MODÉLISATION MULTIPHYSIQUE

Scilab et les systèmes à (première partie)

ALAIN CAIGNOT, VINCENT CRESPEL ET DAVID VIOLEAU [1]

L'étude des modèles de comportement en STI2D et en CPGE va de concert avec l'utilisation de logiciels multiphysiques. Celui, gratuit, que nous vous présentons ici permet de construire rapidement des schémas à partir d'une bibliothèque et de faire l'analyse temporelle et fréquentielle des systèmes. Démonstration.

mots-clés
logiciel, simulation

Durant l'année scolaire 2011-2012, une formation de très grande ampleur a été organisée dans l'académie de Paris à destination de l'ensemble des professeurs de la filière STI2D. Pour la modélisation comportementale, la formation proposée s'est basée sur l'utilisation du logiciel Scilab (développé en France), de son extension Xcos et d'une « boîte à outils » (*toolbox*) spécifique, nommée CPGE parce qu'initialement développée pour les classes préparatoires aux grandes écoles, mais très facilement adaptable pour la filière STI2D.

Durant cette formation de longue durée (6 séances de 4 heures), des applications ont été proposées aux participants pour la simulation de processus continus, puis, dans un second temps, pour l'acquisition de données et le pilotage d'une imprimante du commerce (le modèle HP2050A de Hewlett-Packard) qu'ils avaient préparée et câblée. Tous les documents fournis pendant cette formation ainsi que les fichiers associés à cet article sont disponibles à l'adresse suivante :

https://sites.google.com/site/sti2dparis/

Notre but ici est de montrer les potentialités de la boîte à outil CPGE pour les STI2D quant à la génération de modèles à destination des élèves, c'est-à-dire évitant au maximum les développements mathématiques ou les descriptions trop complexes. Une seconde partie mettra en évidence les potentialités de Scilab pour la modélisation multiphysique via la boîte à outil Coselica (encore incomplète et en développement), dont le fonctionnement permet de réaliser des simulations du même type que celles pouvant être mises en œuvre sous Simscape (Matlab) ou MapleSim (Maple).

[1] Professeurs en classes préparatoires aux grandes écoles, formateurs STI2D pour l'académie de Paris.

La problématique de la simulation comportementale

Le but de la simulation est de prévoir le comportement d'un système, afin d'améliorer son comportement tout en limitant les essais coûteux, ces derniers restant cependant nécessaires pour valider ou recaler les modèles utilisés. Dans le cadre de la formation en sciences et techniques industrielles, le but de la modélisation comportementale sera d'analyser les écarts entre les résultats de la simulation et les mesures réellement réalisées sur le système 1.

Ainsi, dans le cadre de la (re)conception ou du réglage d'un système, on aura un dialogue constant entre les trois domaines, réalité, banc d'essai et modèles, la simulation servant plus particulièrement à valider ou invalider certains choix d'évolution ou à prédire les performances attendues.

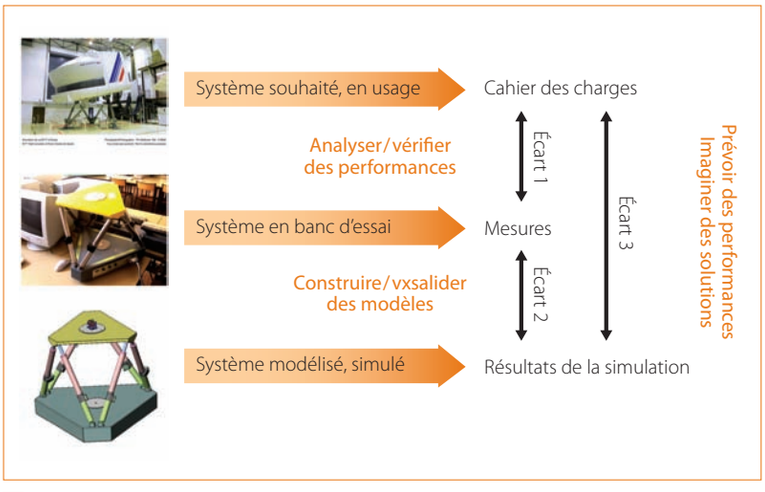
L'élaboration d'un modèle pertinent pour une phase de vie donnée d'un système est une étape très importante au niveau industriel, car elle conditionne la pertinence des résultats obtenus (point essentiel, difficile à apprécier), la facilité de choisir et de justifier le paramétrage adéquat par rapport aux objectifs visés, la possibilité de remettre en cause ce modèle, la qualité des échanges au sein de l'équipe projet…, et donc, *in fine*, l'innovation, la créativité, l'attractivité, etc., du produit fini.

La modélisation passe par plusieurs étapes dont seules certaines sont réellement accessibles aux élèves de la filière STI2D. En conséquence, nous nous limiterons ici à analyser les écarts entre les mesures réellement réalisées sur le système (lors d'un protocole maîtrisé) et les simulations obtenues par différents modèles, plus ou moins optimisés par la suite.

L'installation du logiciel et de la boîte à outils

Le logiciel Scilab (sous licence *open source* CeCILL, ce qui garantit le respect du droit français) est disponible en libre téléchargement pour les environnements Windows, Mac et Linux, en versions 32 et 64 bits. Dans le doute, il est fortement conseillé d'installer la version 32 bits, la différence dans la durée de compilation étant négligeable. Ce logiciel existe dans de très nombreuses langues ; à l'installation, la version française sera automatiquement installée.

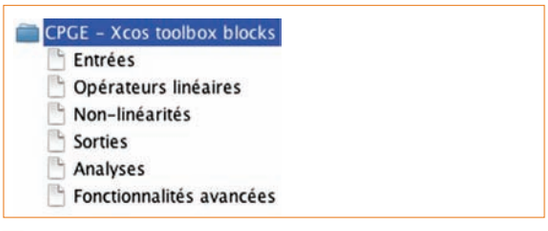
temps continu



1 L'analyse des écarts entre modèles et réel

2 La console Scilab

3 Le bandeau supérieur



4 Le contenu de la palette « CPGE – Xcos toolbox blocks »

Après l'installation, lancez le logiciel puis installez la boîte à outil CPGE par Applications/Gestionnaire de modules – ATOMS/Éducation/CPGE. (Pour information, la version à la date de rédaction de ces pages est la 1.4.1, mais la mise à jour est continue ; il est donc conseillé de mettre régulièrement à jour sa version.) À l'issue de l'installation, redémarrez le logiciel Scilab : la « console Scilab » (colonne du milieu) doit indiquer que la boîte à outil est correctement installée **2**.

Lorsque la console Scilab est lancée, démarrez l'extension Xcos en cliquant sur l'icône dédiée située dans le bandeau supérieur (lorsque le pointeur passe sur les icônes, des infobulles indiquent les fonctionnalités des différentes icônes) **3**. Pour lancer Xcos, on peut également tout simplement taper « xcos » (sans majuscules, le logiciel est sensible à la casse) dans la console, après l'indicateur « --> » **2**. Cette action a pour effet d'ouvrir le « navigateur de palettes », qui contient les différentes icônes et la feuille de travail, initialement nommée « Sans titre – hh:mm:ss », où « hh:mm:ss » correspond à l'heure de création du modèle. Dans le navigateur de palettes se trouvent toutes les icônes nécessaires à la simulation de processus, et on retrouve dans la partie inférieure la palette nommée « CPGE – Xcos toolbox blocks » qui contient des blocs optimisés pour la simulation en CPGE et en STI2D **4**.

Quelques remarques sur la transformée de Laplace

Dans le cadre spécifique de la filière STI2D, un des points les plus souvent évoqués avec les participants à la formation est l'insuffisante connaissance des notions mathématiques pour la compréhension des problématiques. En effet, si la modélisation multiphysique s'affranchit de cette difficulté (mais pour en ajouter d'autres, comme on le verra dans la seconde partie), la modélisation à temps continu demande de comprendre la structure d'un outil symbolique très compliqué théoriquement, mais au final très simple d'utilisation : la transformée de Laplace.

Cet outil, dont absolument aucun développement n'est utile dans le cadre de la filière STI2D, permet de transformer les équations différentielles représentant les évolutions de processus à temps continu en fonctions polynomiales d'une variable, notée s dans le cas de Scilab (ce qui est également le cas avec les logiciels Matlab/Simulink et Maple/MapleSim).

Après quelques développements mathématiques ne devant surtout pas être abordés en STI2D, on peut ainsi montrer que, si la transformée dans le domaine de Laplace d'une fonction $x(t)$ est $L[x(t)] = X(s)$, la transformée de sa dérivée sera $L[x'(t)] = s\,X(s)$, celle

5 L'angiographe biplan

de sa dérivée seconde sera $L[x''(t)] = s^2 X(s)$, etc., si l'on considère des conditions initiales nulles. De même, la transformée de la primitive sera $L[\int_0^t x(u)du] = X(s)/s$, et ainsi de suite.

On voit donc que, moyennant de grandes contraintes mathématiques heureusement hors de propos pour les fonctions rencontrées physiquement, le fait de dériver dans le domaine temporel revient à multiplier la fonction transformée par la variable s, et qu'intégrer dans le domaine temporel revient à diviser la fonction transformée par la variable s.

Comme l'utilisation d'une telle variable symbolique peut surprendre les élèves, voire bloquer leur réflexion, nous verrons qu'il est tout à fait possible de remplacer les différents blocs comportant des équations mathématiques par des images, l'étudiant associant alors des blocs graphiques pour mettre en place très rapidement un modèle du système qu'il étudie.

L'exemple proposé

L'étude se fera sur le déplacement en translation de l'armature suspendue au plafond d'une chaîne image d'un angiographe biplan 5 permettant la création d'images tridimensionnelles de la structure veineuse d'un patient afin de prévoir les risques d'anévrisme (éclatement d'un vaisseau sanguin, qui aboutit très souvent à une perte partielle ou totale des capacités motrices, voire un décès). La prise de vue est réalisée en positionnant longitudinalement et angulairement une tête et un récepteur à rayons X, un logiciel dédié se chargeant de la création d'une image colorée à destination du médecin.

Le mouvement de translation indiqué en 5 est commandé par le médecin à l'aide d'un joystick. La position voulue étant atteinte, le médecin lâche le joystick et déclenche la prise de vue à l'aide d'une pédale de commande. Le médecin exige que la prise de vue puisse commencer dès la demande d'arrêt. Le tableau 6 propose un extrait très réduit du cahier des charges, auquel nous allons nous consacrer.

Une rapide présentation du système, les données numériques des différents éléments techniques (moteur, système mécanique, etc.) ainsi que l'ensemble des modèles Scilab/Xcos sont disponibles sur le site évoqué plus haut. Pour information, l'étude proposée a fait l'objet d'un sujet de concours (Centrale-Supélec filière MP, session 2009) ainsi que d'un TP de formation – à destination des professeurs de CPGE – sur le stand Démosciences du salon Éducatec 2012.

	Fonction de service	Critère	Niveau
FS1	Déplacer le LP en translation	Être rapide sans mettre en danger le personnel médical	Vitesse : 100 mm·s^{-1} Pas de dépassement sur un échelon de vitesse Écart < 10 % sur une rampe de vitesse

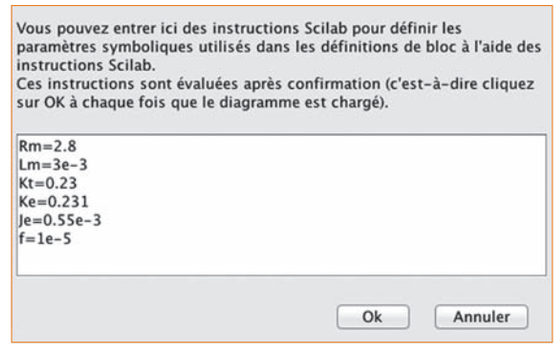
6 Le cahier des charges de la fonction étudiée

Équation électrique	Équation mécanique
$u_m(t) = R_m i(t) + L_m [di(t) / dt] + e(t)$	$J_e [d\omega_m(t) / dt] = c_m(t) + c_r(t) - f\omega_m(t)$
Équation de couplage	**Équation de couplage**
$c_m(t) = K_t i(t)$	$e(t) = K_e \omega_m(t)$

7 Les équations du modèle linéarisé du moteur à courant continu

Équation électrique	Équation mécanique
$U_m(s) = (R_m + L_m s)\, I(s) + E(s)$	$(J_e s + f)\boldsymbol{\Omega}_m(s) = C_m(s) + C_r(s)$
Équation de couplage	**Équation de couplage**
$C_m(s) = K_t I(s)$	$E(s) = K_e \boldsymbol{\Omega}_m(s)$

8 Les équations du modèle dans le domaine de Laplace



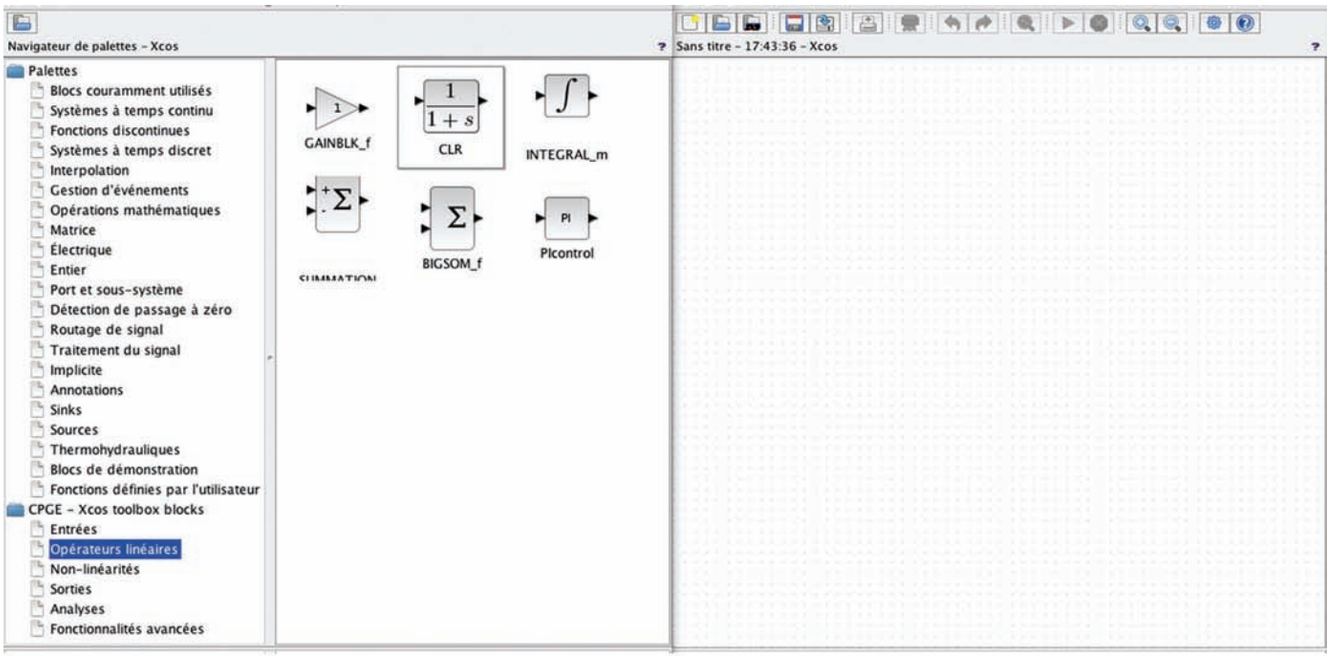
9 Les données du modèle de simulation

Un premier schéma-bloc : le moteur à courant continu

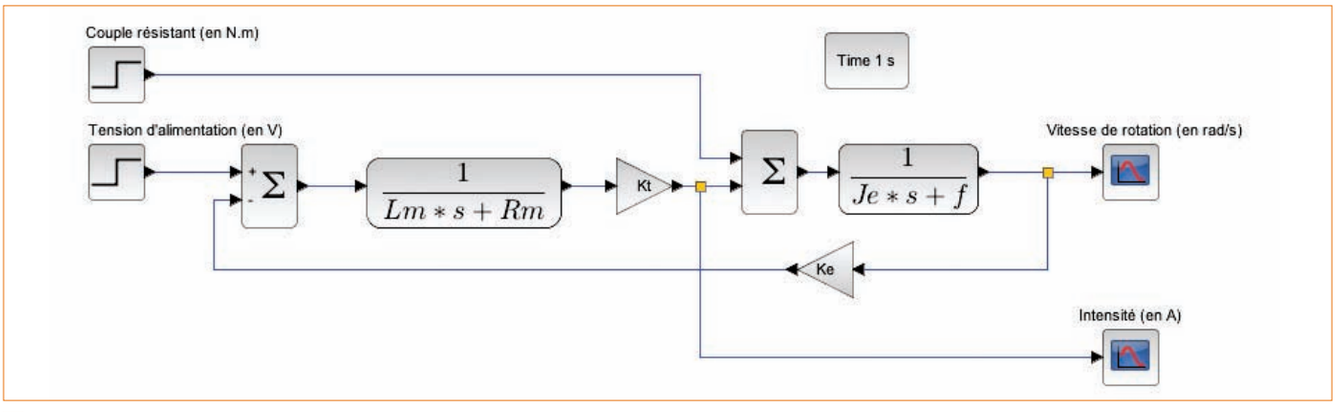
Afin de se familiariser avec le logiciel, on commence par mettre en place le schéma-bloc du moteur à courant continu, défini, au niveau du modèle linéarisé classique, par sa résistance R_m (en Ω), son inductance L_m (en H), sa constante de couple K_t (en Nm/A), sa constante de vitesse K_e (en V·s/rad), le frottement visqueux f (en Nm·s/rad) et l'inertie équivalente rapportée à l'axe moteur J_e (en kg·m^2). On rappelle en 7 les quatre équations de ce modèle linéarisé.

Ces équations se transforment dans le domaine de Laplace en fonctions de la variable symbolique s 8 (traditionnellement, on note en minuscules les fonctions du temps et en majuscules celles de la variable symbolique de Laplace, mais ce n'est pas obligatoire).

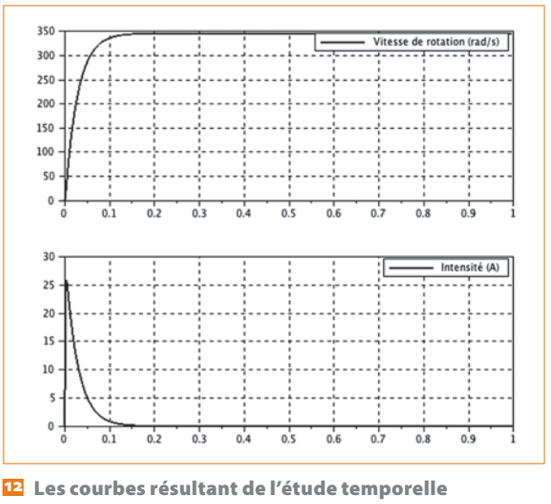
Pour créer le modèle dans l'environnement Scilab/Xcos, il est possible d'entrer directement les valeurs



10 **Le navigateur de palettes et la fenêtre de travail**



11 **Le schéma-bloc du moteur**

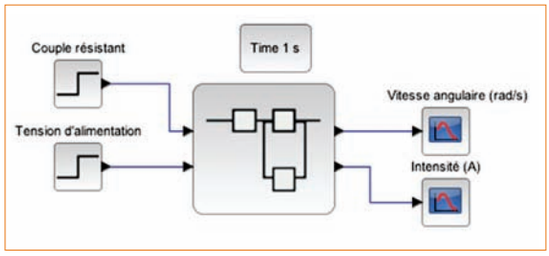


12 **Les courbes résultant de l'étude temporelle**

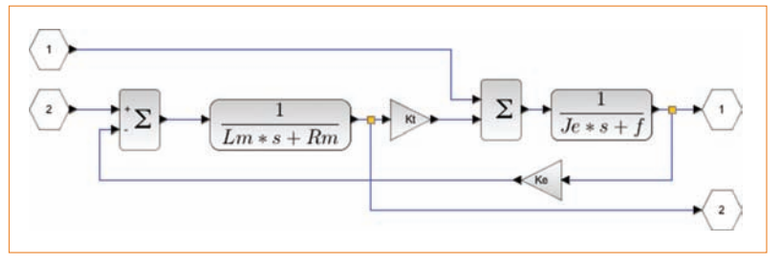
numériques dans les différents blocs graphiques ou bien d'utiliser les notations littérales : dans ce dernier cas, et contrairement à quelques autres logiciels, Scilab/Xcos demande que les variables soient définies numériquement *avant* toute mise en place d'un modèle, ce qui se fait dans le contexte de la simulation, accessible par un clic droit dans la fenêtre, puis « Modifier le contexte » (ou « Simulation » > « Modifier le contexte »). Pour mettre en place la simulation, on commence donc par rentrer les données comme sur l'écran 9.

Il est alors possible de créer le fichier directement en sélectionnant tout d'abord l'icône souhaitée dans le navigateur de palettes (un carré apparaît alors autour du symbole correspondant), puis en la glissant-déposant dans la fenêtre de travail (située à droite sur l'écran 10). Dans le cadre de l'étude proposée ici, utilisez *exclusivement* les icônes du module CPGE situé en bas du navigateur de palettes et nommé « CPGE – Xcos toolbox blocks ».

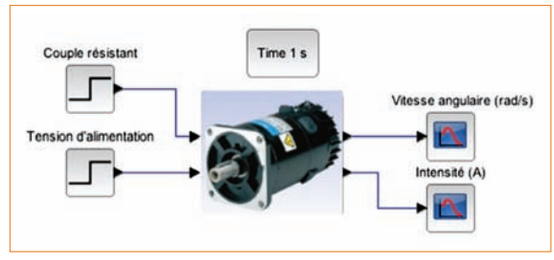
Pour créer le schéma-bloc du moteur présenté en 11, il suffit d'insérer deux blocs STEP_FUNCTION (disponibles dans le répertoire « Entrées » et correspondant au signal d'entrée le plus simple pour une première étude), deux blocs CLR (fonction de transfert continue, à définir en fonction de la variable symbolique de Laplace notée *s* dans le logiciel), deux blocs GAINBLK_f, deux blocs SUMMATION (disponibles dans le répertoire « Opérateurs linéaires ») et enfin deux blocs SCOPE (disponibles



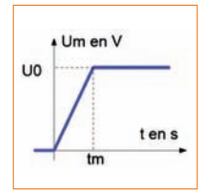
13 Le schéma-bloc simplifié



14 Le « superbloc » du moteur



15 Le schéma-bloc intégrant l'image du moteur Sanyo

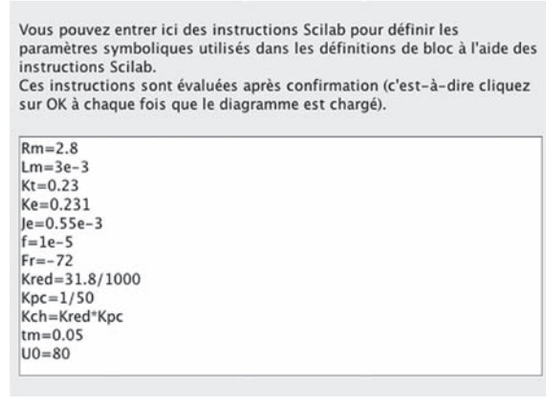


16 L'alimentation progressive du moteur

dans le répertoire « Sorties » et qui vont permettre de visionner l'évolution de la vitesse de rotation du moteur et celle de l'intensité d'alimentation). Afin d'améliorer la structure, il est souvent nécessaire de changer l'orientation d'un bloc. Pour cela, après l'avoir déposé sur la fenêtre, sélectionnez le bloc, puis, après un clic droit, « Format » et « Pivoter » ou « Format » et « Miroir ». Ce changement peut également être réalisé au clavier avec la touche de commande (Ctrl sous Windows ou Linux et cmd – ou pomme – sous Mac OS) puis M ou R. Pour changer le signe des sommateurs, cliquez sur le bloc et choisissez la forme du vecteur (« [1, – 1] » ou « 1 – 1 » sans crochet ni virgule pour un soustracteur) ; il est possible de sommer ou soustraire plus de deux valeurs en augmentant la taille du vecteur (exemple : « 1 1 1 » pour une somme à trois entrées).

Pour définir les grandeurs de sortie (afin que les courbes obtenues soient ensuite repérables facilement), il suffit de double-cliquer sur chacun des deux blocs SCOPE et de renseigner le nombre de courbes (une seule dans notre cas) ainsi que le nom du signal, à savoir une vitesse de rotation de l'axe du moteur (en rad/s) et l'intensité dans l'induit (en A). Par ailleurs, en double-cliquant n'importe où dans la fenêtre (sauf sur une icône), il est également possible d'insérer du texte permettant d'améliorer la lecture du schéma-bloc : sur la figure 11, on a ainsi pu commenter à quoi correspondaient les deux échelons d'entrée et les deux scopes de sorties.

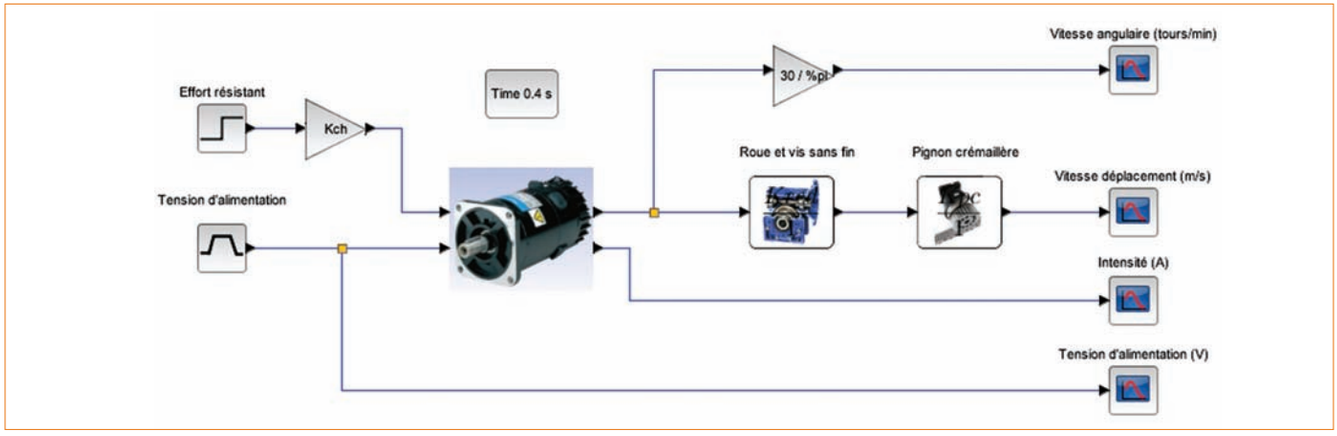
Pour lancer une simulation, il est nécessaire de spécifier le type d'étude retenu (temporel et/ou fréquentiel). Ici, nous allons réaliser une étude temporelle et, pour cela, glisser un bloc REP_TEMP (dans le répertoire « Analyse » de la palette), puis, en double-cliquant dessus, définir la durée de la simulation et le nombre de points.



17 L'ajout des variables dans le contexte

En prenant une tension d'entrée de 80 V à l'instant $t = 0$, un couple perturbateur nul (dont l'influence sera étudiée plus loin), une durée de 1 seconde et 500 points, on obtient les deux courbes 12.

Ce schéma-bloc étant réalisé, on va le modifier pour le rendre plus facilement accessible à un élève en prébac, avec lequel la notion de variable symbolique ne peut ni

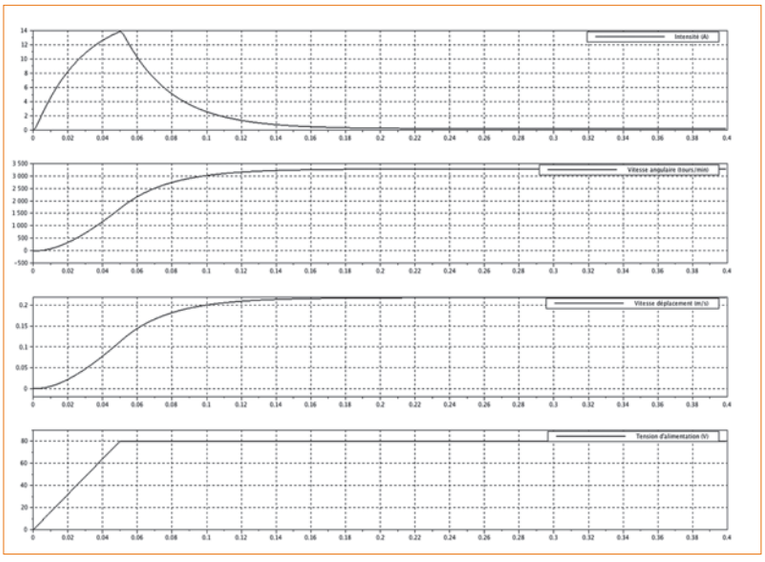


18 Le schéma-bloc de la commande en boucle ouverte

ne doit être abordée : on commence par transformer le système en un « superbloc » en sélectionnant toute la zone correspondant au moteur, puis on clique droit et choisit « Zone vers super-bloc », ce qui donne, après une légère réorganisation des connexions, la forme du schéma-bloc 13. En double-cliquant sur le superbloc, on retrouve bien entendu les différents blocs installés précédemment, avec des indicateurs d'entrée et de sortie 14.

On choisit alors de « recouvrir » le superbloc par une photo ou un dessin afin de masquer la structure mathématique du bloc et de le rendre facile à manipuler par un élève. Pour cela, après avoir cliqué droit sur le superbloc, on choisit « Format », puis « Édition… ». Dans l'onglet Couleur de remplissage (agrandir la fenêtre pour le faire apparaître), on peut choisir une image, de préférence dans le même répertoire que le fichier, par exemple celle du moteur Sanyo réellement implanté sur ce système d'angiographie 15.

On pourrait bien entendu procéder ainsi pour de nombreux composants classiques (moteurs, hacheurs, réducteurs, capteurs, etc.) et demander ensuite aux élèves d'associer ces composants en se basant sur la structuration en chaîne fonctionnelles.



19 **Les courbes résultant de la simulation en boucle ouverte**

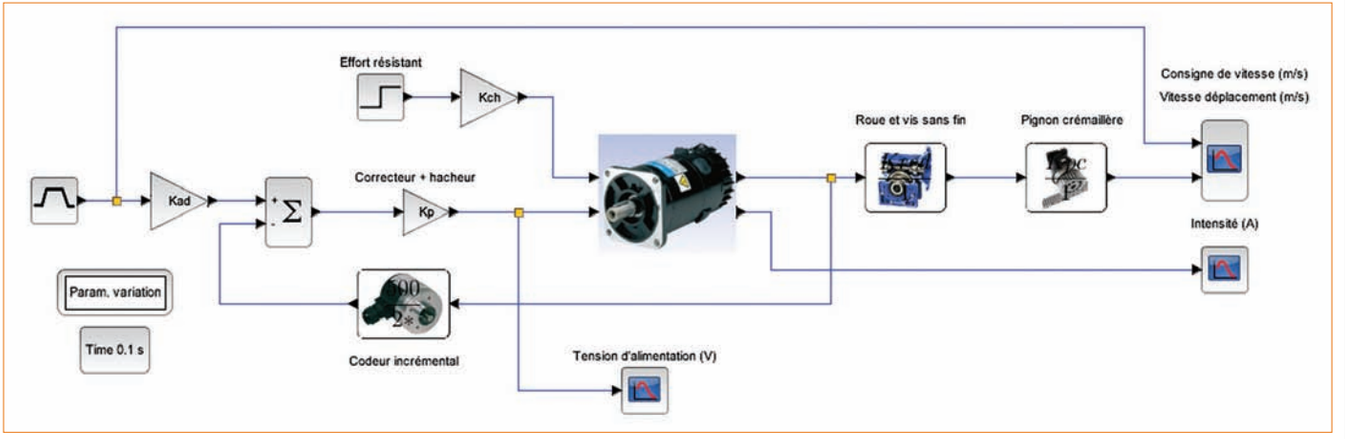
Un deuxième schéma-bloc : la commande en boucle ouverte

En aval du moteur à courant continu étudié précédemment se trouve une chaîne cinématique composée par la mise en série d'un réducteur à engrenages et d'un système roue et vis sans fin (gain $K_{red} = \omega_p / \omega_m = 31{,}8 \times 10^{-3}$) puis d'un système pignon-crémaillère (gain $K_{pc} = v / \omega_p$ = 1/50 m/rad) permettant de transformer la rotation en une translation de l'axe linéaire. Afin de toujours avoir la forme la plus compréhensible pour les élèves, on peut créer des blocs et les recouvrir de photos plus parlantes que des expressions mathématiques.

Afin de ne pas trop solliciter le moteur électrique, on l'alimente de manière progressive jusqu'à la valeur nominale selon la courbe 16 : dans le cadre du système étudié, on donne t_m = 50 ms (temps de montée) et U0 = 80 V (tension nominale). Afin de créer simplement ce signal, la boîte à outils CPGE offre la possibilité de définir un signal trapèze dont nous n'exploiterons que la partie initiale (montée et maintien) en prenant un temps de maintien en valeur maximale très grand.

Par ailleurs, un effort de $F_r = -72$ N est exercé sur la structure qui se translate à l'instant t_r = 100 ms ; cet effort est ressenti comme un couple résistant au niveau du moteur avec, si on suppose un rendement unitaire, un rapport de proportionnalité correspondant au gain de la chaîne cinématique. On complète alors le contexte (accessible par un clic droit) avec les variables Fr = – 72, Kred = 31.8/1000, Kpc = 1/50, Kch = Kred*Kpc (gain de la chaîne cinématique), tm = 0.05 et U0 = 80 17.

On ajoute trois gains (icônes GAINBLK_f dans le répertoire « Opérateurs linéaires ») et deux afficheurs (icônes SCOPE dans le répertoire « Sorties ») pour le tracé de la tension d'alimentation et de la vitesse de déplacement. On efface alors l'échelon de tension, puis on ajoute un bloc TRAPEZOID (répertoire « Entrées »)

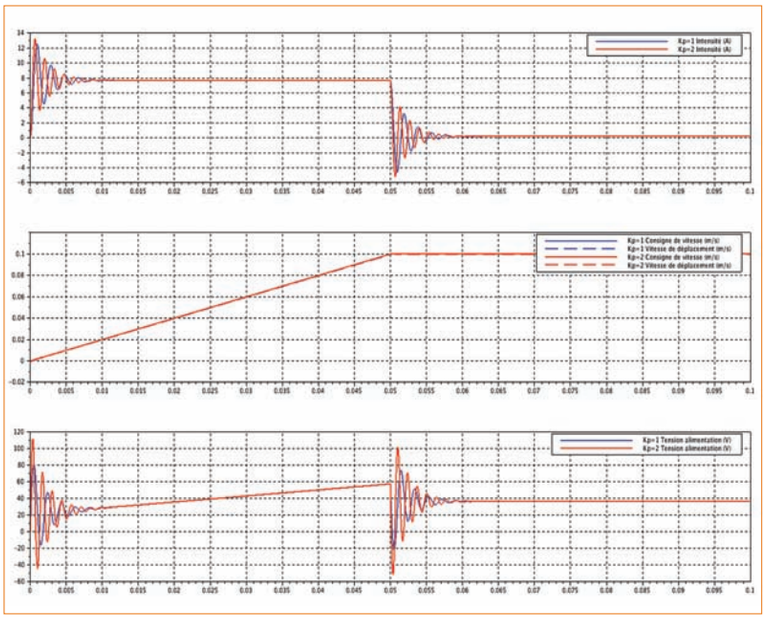


20 **Le schéma-bloc en boucle fermée (asservie)**

et on renseigne l'amplitude (U0), le temps de montée, noté « *rising time* » (t_m), et le temps de maintien, noté « *width* » (par exemple 10, donc très supérieur au temps de simulation) ; on ne pas touche pas aux autres paramètres.

À partir du schéma-bloc initial, on peut alors créer le schéma-bloc 18, correspondant à la commande en boucle ouverte (donc sans retour d'information) du système, d'entrée la tension du moteur en V, de sortie la vitesse de déplacement de l'axe en mètres par seconde, et de perturbation la force F_r (également à $t = 0$). Par ailleurs, on place un gain de $30/\pi$ (noté « %pi » dans Scilab) qui permet d'obtenir la vitesse de rotation en tours par minute, grandeur plus facile à appréhender que le radian par seconde.

On lance la simulation, ce qui a pour effet d'ouvrir une fenêtre avec quatre courbes : la tension d'alimentation en V, l'intensité dans l'induit en A, la vitesse angulaire en tr/min et la vitesse de déplacement en m/s 19.



21 **Les fluctuations de tension en correction proportionnelle**

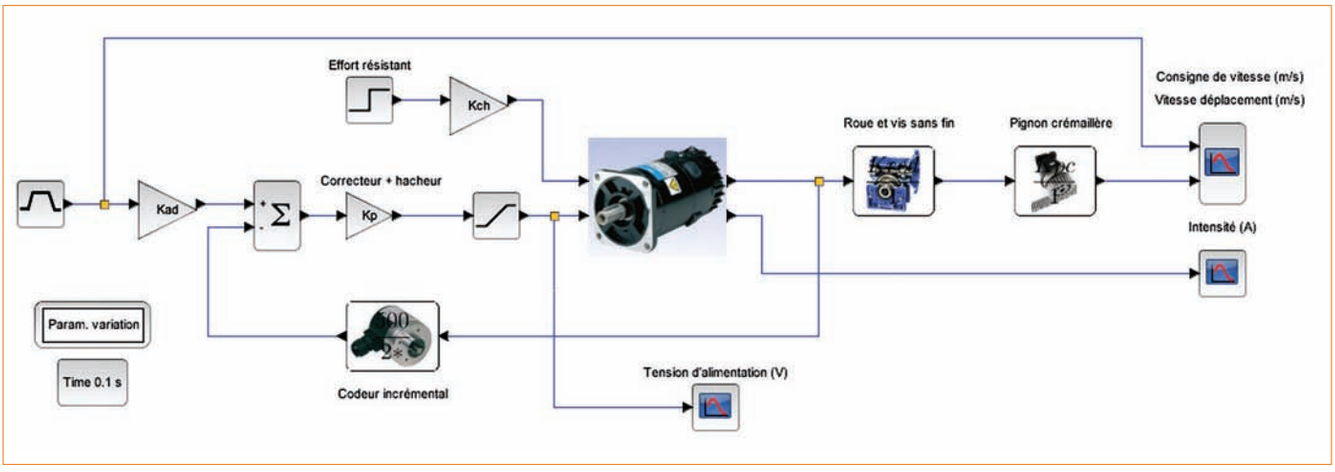
La structure de l'asservissement en vitesse

La structure de commande précédente n'est pas suffisamment performante, car elle est très sensible à la perturbation (pour s'en convaincre, il suffit d'augmenter la valeur de F_r), et que, afin d'arrêter le déplacement de l'axe au niveau souhaité, il serait nécessaire de couper l'alimentation au bon moment (donc en tenant compte à la fois des perturbations et de l'inertie du système). C'est bien entendu illusoire, d'autant que la valeur de l'effort résistant est une valeur moyenne estimée.

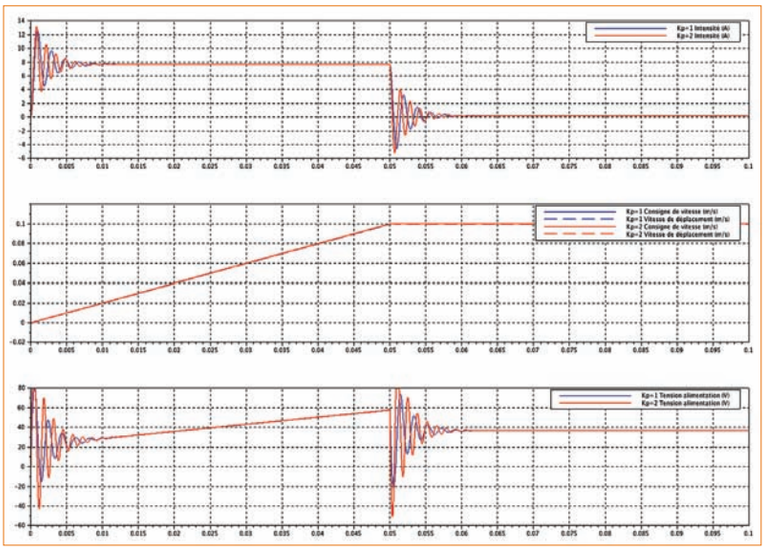
Pour maîtriser la vitesse de déplacement de l'axe et compenser les perturbations, il est donc nécessaire de passer à une structure asservie ; dans le cas du système étudié, cela est réalisé par l'implantation d'un codeur incrémental à 500 points par tour implanté sur l'axe moteur et l'adjonction d'un calculateur permettant de traiter les informations de manière numérique. On propose donc une nouvelle structure du schéma-bloc, proposée en 20. Le fichier correspondant à cette nouvelle structure est nommé « ComBFVitKp.zcos » sur le site cité précédemment.

Dans ce schéma-bloc, on note les évolutions suivantes :

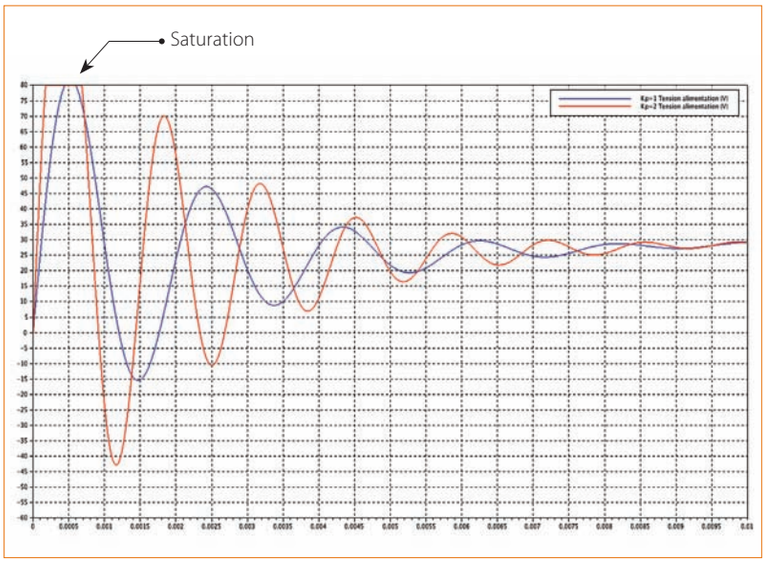
- Modification de la typologie de l'entrée pour réaliser cette fois une commande en trapèze de vitesse (valeur de maintien de 0,1 m/s au bout de 0,05 s), ce qui correspond à une commande classique de ce type d'axes linéaires asservis (on a cette même structure sur l'imprimante HP2050 étudiée en STI2D dans l'académie de Paris) ;
- Retour d'information par le codeur incrémental et ajout d'un module d'adaptation de gain K_{ad} afin d'avoir un écart nul en sortie du comparateur quand la sortie et l'entrée sont égales ;
- Mise en place d'un soustracteur et d'un gain pur K_p représentant l'ensemble {correcteur + module d'amplification (hacheur)} en amont de l'ensemble constitué du moteur et de la chaîne cinématique ;
- Ajout d'un bloc PARAM_VAR disponible dans le répertoire « Analyse ». En double-cliquant dessus, on peut définir la plage de variation d'un certain nombre



22 **Le schéma-bloc incluant une saturation**



23 Les fluctuations de tension limitées par la saturation



24 Zoom sur l'évolution de la tension d'alimentation

de paramètres afin d'analyser leur influence relative sur les réponses temporelles ou fréquentielles. Dans le cas étudié, seul le gain K_p sera modifié.

L'asservissement de vitesse et la correction proportionnelle

Nous allons, dans un premier temps, analyser l'influence de la simple correction proportionnelle implantée sur le comportement temporel du système. Pour information, dans ce contexte, la valeur de K_p est unitaire (la valeur n'a pas d'importance, mais, pour être prise en compte, elle doit être non nulle).

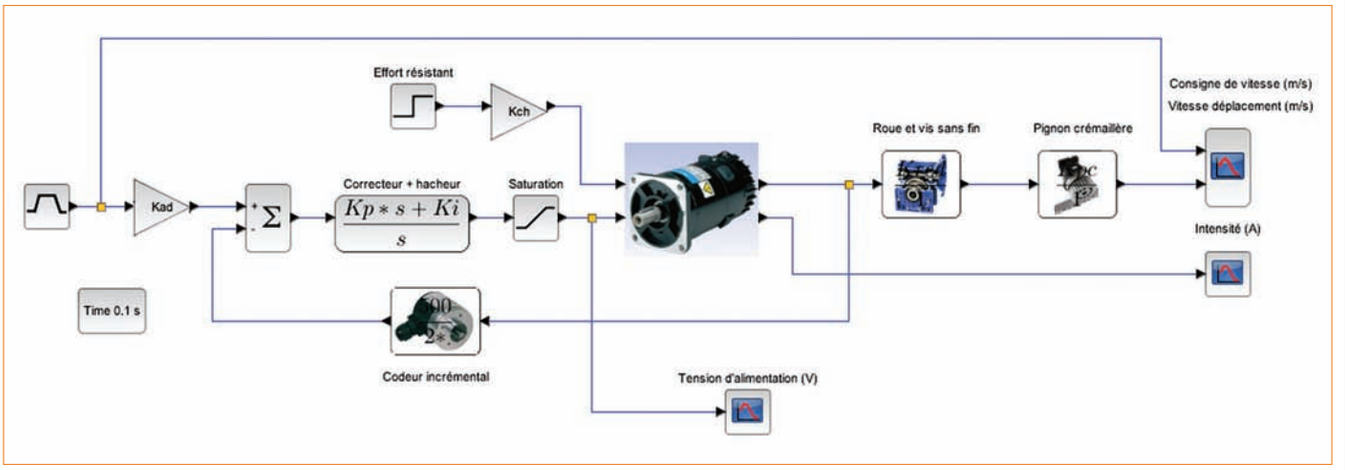
En double-cliquant sur le bloc PARAM_VAR, on vérifie que le nom K_p est entré comme premier paramètre, et on donne la plage de valeurs de K_p sous la forme d'un vecteur (par exemple [1 2 pour les deux valeurs de gain $K_p = 1$ et $K_p = 2$). En lançant la simulation, on obtient les courbes 21, où l'on constate qu'il y a des fluctuations de tension assez conséquentes.

Afin de protéger le moteur, on choisit de limiter la tension d'alimentation à ± 80 V, correspondant à sa tension nominale ; on constate que cette valeur est largement dépassée, même pour de faibles valeurs du gain K_p. Pour prendre en compte cette limitation dans le modèle Scilab/Xcos et observer son influence sur le comportement transitoire, il est indispensable d'ajouter une non-linéarité physique : la saturation, bloc disponible dans le répertoire « Non linéarités » du module CPGE. Après réorganisation des différents blocs et mise en place de la saturation, on obtient la structure 22.

En réglant cette saturation à ± 80, on obtient les courbes 23, mettant clairement en évidence le « blocage » de la tension d'alimentation à la valeur de ± 80 V. La figure 24 effectue un zoom sur l'évolution de la tension d'alimentation.

L'asservissement de vitesse et la correction proportionnelle et intégrale

Même si le suivi de la consigne de vitesse est relativement correct, les rapides fluctuations d'alimentation et



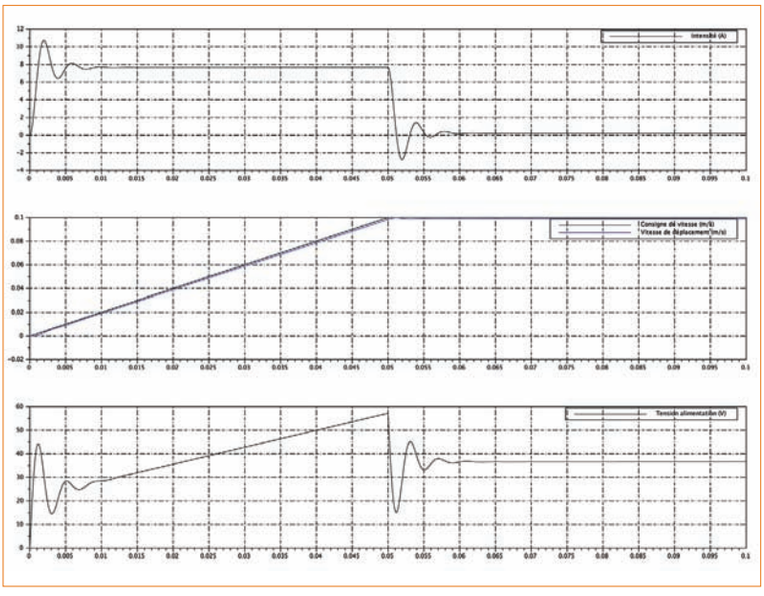
25 Le schéma-bloc incluant un correcteur PI

d'intensité dans l'induit peuvent poser problème dans ce système médical qui doit rester très silencieux. On peut les limiter grandement en diminuant le gain du correcteur, mais la qualité du suivi est alors déplorable (écart important lors de la phase d'accélération et en vitesse stabilisée, etc. – tout ceci étant bien entendu vérifiable directement via la simulation Scilab/Xcos en modifiant la valeur de K_p).

On peut améliorer ce suivi en apportant un effet intégral à la correction ; on choisit alors d'implanter un correcteur PI de fonction de transfert $C(s) = K_p + (K_i / s)$. On obtient alors la structure de la figure 25 (fichier « ComBFVitPISat.zcos » sur le site).

Le réglage de la correction PI se fait à partir d'une étude fréquentielle qui n'est pas du tout utilisable en STI2D. On peut cependant observer l'influence des deux gains K_p et K_i même si, en STI2D, cette étude n'est pas forcément pertinente.

Après un réglage rapide, on prend $K_p = 0{,}25$ et $K_i = 1$, on lance la simulation et on observe que ce réglage PI, pourtant très classique, n'est pas du tout idéal dans ce cas d'étude, car le moteur utilisé peine à entraîner de manière efficace le système avec cette structure série. En effet, la correction intégrale, si elle apporte de la précision, ralentit sensiblement le système, et il apparaît donc une erreur dans le suivi de la rampe, point qui peut être réellement problématique selon les cas 26.



26 Les courbes résultant de la correction PI

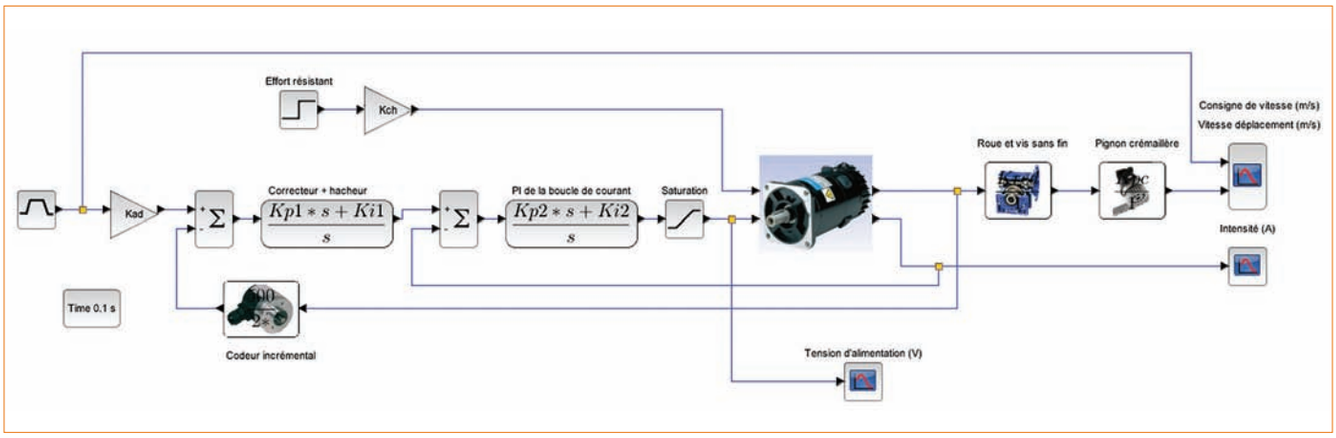
L'implantation d'une correction à boucle de courant

Afin d'améliorer sensiblement les performances, on va commander le moteur avec l'ajout d'une boucle de courant et, grâce à une structure adéquate, d'une mesure de l'intensité au niveau de l'induit qui va permettre de modifier en continu l'alimentation réelle du moteur après une correction le plus souvent de type PI, comme on le voit sur le schéma-bloc 27 (fichier « ComBFVitBC.zcos »).

Dans ce schéma-bloc, le retour de courant a été choisi unitaire, car le gain du capteur (résistance de shunt ou capteur à effet Hall par exemple) a été intégré dans le correcteur PI.

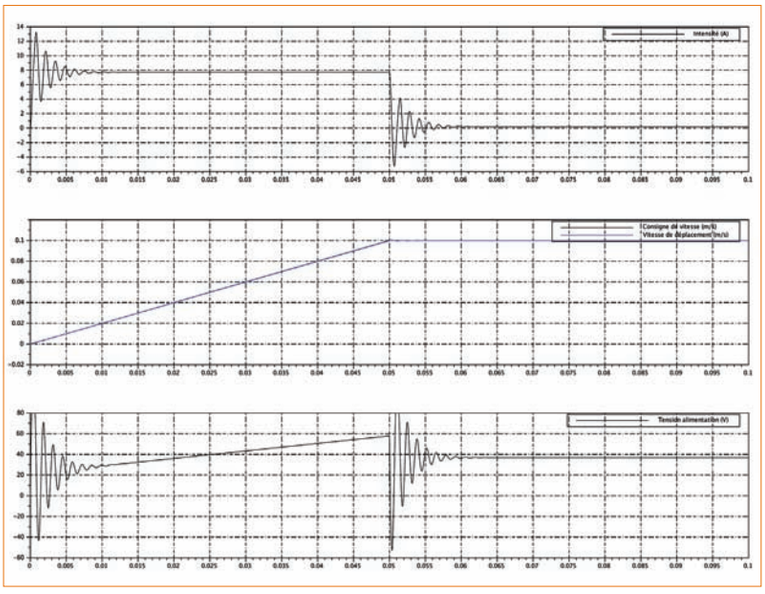
Cette structure de commande est très classique de nos jours ; on la retrouve par exemple sur l'imprimante HP2050 développée en 2011 pour Démosciences puis utilisée pour la formation STI2D dans l'académie de Paris. D'après plusieurs sources (mais nous n'avons pas procédé à une étude exhaustive), il semblerait que la quasi-totalité des cartes de commande actuelles dispose de cette fonctionnalité, qu'il est alors possible d'utiliser si besoin, ce qui est le cas ici.

Cette structure étant créée, on peut travailler sur l'impact d'une boucle de courant sur les performances globales du système en réglant les gains K_p2 et K_i2 du PID de la boucle de courant sans toucher aux gains K_p1 et K_i1 du PID initial (donc en amont de la boucle de courant). Là encore, le réglage complet est très long et devrait idéalement commencer par le réglage de la boucle de courant, mais on peut cependant analyser l'impact d'une telle boucle sur la qualité du suivi.

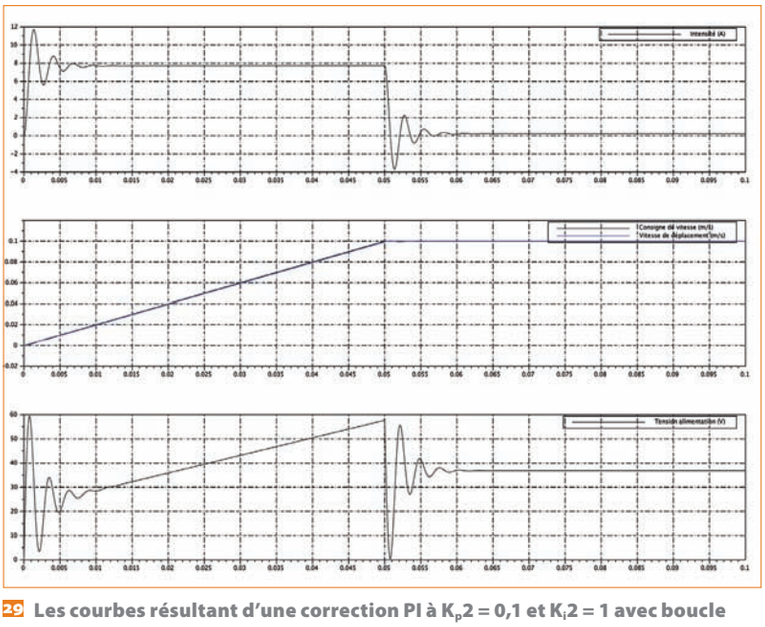


27 Le schéma-bloc incluant une correction à boucle de courant

Par exemple, pour un gain $K_p1 = 5$ et $K_i1 = 2$ dans le correcteur PI et sans boucle de courant, on obtient la courbe 28 : on a un excellent suivi de la consigne en rampe, mais de très rapides et importantes fluctuations dans la tension et l'intensité. Avec ces mêmes gains dans le correcteur PI et une boucle de courant réglée avec $K_p2 = 0,1$ et $K_i2 = 1$, on trouve les courbes 29 : toujours un excellent suivi et un très bon respect du cahier des charges fourni en 6 (un zoom autour de l'instant $t = 0,05$ s montrerait que l'écart entre la rampe de consigne et la vitesse réelle est très inférieur aux exigences du cahier des charges et qu'il n'y a pas de dépassement), le tout avec des fluctuations bien moindres au niveau de l'alimentation.



28 **Les courbes résultant d'une correction PI à $K_p1 = 5$ et $K_i1 = 2$ et sans boucle de courant**



29 **Les courbes résultant d'une correction PI à $K_p2 = 0,1$ et $K_i2 = 1$ avec boucle de courant**

Au vu de ces résultats, on voit qu'il est possible, en utilisant cette structure, d'atteindre de bonnes performances en termes de qualité du suivi en sollicitant beaucoup moins le moteur, le tout en ne modifiant ni la structure globale de l'asservissement (donc en gardant un codeur incrémental sur l'axe du moteur) ni même le choix du moteur à courant continu, dont la remise en cause est toujours assez délicate pour des raisons de coût, d'implantation, etc.

L'optimisation complète des deux correcteurs PI est cependant relativement complexe sur une structure à temps continu, mais est assez simple sur une structure à commande numérique, ce qui est le cas par exemple de l'imprimante HP2050. Sur ce matériel, l'implantation d'une telle structure à boucle de courant et correction PI permet d'atteindre des performances intéressantes sans pour autant solliciter trop fortement l'actionneur. En shuntant la carte de commande et en la remplaçant par une carte externe (une carte à base d'un microcontrôleur PIC + un hacheur dédié), on a pu observer un véritable gain quand on implante la boucle de courant.

Une autre boîte à outils

Cet article a montré quelques possibilités d'utilisation du logiciel libre Scilab, de son extension Xcos et de la boîte à outil CPGE pour la simulation de processus à temps continu. Moyennant quelques préparations de la part du professeur (blocs masqués par des images), il est possible de mettre en œuvre des modèles relativement simples à comprendre pour les élèves de STI2D, et tous les matériels du laboratoire peuvent être ainsi modélisés. La boîte à outil CPGE est en perpétuel développement. À présent très stable et efficace en calcul, elle peut être adaptée pour les STI2D avec une encore plus grande simplification de l'interface élève.

Par ailleurs, de par sa possibilité d'étude fréquentielle (non présentée ici), elle peut également être très intéressante pour chercher des paramètres de réglage cohérents ; cette fonctionnalité d'une grande complexité théorique ne doit cependant pas être présentée aux élèves de STI2D.

Ce n'est pas le seul axe d'étude : une autre boîte à outil, Coselica, permet la modélisation de processus multiphysiques ; elle sera présentée dans la seconde partie. Bien que sans aucun doute moins performants que d'autres solutions disponibles (Matlab + Simulink + Simscape, ou Maple + Maplesim), ces deux outils permettent de réaliser des modèles relativement faciles à comprendre par les élèves, qui peuvent, et c'est le plus intéressant de notre point de vue, les développer sur leur propre ordinateur, le logiciel pouvant être téléchargé librement. ■