un des blocs.



La deuxième fenêtre de paramétrage demande de renseigner les valeurs des paramètres constants.

La dernière fenêtre comporte le corps de la fonction à définir par l'utilisateur.



La partie orange est générée automatiquement à partir des variables et types définis précédemment. Il ne faut pas la modifier car elle est automatiquement écrasée et générée à chaque fois.

On retrouve la définition des paramètres constants en langage Modelica, la définition des variables d'entrée et de sortie avec le type et le nom de chaque variable. Une aide est proposée après chaque ligne de déclaration de variable pour savoir comment utiliser les variables dans la fonction Modelica. Cet aspect sera décrit dans le paragraphe suivant.

La zone bleue est utilisée pour définir des variables internes utilisées dans la fonction. Cette zone reste si on modifie des éléments dans les fenêtres de paramétrage sauf si on modifie le nom de la fonction.

La zone verte permet de définir la fonction Modelica et n'est pas effacée si les paramètres de définition sont changés.

Eléments de langage Modelica

a) Exemple de modèle simple

La déclinaison de Modelica utilisée dans Scilab n'est pas la version la plus actuelle de Modelica. Les éléments syntaxiques sont relativement restreints.

Quelques règles doivent être respectées :

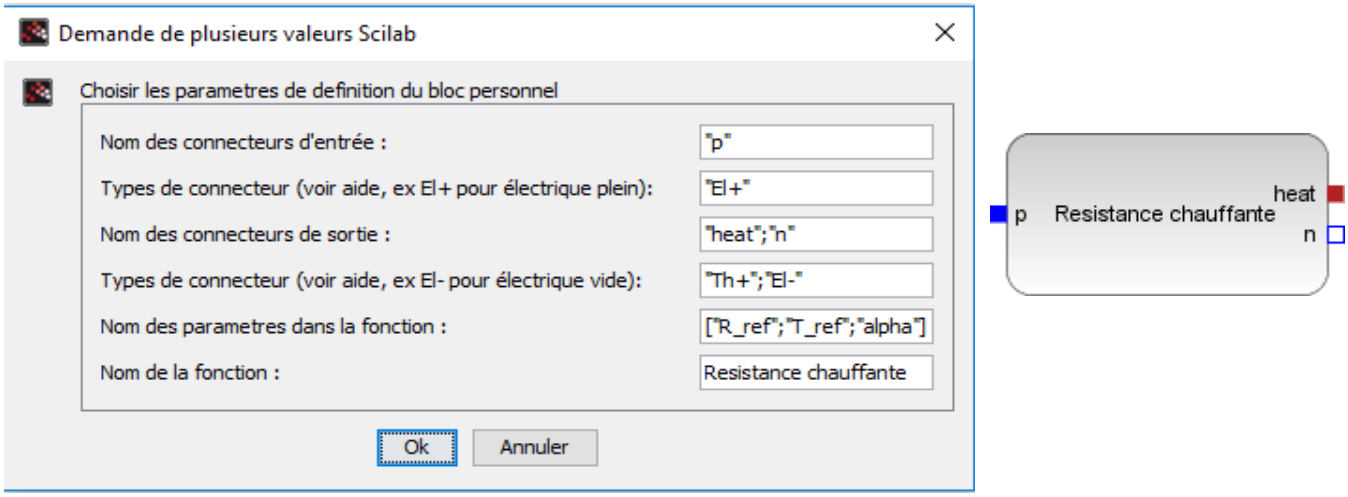
- Chaque ligne d'instruction doit être terminée par un point-virgule. Les commentaires commencent pas //.

- En utilisant les variables générées automatiquement et rappelées en entête de fonction, on peut relier les connecteurs entre eux.

Exemple de fonction : résistance chauffante.

Quand la résistance est parcourue par un courant, les pertes par effet Joule sont évacuées sous forme de chaleur. Ce modèle est déjà présent dans la palette SIMM (Electrique/Composants basiques/Passif/MEAB_HeatingResistor).

Le paramétrage retenu est le suivant :



La relation électrique est U=R.i avec U la tension aux bornes de la résistance et la résistance évolue linéairement en fonction de la température T : R=R_ref*(1 + alpha*(T - T_ref)).

L'entête de la fonction générée est la suivante :

```
class ResistanceChauffante
  ////Generation automatique ////
      //parameters
      parameter Real R_ref = 1.000000e+02;
      parameter Real T_ref = 2.930000e+02;
      parameter Real alpha = 1.000000e-01;
      //input variables
      Modelica.Electrical.Analog.Interfaces.PositivePin p;
      //les variables disponibles sont p.v potentiel en V, p.i courant en A,
      //output variables
      Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heat;
      //les variables disponibles sont heat.T température en Kelvin, heat.Q_flow flux thermique en W,
      Modelica.Electrical.Analog.Interfaces.NegativePin n;
      //les variables disponibles sont n.v potentiel en V, n.i courant en A,
  ////Ne pas modifier avant cette ligne ////
```

On retrouve donc la température en utilisant la variable heat.T et la tension aux bornes de la résistance sera par convention obtenue par la relation U=p.v-n.v.

On introduit donc deux variables internes pour simplifier les écritures : U et R

Real U ;

Real R ;

On peut alors définir le corps de la fonction en faisant attention à bien terminer chaque ligne par un point-virgule :

U=p.v-n.v ;

R=R_ref(1+alpha*(heat.T-T_ref)) ;*

*U=R*p.i ;*

Le modèle ainsi défini n'est pas complet et si on lance une simulation avec ce modèle, le compilateur Modelica indiquera qu'il y a trop d'inconnues par rapport au nombre d'équations. En effet, il faut définir la manière dont transitent les flux à l'intérieur du bloc. Le flux électrique en entrée est égal au flux électrique en sortie. Le flux thermique est égal (au signe près par convention) à la puissance électrique perdue par effet Joule.

On ajoute donc les équations suivantes :
p.i=n.i ;
*heat.Q_flow=-U*p.i ;*

On a donc le modèle complet suivant

```
class ResistanceChauffante
  ////Generation automatique ////
      //parameters
      parameter Real R_ref = 1.000000e+02;
      parameter Real T_ref = 2.930000e+02;
      parameter Real alpha = 1.000000e-01;
      //input variables
      Modelica.Electrical.Analog.Interfaces.PositivePin p;
      //les variables disponibles sont p.v potentiel en V, p.i courant en A,
      //output variables
      Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heat;
      //les variables disponibles sont heat.T température en Kelvin, heat.Q_flow flux thermique en W,
      Modelica.Electrical.Analog.Interfaces.NegativePin n;
      //les variables disponibles sont n.v potentiel en V, n.i courant en A,
  ////Ne pas modifier avant cette ligne ////

 //    Vous pouvez definir après ce commentaire d'autres variables internes ou parametres
 //Exemple :  Real x(start=1), y(start=2); Parameter Real a=1;
Real U;
Real R;
equation
      // Renseigner ici votre fonction
      // Les noms des variables de flux et potentiel associées à chaque connecteur sont définies dans la documentation.
      // Exemple : der(flange.phi) pour la dérivée de la position angulaire
U=p.v-n.v;
R=R_ref*(1+alpha*(heat.T-T_ref));
U=R*p.i;
p.i=n.i;
heat.Q_flow=-U*p.i;

end ResistanceChauffante;
```

On teste le modèle ainsi défini en plaçant la résistance chauffante dans un circuit électrique et on relève la température de la pièce chauffée.

Remarque : lorsque le modèle n'est pas satisfaisant physiquement, le compilateur indique qu'il manque des équations ou des variables. Il faut alors bien réfléchir aux équations de potentiels et de flux à utiliser.

Pour simplifier également la résolution et limiter le temps de calcul, il est pertinent d'inverser des relations de manière à avoir des comportements explicites si possible.

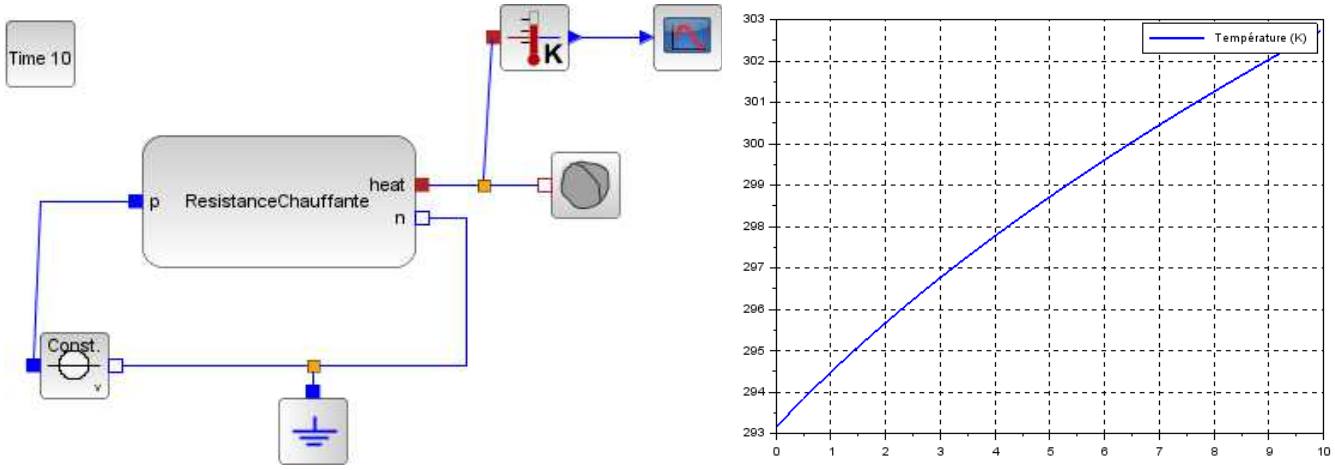


Figure 31: Modélisation et simulation du chauffage d'une pièce

On pourra noter que si une simulation ne converge pas, une raison souvent valable est que le modèle utilisé n'est pas physique et par conséquent le solveur a du mal à converger. Il faut donc bien réfléchir à définir un modèle physiquement valide.

b) Syntaxes supplémentaires

Il est possible d'utiliser différents mots clés Modelica pour définir des équations.

Instruction conditionnelle

Pour qu'une variable var prenne deux valeurs différentes v1 ou v2 en fonction d'une condition donnée, on utilise la syntaxe :

```
    var=if condition then v1 else v2 ;
ou  if condition then
        var=v1 ;
    else var=v2 ;
    end if
```

Plusieurs conditions if peuvent être imbriquées.

Les conditions logiques utilisent les mots clés *and, or,* <,<=,>,>=. On ne peut pas faire d'égalité ou de test de valeurs différentes.

Fonctions particulières

Pour accéder à la valeur du piquet de temps précédent d'une variable a, on utilise la fonction pre(a) ;

Le temps courant de simulation est accessible par le mot clé : time. Cette variable est incrémentée du pas de temps à chaque pas de calcul.

Pour réinitialiser cette valeur à un instant T0, on peut utiliser l'instruction

```
when time > T0 + period then
    reinit( T0, time );
end when;
```

Les fonctions mathématiques utilisées dans Scilab sont disponibles dans Modelica : sin, cos, tan, asin, acos, atan, exp, sqrt, puissance (symbole ^), log (pour ln),...

La dérivée d'une grandeur réelle est calculée numériquement par l'instruction : der(var) ;

Constantes, valeurs particulières

Des constants sont définies et accessibles dans une fonction depuis le module Coselica. On tape directement : Coselica.Constants.pi ; par exemple ou Coselica.Constants.eps ; pour exp(1), Coselica.Constants.small pour un nombre très petit non nul, Coselica.Constants.inf pour un nombre très grand (infini).

Pour définir une valeur initiale à une variable interne, on utilise l'instruction :

```
Real mavar(start=10) ;
```

Pour fixer une valeur à une variable (elle ne pourra pas être modifiée dans le programme), on utilise l'instruction :

```
Real mavar(start=10,fixed=true) ;
```

Un vecteur ou une matrice est défini par l'instruction suivante :

```
//Vecteur de dimension 3
Real vec[3] ;
```

```
//Matrice de dimension 2x2
Real mat[2,2] ;
```
On accède à une valeur par le nom de la variable et l'indice souhaité (qui commence à 1).
```
mavar=2*vec[1]+mat[1,2];
```

Pour faire une boucle finie, on utilise l'instruction :
```
for i in 1:10 loop
 // instruction
end for ;
```

D'autres éléments de syntaxes peuvent être utilisés (connect() ; pour relier des blocs en donnant leur nom complet par exemple). Des informations supplémentaires sur le langage Modelica peuvent être obtenues sur internet si besoin est.

Nous avons donc montré dans cet article les possibilités offertes par Scilab et SIMM pour simuler le comportement d'ensemble commande-motorisation et pour créer des modèles personnalisés assez simplement. De nouveaux blocs seront ajoutés au fur et à mesure dans le module SIMM. Il sera nécessaire de mettre en place une aide intégrée au module pour aider les utilisateurs.