MODELING OF PLANT CONTROLLERS USING THE PSS/E TO MATLAB-SIMULINK INTERFACE (PMSI) IN PSS/E 30

Kah Leong, Koo
Consultant
kahleong.koo@shawgrp.com

Modeling a plant control using standard libraries

The plant control models can be graphically constructed in block diagram form which resembles the physical models, manufacturer’s models, IEEE recommended representation or any other derived models.

The standard blocks provided within the Matlab-Simulink library are very comprehensive and provide a first basis in constructing any models. To illustrate this point, the simple exciter model in Figure 1 uses two transfer functions and an adder from the standard Matlab-Simulink library. But more importantly, it is also very easy to model this model in a state-space form using the State-Space element provided. This is a general control method for representing controls. Using the transfer function model of Figure 1 as an example, we can demonstrate this general form of modeling. The equations below represent the state-space equations for Figure 1.

Figure 1 - Simple exciter model using two transfer function blocks and a saturation function block
For transfer Fcn element in Figure 1:

From established control theory,

\[ T(s) = \frac{bo + b1/s}{1 - (-a1/s)} \]  

(1)

\[ x = -a1.x + (b1 + a1.bo)u \]  

(2)

\[ y = x1 + bo.u \]  

(3)

For transfer function element in Figure 1,

\[ T(s) = \frac{0.1 + 0.1/s}{1 - (-0.1/s)} \]  

(4)

Comparing equations (1), (4) and (2) to obtain the coefficients,

\[ x1 = -0.1.x1 + 0.09.u1 \]  

(5)

Comparing equations (1), (4) and (3) to obtain the coefficients,

\[ y1 = x1 + 0.1.u1 \]  

(6)

For transfer Fcn1 element in Figure 1:

\[ y2 = x2 = Efd \]  

(7)

\[ 100.u2 = 0.1.x2 + x2 \]  

(8)

Re-arranging equation (8),

\[ x2 = 1000.u2 - 10.x2 \]  

(9)

The same control model of Figure 1 can now be represented using two state-space models as shown in Figure 2 (with the state-space dialog windows opened showing the relevant parameters derived from equations (5), (6), (7) and (9)). Note that the user can specify the “Initial conditions” for the initial states for the two state-space blocks if known. These are zeroes by default and will be automatically calculated when used in a PSS/E dynamic simulation. For the more advance user, a custom S-function element block can be created and Figure 3 shows a single state-space custom created block for the same control model in this particular model with a non-windup limiter. This Figure 3 model is shown with the customized user-interface dialog to accept the relevant parameters (also entries for the VMAX and VMIN limits for the non-windup limiter).

The models shown in Figure 1 and Figure 2 for the exciter models use a windup limiter. For a non-windup limiter to be modeled as in Figure 3, this can be implemented using custom S-function element blocks which are demonstrated in the next section on creating custom libraries through S-functions.
Figure 2 - Simple exciter model using two state-space blocks and a saturation function block
Figure 3 - Simple exciter model using user-written code S-function

The two transfer functions of Figure 1 can also be represented as a single S-function block as shown in Figure 3 in which the derived equations in state-space form can be represented as:

\[
\begin{bmatrix}
\dot{x}_2 \\
\dot{x}_1
\end{bmatrix} = \begin{bmatrix}
\frac{Kb}{Tb} & -\frac{1}{Tb} \\
-\frac{1}{Ta} & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} + \begin{bmatrix}
0 & 0
\end{bmatrix} \begin{bmatrix}
u_1
\end{bmatrix}
\]

(10)

where,

\[ u_2 = Ve + Gain \times Ve \]  

(11)

**Modeling a plant control using own custom libraries**

For users who need to further exploit the more advanced modeling capabilities of Matlab-Simulink for more specialized requirements can create their own custom libraries. Although custom libraries can be created using subsystems built from standard libraries, this is not what we are interested in here. This section focuses on the powerful S-function which is available as part of Matlab-Simulink block. This allows the user to model part/full of the model in coded high level language like Matlab®, C, C++, Ada, or Fortran. The advantages offered through using S-functions are:
- Extremely general and accommodate continuous, discrete, and hybrid systems.
- Place the S-function block as part of the Matlab-Simulink library browser available for use just like any other standard Matlab-Simulink blocks (similar to a Component Object Model (COM) approach in developing re-usable code).
- A masking facility to easily set up a customized user-interface for the S-function.
- The S-function created uses Matlab-Simulink solvers and interacts in a similar way to the built-in Matlab-Simulink blocks.
- If the S-function code is written using C, C++, Ada or Fortran, the code is compiled and linked into a dynamic executable which can be distributed for use without exposing the source code itself.

Modeling the 1992 IEEE type AC1A excitation system

As had been mentioned in the previous section for the model illustrated in Figure 3, an S-function was used with a specific code written to emulate a non-windup limiter. In demonstrating the usefulness of this feature, the 1992 IEEE type AC1A excitation system model will be modeled using Matlab-Simulink. The input signals VOEL and VUEL are not connected as no limit protection is used in this model which is used in the dynamic simulation demonstration in the later part of this article.

![Figure 4 - 1992 IEEE type AC1A exciter model](image)

Figure 4 shows this exciter modeled using Matlab-Simulink. In particular, there are three S-function Matlab-Simulink blocks (also highlighted in colours) used in this model. The three S-function blocks developed for this model are (1) S-function: Voltage-regulator, (2) S-function: FEX=F(IN) and (3) S-function: Exciter saturation Function using quadratic method.

Most of the languages used to build the S-function can be started from commonly used templates or standard templates provided by Matlab-Simulink. In the S-functions developed by Shaw PTI, C was the chosen language as Matlab-Simulink provides extensive SimStruct functions available through the callback methods - (1) Callback methods are S-functions that allow different tasks to be performed at different stages of the Matlab-Simulink simulation which are provided within Matlab-Simulink, (2) SimStruct functions are a well documented set of functions for accessing data for the written S-function. The source codes in Figure 6 and Figure 8 for the S-functions illustrate some SimStruct functions used within the callback methods (the main function) used.
S-function: voltage-regulator

Figure 5 shows Shaw PTI’s customized graphical user-interface S-function dialog using the masking facility for a typical voltage-regulator with non-windup limits S-function block. This S-function was written to allow the non-windup behavior of voltage-regulator to be modeled. Figure 6 in turn shows part of the C-code that defines this S-function showing the callback method mdlDerivatives in which the derivative of the continuous state is calculated for when the input signal is within and outside the limits defined by VMAX and VMIN shown in the dialog of Figure 5.

Figure 5 - Graphical user-interface for voltage-regulator S-function dialog
Figure 6 - Calculating the derivative for the continuous state for the voltage-regulator S-function

S-function: $F_{EX}=F(IN)$

For a wide range of IEEE Type AC (Alternator Supplied Rectifier) and IEEE Type ST (Static Excitation Systems) requires the rectifier regulation to be modeled. This can in general be described by the equations:

If $I_N < 0.433$  \[ F_{EX} = 1 - 0.577I_N \]  
(12)

If $0.433 < I_N < 0.75$  \[ F_{EX} = \sqrt{0.75 - (I_N^2)} \]  
(13)

If $I_N > 0.75$  \[ F_{EX} = 1.732(1 - I_N) \]  
(14)

If $I_N > 1$  \[ F_{EX} = 0 \]  
(15)

Although these equations can be modeled using standard functions already provided within Matlab-Simulink, an S-function was specially created to accommodate the different modes required easily and neatly. Figure 7 shows Shaw PTI’s customized graphical user-interface for the S-function dialog using the masking facility for the rectifier regulation S-function block. In this dialog, provision is also made to allow the different values to be modified to allow the slopes of the rectifier regulation to be modified as to when the different modes come into operation. For example, for the parameter val3 ($FEX=1-val3*IN$) in the dialog of Figure 7, this corresponds to equation (12) and the value of val3 is 0.577.
Figure 7 - Graphical user-interface for rectifier regulation S-function dialog

Figure 8 in turn shows part of the C-code that defines this S-function showing the callback method mdlOutputs in which the rectifier regulation equations are described.
Figure 8 - Calculating the rectifier regulation for the different modes for rectifier regulation S-function
S-function: exciter saturation function using quadratic method

In the case of the saturation function, the IEEE recommends specifying Se at maximum field voltage (Semax) and at 0.75 of maximum field voltage (Se0.75max). The saturation Se can then be determined at any field voltage by linear interpolation, quadratic method or fitting an exponential function. S-functions were written for the linear, quadratic and exponential methods of calculating Se to accurately and neatly reflect the IEEE recommendation. Figure 9 shows Shaw PTI’s customized graphical user-interface for the S-function dialog using the masking facility for working out Se×Ve using the quadratic approach for this S-function block.

![Figure 9 - Graphical user-interface for Se×Ve S-function dialog using quadratic method](image)

Incorporating own S-functions within Matlab-Simulink library browser

The S-functions that have been written are re-usable and conveniently made available as part of a Matlab-Simulink library. What this means is the S-functions become Matlab-Simulink building blocks which can be used in creating other models. The objective here is not to unnecessarily create S-functions but when this needs to be done, it is not a one-off but is re-usable and made available as an independent object.

Figure 10 shows all the S-functions written being grouped under Shaw PTI: Custom blocks made accessible through the Matlab-Simulink library browser. Using them is a matter of drag and drop in the Matlab-Simulink model window.
Using these models in a dynamic simulation

Figure 11 shows the Power Technologies large benchmark case consisting of 7917 buses, 13014 branches used to demonstrate participation of the Matlab-Simulink models in a dynamic simulation.

The exciter models created in Matlab-Simulink are directly accessible as part of a PSS/E dynamic simulation. Their inclusion into a PSS/E dynamic simulation is specified through the dynamics data input file. Figure 12 shows the dynamics data input file being specified with the Matlab-Simulink exciter model ESAC1A of Figure 4 connected to the machines connected to the bus with BUSID 1, 2 and 7. The keyword ‘USRMAT’ informs PSS/E that a Matlab-Simulink model is being used. This file is read by the DYRE activity to generate the appropriate calls in the CONEC subroutine.

Figure 13 shows the CONEC generated call to MATPLX which passes the Matlab-Simulink model name as an argument, here ESAC1A. This subroutine invokes the PSS/E to Matlab-Simulink interface to perform the Matlab-Simulink part of the dynamic simulation.

Figure 14 shows the dynamic plots for the exciter voltages from the Matlab-Simulink AC1A exciter models showing their dynamic responses for a bus fault applied at bus with BUSID 1 at 0.5 second and cleared at 0.7 second.
Figure 11 - Power Technologies large benchmark case used to demonstrate the dynamic simulation
Figure 12 - Dynamics data input file for specifying Matlab-Simulink models

```
Figure 13 - USRXXX containing call to Matlab-Simulink models through MATPLXL

C  PLANTRELATEDUSERMODELS
C
SUBROUTINE USRXXX(MC,SLOT,IT)
INTEGER MC,SLOT,IT
SELECT (IT)
(1001) CALL MATPLXL(MC,SLOT,IT,'ESAC1A')
(OTHERWISE) CALL USMAIR(MC,IT)
FIN
RETURN
END
C
```
Figure 14 - Dynamic plots of exciter voltages for the Matlab-Simulink AC1A exciter models