Sciences de l'Ingénieur
TP n°1 - Introduction à Matlab Simulink
Saisie d'une fonction de transfert et d'un schéma-bloc

Les 8 séances de Travaux Pratiques que vous allez avoir ont pour but de vous apprendre à saisir, simuler et corriger un modèle d'un SLCI à l'aide de Matlab, et en particulier de Matlab Simulink. Le programme des 8 séances est le suivant :

TP 1	Introduction à Matlab Simulink, saisie d'une fonction de transfert et d'un schéma-bloc
TP 2	Réponse temporelle des SLCI
TP 3	Réglage des correcteurs à partir de la réponse temporelle (partie 1)
TP 4	Réglage des correcteurs à partir de la réponse temporelle (partie 2)
TP 5	Introduction à Sisotool, réponse fréquentielle des SLCI
TP 6	Réglage des correcteurs à partir de la réponse fréquentielle
TP 7-8	Exercice de synthèse

1 Introduction à Matlab Simulink

Commencez par lancer Matlab **R2014b** (Simulink ne fonctionne pas sur Matlab R2015b, donc si Matlab R2014b n'est pas présent sur votre poste de travail, changez d'ordinateur), puis lancez Simulink en cliquant sur l'onglet « Home » puis sur l'onglet « New » , puis en sélectionnant « Simulink Model » (voir Figure 1). Une fenêtre vierge s'ouvre alors : c'est dans cette fenêtre que vous allez pouvoir ajouter différents composants afin de saisir les modèles des systèmes étudiés.

Ouvrez ensuite la librairie des composants Simulink, soit en tapant « simulink » dans la fenêtre de commande de Matlab, soit en cliquant sur l'icône  dans la fenêtre de Simulink ou dans la fenêtre de Matlab. Une fenêtre s'ouvre alors avec l'ensemble des composants de Simulink, représentée sur la Figure 2.

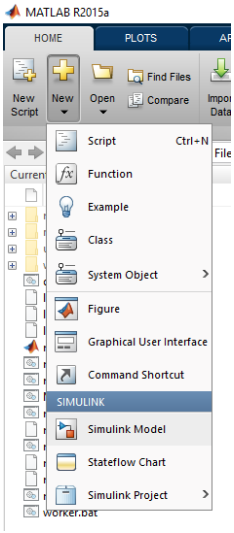


FIGURE 1 – Lancement de Matlab Simulink

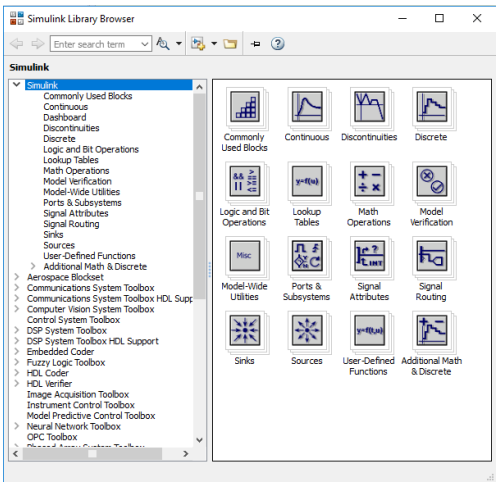


FIGURE 2 – Librairies de composants de Simulink

16 librairies de Simulink apparaissent, dont seules 3 vont nous être particulièrement utiles :

- **Commonly Used Blocks**, qui, comme son nom l'indique, regroupe les composants que vous serez amenés à utiliser le plus souvent,
- **Continuous**, qui regroupe les composants permettant de modéliser les systèmes continus,
- **Sources**, qui regroupe les différents types de consignes qu'il est possible d'imposer au modèle.

En règle générale, il vous est possible d'ajouter un composant dans la fenêtre de Simulink en faisant un clic gauche de la souris sur ce composant, en maintenant le bouton de la souris enfoncé tout en faisant glisser le composant dans la fenêtre de Simulink, puis en relâchant le bouton. Les paramètres du composant peuvent ensuite être modifiés en faisant un double clic avec le bouton gauche de la souris sur le composant dans la fenêtre de Simulink. Les entrées de chaque composant sont représentées par des flèches pointant vers le composant, tandis que les sorties de chaque composant sont représentées par des flèches sortant du composant. Nous allons passer en revue les composants de ces 4 librairies qui vont nous être le plus utiles.

- Commonly Used Blocks : les composants de cette librairie sont représentés sur la Figure 3.

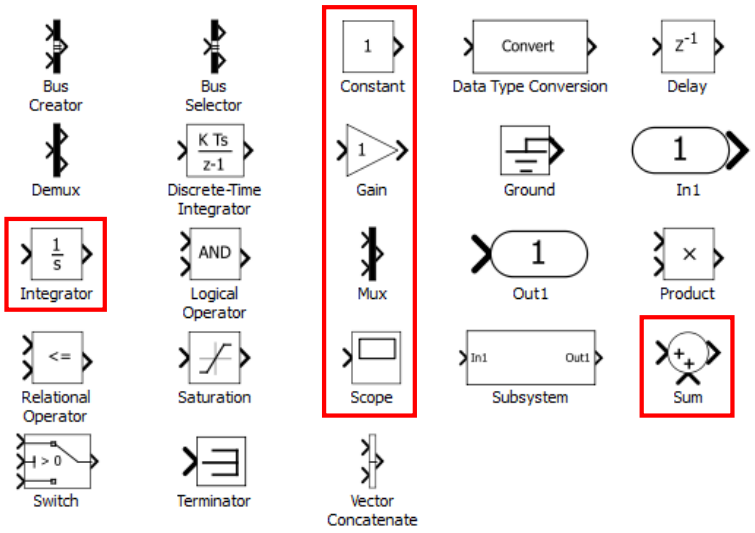


FIGURE 3 – Composants de la librairie Commonly Used Blocks

- **Constant** : ce bloc n'a aucune entrée et une seule sortie. Il modélise une consigne constante, équivalente à un échelon. L'amplitude de l'échelon peut être indiquée dans la case « Constant value » des paramètres de ce composant.
- **Gain** : ce bloc possède une entrée et une sortie. Il modélise un gain pur, au sens où la sortie du composant est égale au produit de son entrée par le gain du composant. Ce gain peut être un scalaire ou une matrice.
- **Integrator** : ce bloc possède une entrée et une sortie. Il modélise un intégrateur, la variable de Laplace p étant ici remplacée par s.
- **Mux** : ce bloc possède au moins 2 entrées et une sortie. Il modélise un multiplexeur permettant de regrouper plusieurs entrées sur une seule sortie. Ce composant est particulièrement utile pour visualiser plusieurs signaux sur un même écran (par exemple, la réponse du système à une consigne et cette consigne, afin de voir si le système est précis). Le nombre d'entrées peut être choisi dans les paramètres du composant.

- **Scope** : ce bloc a une seule entrée et aucune sortie. Il modélise un écran d'oscilloscope affichant le signal en entrée, ou les signaux en entrée si un multiplexeur se trouve en amont.
- **Sum** : ce bloc possède au moins 2 entrées et une sortie. Il permet de générer une sortie qui est le résultat d'une opération (par défaut une somme) sur ses entrées, nous l'utiliserons donc dès qu'il y aura besoin d'un comparateur (et donc pour tous les systèmes asservis). Les opérations à effectuer sur les entrées peuvent être choisies dans les paramètres du composant. Ce composant possède, par défaut, 3 entrées potentielles qui correspondent aux trois faces libres du composant (haut, gauche, bas dans cet ordre), la première (en haut) étant bloquée à l'aide du symbole | tandis que les deux autres sont ajoutées entre elles à l'aide du symbole +. Il est possible de bloquer (symbole |), additionner (symbole +) ou soustraire (symbole −) l'une de ces entrées, ou même d'ajouter des entrées supplémentaires.

- Continuous : les composants de cette librairie sont représentés sur la Figure 4.

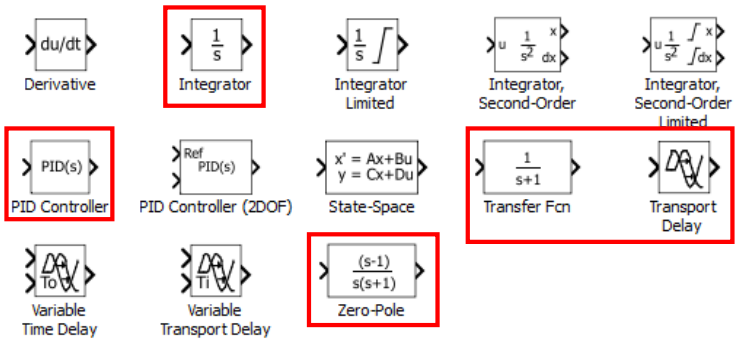


FIGURE 4 – Composants de la librairie Continuous

- **Integrator** : il s'agit du même composant que celui de la librairie Commonly Used Blocks.
- **PID Controller** : ce bloc modélise un correcteur PID avec un gain proportionnel P, un gain intégral I, un gain dérivé D et une constante de temps $\tau = \frac{1}{N}$ pour la correction dérivée filtrée.
- **Transfer Fcn** : ce bloc permet de définir une fonction de transfert à l'aide des polynômes de son numérateur et de son dénominateur. Nous verrons un peu plus tard comment le paramétrer.
- **Transport Delay** : ce bloc modélise un retard de sa sortie par rapport à son entrée. La valeur de ce retard peut être choisie dans les paramètres du composant.
- **Zero-Pole** : ce bloc permet de définir une fonction de transfert à l'aide de ses zéros et de ses pôles. Nous verrons un peu plus tard comment le paramétrer.

- Sources : les composants de cette librairie sont représentés sur la Figure 5.

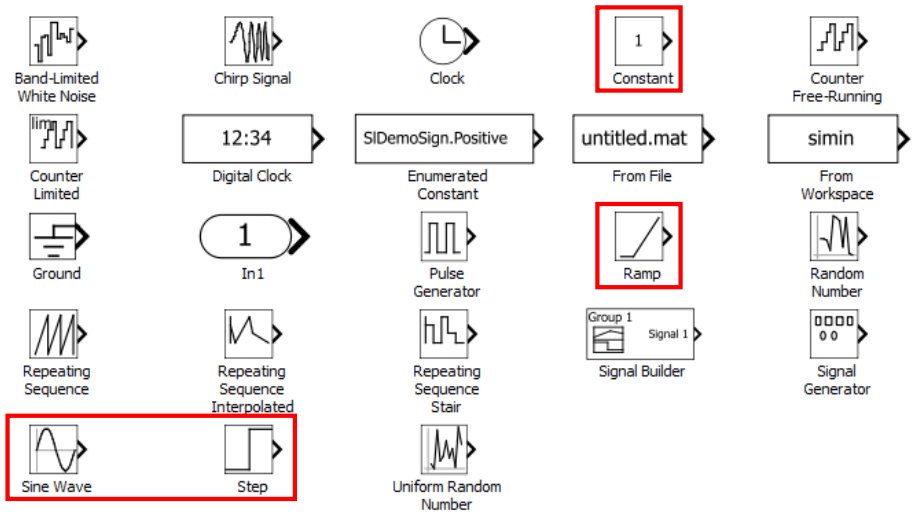


FIGURE 5 – Composants de la librairie Sources

- **Constant** : il s'agit du même composant que celui de la librairie Commonly Used Blocks.
- **Ramp** : ce bloc modélise une consigne en rampe. La pente de la rampe, sa valeur initiale et un éventuel retard peuvent être choisis dans les paramètres de ce composant.
- **Sine Wave** : ce bloc modélise une consigne sinusoïdale. Les paramètres de ce bloc sont l'amplitude de la fonction, sa composante continue (bias), sa pulsation (frequency) et son déphasage (phase).
- **Step** : ce bloc modélise un échelon. Il est différent du bloc Constant car le bloc Constant modélise un échelon de valeur initiale 0 et de valeur finale souhaitée alors que, pour un bloc Step, les valeurs initiale et finale peuvent être choisies, de même qu'un éventuel retard (step time).

On peut remarquer que la librairie Sources ne contient aucun composant permettant de modéliser une impulsion de Dirac. Cela est dû au fait que l'impulsion de Dirac n'est pas physiquement réalisable.

2 Saisie d'une fonction de transfert

Maintenant que nous avons passé en revue les différents composants de Simulink que nous allons utiliser à partir d'aujourd'hui, nous allons voir comment saisir une fonction de transfert. Il est possible de le faire aussi bien sous Matlab que sous Simulink.

2.1 Saisie d'une fonction de transfert sous Matlab

Il existe trois manières de définir une fonction de transfert sous Matlab :

1. la première manière consiste à définir la fonction de transfert à l'aide des polynômes situés au numérateur et au dénominateur de la fonction de transfert. Dans Matlab, un polynôme est défini comme un vecteur dont les coefficients sont les coefficients du polynôme, du coefficient de plus haut degré à celui de plus bas degré. Par exemple, le polynôme p^2+2p+3 sera défini dans Matlab à l'aide du vecteur [1 2 3] ou $[1,2,3]$. Ainsi, pour définir la fonction de transfert $H(p)=\dfrac{p^2+2p+1}{p^2-2p-1}$, on peut :
   (a) soit définir dans Matlab le numérateur (num = $[1,2,1]$) et le dénominateur (den = $[1,-2,-1]$) avant de définir la fonction de transfert à l'aide de la commande tf (pour transfer function) : sys = tf(num,den),
   (b) soit définir directement la fonction de transfert comme : sys = tf($[1,2,1],[1,-2,-1]$).
2. la deuxième manière consiste à définir la fonction de transfert à l'aide de ses zéros z_i, de ses pôles p_i, et du gain K tel que la fonction de transfert peut s'écrire sous la forme
$$K\frac{\prod_i (p-z_i)}{\prod_j (p-p_j)}$$
   Si les zéros et les pôles de la fonction de transfert ne sont pas évidents, ils peuvent être déterminés sous Matlab à l'aide de la commande roots. On peut ainsi :
   (a) soit définir dans Matlab les zéros (z = $[-1,-1]$), les pôles (p = roots($[1,-2,-1]$)) et le gain (K = 1) avant de définir la fonction de transfert à l'aide de la commande zpk : sys = zpk(z,p,K),
   (b) soit définir directement la fonction de transfert comme : sys = zpk($[-1,-1]$, roots($[1,-2,-1]$), 1).
3. la troisième manière (la plus simple) consiste à définir la fonction de transfert de manière littérale. Pour cela, il faut dans un premier temps indiquer quelle est la variable symbolique de Laplace à utiliser pour exprimer les fonctions de transfert, en utilisant (par exemple) la commande « p = tf('p') » si la variable symbolique de Laplace est notée p. Il nous suffit ensuite de définir la fonction de transfert comme : sys = $(p \wedge 2 + 2 * p + 1)/(p \wedge 2 - 2 * p - 1)$.

Il est possible de passer de l'une à l'autre des deux premières méthodes en utilisant les commandes tf2zp et zp2tf :

- la commande « [z,p,k]=tf2zp(num,den) » fournira les zéros z, les pôles p et le gain k de la fonction de transfert de numérateur num et de dénominateur den,
- la commande « [num,den]=zp2tf(z,p,k) » fournira le numérateur num et le dénominateur den de la fonction de transfert de zéros z, de pôles p et de gain k.

2.2 Saisie d'une fonction de transfert sous Simulink

Il existe deux manières de définir une fonction de transfert sous Simulink, à l'aide des deux composants de la librairie Continuous identifiés précédemment. Ces deux manières sont équivalentes aux deux premières manières de définir une fonction de transfert sous Matlab :

1. la première manière consiste à définir la fonction de transfert à l'aide des polynômes situés au numérateur et au dénominateur de la fonction de transfert en utilisant le composant Transfer Fcn. Il suffit pour cela de renseigner, dans les paramètres du composant, les polynômes au numérateur et au dénominateur. Si ces polynômes ont déjà été définis sous Matlab (par exemple, à l'aide des noms num et den), il suffit alors de renseigner leurs noms dans les paramètres du composant, sinon, les vecteurs associés aux deux polynômes peuvent être renseignés,
2. la deuxième manière consiste à définir la fonction de transfert à l'aide de ses zéros, de ses pôles et de son gain, en utilisant le composant Zero-Pole. Là encore, si les zéros, pôles et gain de la fonction de transfert ont déjà été définis sous Matlab, il suffit de renseigner leur nom dans les paramètres du composants, sinon, les vecteurs associés peuvent être renseignés.

3 Saisie d'un schéma-bloc

Après avoir vu comment définir une fonction de transfert, que ce soit sous Matlab ou sous Simulink, nous allons maintenant voir comment définir un schéma-bloc. Là encore, il existe deux manières de le faire, même si ces deux manières ne sont pas équivalentes. Simulink va nous permettre de saisir **graphiquement** un schéma-bloc en reliant tous les blocs entre eux, permettant ainsi de comprendre le fonctionnement du système, mais sans permettre de connaître la fonction de transfert équivalente au schéma-bloc. Inversement, Matlab se limitant à une interface de commande, il ne nous permettra pas de « visualiser » le schéma-bloc, mais nous allons voir que Matlab peut calculer facilement la fonction de transfert équivalente. Ces deux méthodes sont donc complémentaires.

3.1 Saisie d'un schéma-bloc sous Simulink

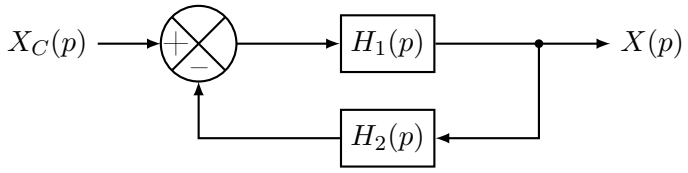
La saisie d'un schéma-bloc sous Simulink est on ne peut plus simple. Il vous suffit de définir les différentes fonctions de transfert dont vous avez besoin, de placer le ou les comparateurs, la consigne (et éventuellement la perturbation) et le Scope en sortie pour visualiser la réponse du système (et éventuellement sa consigne, en utilisant un multiplexeur). Et ensuite, vous n'avez plus qu'à relier les composants les uns aux autres en tirant, avec la souris, sur les liens (représentés par des flèches) qui se trouvent en entrée ou en sortie des différents composants, Simulink liant automatiquement les liens entre eux dès lors qu'ils sont suffisamment proches.

3.2 Calcul de la fonction de transfert équivalente à un schéma-bloc sous Matlab

La fonction de transfert équivalente à un schéma-bloc peut être déterminée très facilement à l'aide des commandes series, parallel et feedback. Si deux fonctions de transfert $H_1(p)$ et $H_2(p)$ sont associées sous Matlab à deux systèmes sys1 et sys2, alors :

- la fonction de transfert équivalente à la mise en série de ces deux systèmes peut être déterminée comme : sys = series(sys1,sys2),

- la fonction de transfert équivalente à la mise en parallèle de ces deux systèmes peut être déterminée comme : sys = parallel(sys1,sys2),
- la fonction de transfert équivalente à la structure représentée sur la figure ci-dessous peut être déterminée comme : sys = feedback(sys1,sys2). Remarquez que la commande feedback fournit la fonction de transfert équivalente dans le cas d'un soustracteur ; si c'est un sommateur qui est utilisé, la fonction de transfert équivalente sera déterminée comme sys = feedback(sys1,−sys2) ou sys = feedback(sys1,sys2,+1).



4 Exercices d'entraînement

On prendra dans les exemples qui suivent $\forall i \in [\![1,7]\!], H_i(p) = \dfrac{p+i+1}{p+i}$. Saisir chacun des schémas-blocs suivants sur Simulink, déterminer leur fonction de transfert équivalente, et vérifier par simulation que la fonction de transfert obtenue est correcte.

- exemple n°1

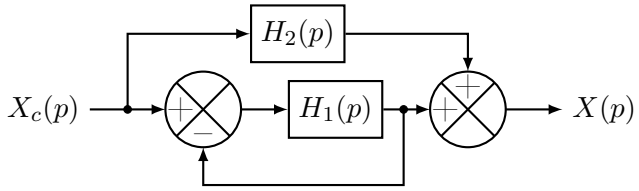


FIGURE 6 – Exemple n°1

- exemple n°2

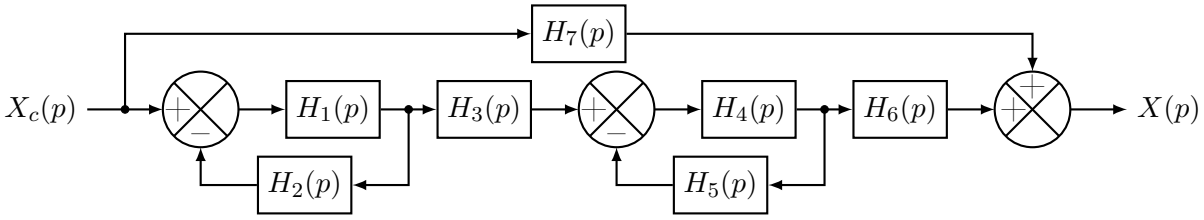


FIGURE 7 – Exemple n°2

- exemple n°3

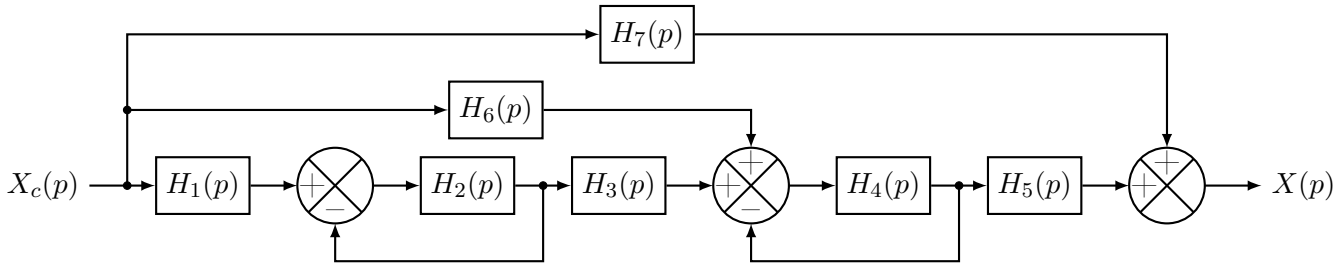


FIGURE 8 – Exemple n°3