**Université Mohamed Boudiaf - M'sila**

**Faculté de Technologie**
**Département de Génie Electrique**
**Master Académique : CE**
**TP : Techniques de la commande électrique**

## Annexe TP. N° : 01

# Simulink M-file S-Function

**Présentation de l'environnement MATLAB/SIMULINK**

**Matlab est un logiciel destiné principalement au calcul scientifique, à la modélisation et à la simulation. Le noyau de calcul est associé à l'environnement Simulink, permettant une modélisation basée sur des schémas-blocs.**
**Des bibliothèques spécialisées sont disponibles (les "Toolboxes") pour la plupart des domaines scientifiques nécessitant des moyens de calcul importants : automatique, traitement de signal, mathématiques appliquées, télécommunications, etc.**
**Des modules temps réel, développés autour des produits dSPACE (Real Time Interface, TRACE, COCKPIT) sont également proposés et rendent l'environnement de travail particulièrement attractif. L'utilisateur dispose en effet de tous les outils nécessaires à l'élaboration d'une application sur procédé réel, de la simulation à la supervision.**

**Introduction aux S-Functions**

**L'élaboration d'un algorithme sous Simulink est basée sur une description arborescente par schémas-blocs. Chaque bloc fonctionnel peut être décrit de deux manières :**

- **graphiquement. C'est la méthode la plus courante et la plus facile à utiliser ;**
- **par une S-function (fonction système).**

**Une S-Function est un langage de programmation d'un bloc Simulink. Les S-Functions permettent d'ajouter nos propres blocs aux modèles proposés par Simulink. Par des règles simples et préalablement établies, il est possible de mettre en application nos propres algorithmes. Les S-Functions peuvent être écrites en langage Matlab, C++, C, ADA ou Fortran. Les S-Functions emploient une syntaxe spéciale d'appel qui permet d'interagir avec Simulink. La forme d'une S-Function est très générale et peut s'adapter aux systèmes continus, discrets et hybrides.**

**Fonctionnement d'une S-Function**

**Tout bloc S-function contenu dans un schéma Simulink possède les caractéristiques suivantes :**

**- un vecteur d'entrée u,**
**- un vecteur de sortie y,**
**- un vecteur d'état x.**

**Le vecteur d'état peut être discret, continu ou une combinaison des deux. Les vecteurs u, x et y sont définis de la manière suivante :**

$$y = f_o(x,t,u) \quad (\text{calcul de la sortie})$$
$$x_d = f_d(x,t,u) \quad (\text{mise à jour des états discrets})$$
$$x_c = f_c(x,t,u) \quad (\text{calcul des dérivées})$$
$$\text{où } x = \begin{pmatrix} x_c \\ x_d \end{pmatrix}$$

**Conventions d'appel des S-functions**

**Au cours de la simulation, Simulink appelle à chaque itération les blocs S-functions et demande un calcul des sorties, une mise à jour des états discrets ou un calcul des dérivées. Des routines complémentaires assurent une initialisation et une sortie correctes des différentes tâches. Dans le cas d'une S-function écrite en langage MATLAB (M-file), ces appels sont gérés par le contenu du paramètre flag dans la fonction appelante comme suit :**

| En utilisant la fonction 'if' | En utilisant la fonction 'switch' |
|---|---|
| **if flag == 0**<br>**Tâches;**<br>**elseif flag==1**<br>**Tâches;**<br>**elseif flag==2**<br>**Tâches;**<br>**elseif flag== 3**<br>**Tâches;**<br>**else**<br><br>**Sys=[];**<br><br>**end** | **switch flag**<br><br>**case 0**<br>**Tâches;**<br>**case 1**<br>**Tâches;**<br>**case 2**<br>**Tâches;**<br>**otherwise**<br><br>**Tâches;**<br><br>**end** |

**Cet algorithme peut être explicité comme suit :**

***Case 0: Initialisation***

**Premièrement, quand Simulink envoi flag=0, il vise à recevoir une déclaration des variables (x, t, y, u) et une initialisation du vecteur x:**

**La fonction doit retourner la variable standard 'sys', contenant les informations suivantes:**

**sys= [nombre des états continues, nombre des états discrets, nombre des sorties, nombre des entrées, 0, direct feedthrough, nombre des périodes d'échantillonnages]**

**De plus, il attend dans x0 le vecteur d´état initial.**

***Case 1: Calcul des dérivées***

**Quand Simulink envoi flag=1, il s'attend à ce que la fonction rende les dérivées en fonction du temps. Admettant que le système d'équations différentielles est défini par l'équation suivante :**

$$\frac{dx}{dt} = f_c(x,u) = \left[ f_{c1}\left(x(1), x(2), \cdots\right) \quad f_{c2}\left(x(1), x(2), \cdots\right) \quad \cdots \right]$$

**Les dérivées sans retournées dans le vecteur 'sys' comme suit :**

$$sys = \left[ f_{c1}, \quad f_{c2}, \quad \cdots \right]$$

***Case 2: Mise à jour des états discrets***

**Quand Simulink envoi flag=2, il s'attend à ce que la fonction rende la mise à jour des états discrets. Admettant que le système d'équations aux déférences est défini par l'équation suivante :**

$$x\left[k+1\right] = f_d(x,u) = \left[ f_{d1}\left(x(1), x(2), \cdots\right) \quad f_{d2}\left(x(1), x(2), \cdots\right) \quad \cdots \right]$$

**Les dérivées sans retournées dans le vecteur 'sys' comme suit :**

$$\text{sys} = \left[ f_{d1}, \quad f_{d2}, \quad \cdots \right]$$

***Case 3: Calcul des sorties***

**Quand Simulink envoi flag=3, il s'attend à ce que la fonction rende les valeurs des sorties. Admettant que les sorties sont calculées par l'équation suivante :**

$$y = f_o(x,u) = \left[ f_{o1}\left(x(1), x(2), \cdots\right) \quad f_{o2}\left(x(1), x(2), \cdots\right) \quad \cdots \right]$$

**Tout dépend du nombre des sorties, la variable 'sys' peut être un vecteur ou un scalaire, elle est définie comme :**

$$\text{sys} = \left[ f_{o1}, \quad f_{o2}, \quad \cdots \right]$$

***Case 4: Calcul du temps nécessaire pour la mise à jour de la variable discrète suivante.***

**Case 9: Exécution de toute tache que vous voulez accomplir à la fin de la simulation.**

**Cases 5-8: Ne sont pas utilisés; en conséquence, la tâche "otherwise" est attribué à ces arguments (une erreur est à signaler si l'un de ses cas est choisi).**

**<u>Utilisation d'un bloc S-Function</u>**

**Pour utiliser une S-Function on ouvre la Simulink Library Browser, on sélectionne Simulink puis User-Defined-Functions. On place une S-Function dans un nouveau modèle. On ouvre ce bloc et on renseigne le nom de la S-Function que l'on a créé. Généralement le même nom que le fichier écrit dans le langage choisit sans l'extension. Il est possible de passer des paramètres à la S-Functions.**

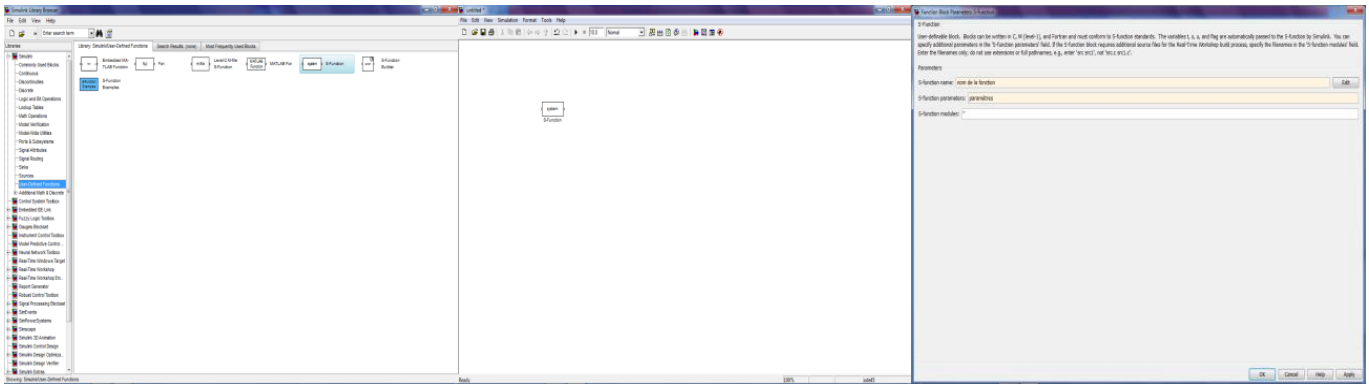

***Figure (1) : Démarches nécessaires pour introduction une s-function dans l'environnement simulink***

**<u>Exemple</u>**

**Le schéma de commande symétrique (pleine onde) de l'onduleur monophasé en pont est donné par la figure suivante :**

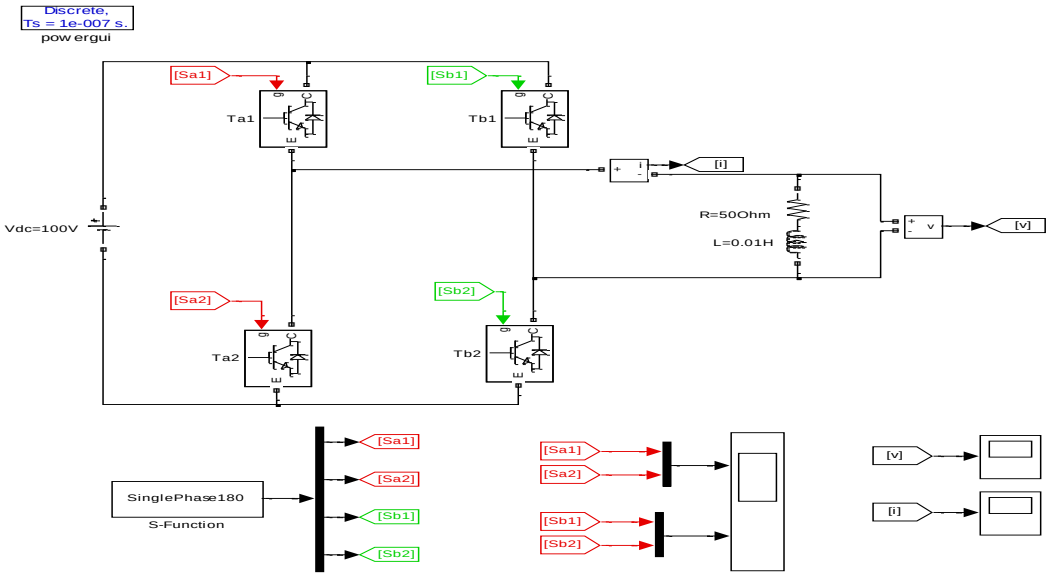


*Figure (2) : Schéma block de simulation d'un onduleur monophasé à commande symétrique*

**La fonction générant la commande est programmée comme suit:**

```
function [sys,x0,str,ts]=SinglePhase180(t,x,u,flag,Ts)

% Commande pleine onde (commande 180) d'un onduleur de tension monophasé

f=50;       % la fréquence de la tension de sortie
T=1/f;      % la période de la tension de sortie

%-------------------------------------------------------------------------
if flag==0

NumContStates = 0;       % Nombre des variables d'état continues
NumDiscStates = 0;       % Nombre des variables d'état discrètes
NumOutputs = 4;          % Nombre des sorties
NumInputs = 0;           % Nombre des entrées
DirectFeedthrough=0;     % Valeur du Direct Feedthrough
NumSampTimes=1;          % Nombre des périodes d'échantillonnages

sys=[NumContStates,NumDiscStates,NumOutputs,NumInputs,0,DirectFeedthrough,NumSampTimes];

x0=[];
str=[];
ts=[Ts 0];

elseif flag==3

% Commande 180

tt=mod(t,T);

if tt > 0 && tt <= T/2, Sa1=1; else Sa1=0; end
if tt > 0 && tt <= T/2, Sb1=0; else Sb1=1; end

% Commandes complémentaires

Sa2=1-Sa1;
Sb2=1-Sb1;

% Sorties
```

```
sys=[Sa1;Sa2;Sb1;Sb2];

else

 sys=[];

end
```