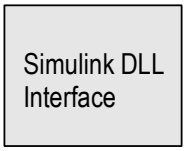
Simulink DLL based Interface

SimulinkDLL1



César Martin (RTE-France), Jean Mahseredjian, 8/24/2019 12:09:00 PM

1 Description

This device named “Simulink DLL Interface” (SDI) is used to interface EMT with DLLs created using Simulink® (MATLAB®). The SDI is an EMTP Toolbox.
Once a DLL is created using Simulink and a compiler, this device is used to create all the necessary connections (pins) and provide access to some modifiable parameters on the Simulink side.
The interfacing pins (ports) on the Simulink side must be created by the user. The interfacing pins in EMTPWorks are automatically created with the SDI device. The user must then finalize all the desired connections in EMTPWorks to use the Simulink code in the simulations on EMTP side.
This version of the SDI is called following the solution of EMTP network equations (as when calling a control diagram in EMTP) and the outputs from an SDI block are used in the network side solution at the following time-point. This means that there is a time-step delay between the network solution on EMTP side and the solution of an SDI block.

2 Requirements and Compatibility

The user must have a valid MATLAB license with Simulink. The following MATLAB Toolboxes are minimal requirements, but this may change according to MATLAB versions. Required Toolboxes:

1. Matlab Coder
2. Simulink Coder
3. Embedded Coder

Some other Toolboxes may be required according to Simulink designs.
This version of the SDI has been tested with the following MATLAB versions: R2014a, R2014b, R2015a and R2017b.

It is noticed that under some circumstances, especially when S-Functions and MATLAB functions are used in Simulink diagrams, the simulation results may be corrupted or the DLL generation process may fail. It is the user's responsibility to verify coherency in simulation results.

In addition to the above MATLAB requirements, the user must install a C++ compiler. The compatible compiler is Microsoft Visual Studio C++ version 2013.

3 DLL generation and integration steps

In the following sections, the name ApplicationDir designates the application folder (path):

1. On a 64 bit Windows system the default value is C:\Program Files (x86)\EMTPWorks.
2. On a 32 bit Windows system the default value is C:\Program Files\EMTPWorks.

The EMTP Toolboxes are located in the folder ApplicationDir\Toolboxes.
The SDI Toolbox is located in the folder ApplicationDir\Toolboxes\SimulinkDLL.

The name UserDataFolder designates the EMTP Documents folder in Windows. On a typical computer this is given by:
C:\Users\<User Name>\Documents\EMTP

The name ModelName.dll designates the DLL file generated from Simulink.

3.1 Initial setup in MATLAB

The following commands must be executed in MATLAB to prepare for automatic generation of DLLs from Simulink.

1. The TLC_EMTP folder
   As a first step it is necessary to access the TLC_EMTP folder. This folder is normally located in the UserDataFolder. An initial version of this folder is also found in ApplicationDir\Toolboxes\SimulinkDLL. The user must run the following command in the MATLAB window:
   ```
   >> addpath(genpath('UserDataFolder\TLC_EMTP'))
   >> savepath
   ```
2. Selecting the compiler in MATLAB
   If the C++ compiler is not already selected in MATLAB, the user must run the following MATLAB command:
   ```
   >> mex -setup
   ```

3.2 Steps for creating a DLL

3.2.1 Sample Simulink diagram

The Simulink example shown in Figure 1 is used here to demonstrate the DLL generation steps. The top level model (design) is shown in Figure 1a and the contents of the sample Subsystem are shown un Figure 1b.

The Simulink diagram may contain the following:

1. One or several Input ports (Inports) for interfacing on EMTPWorks side
2. One or several Output ports (Outports) for interfacing on
3. The ports may be real or complexe of dimension 1
4. The ports may be real or complexe of dimension greater than 1
5. One or several Subsystems
6. Different sampling rates
7. Tunable parameters: parameters than can be modified in EMTPWorks side
8. MATLAB functions
9. S-Functions

Under some circumstances the MATLAB functions and S-Functions may not perform as expected and the user must verify that the simulation results are coherent. The Simulink model (mdl) file corresponding to Figure 1 can

be found in the folder ApplicationDir\Toolboxes\SimulinkDLL\Examples\SimpleDemo. The MATLAB Function and the S-Function are omitted from this file.

The individual components on Simulink side must be correctly specified using Simulink input data tabs. In this example:

1. The ports In_complex_1 and In_complex_2 have “Port Dimensions” set to 1 and “Signal type” set to complex. The “Sample time” is -1 for inherited.
2. For the In_real port, “Port dimensions” is set to -1, “Sample time” is set to Ts, the “Signal type” is auto.
3. The Out_complex_vector is set to inherit its “Port dimensions” and “Signal type” is auto which makes it of dimension 2 and of complex type.
4. The Out_vector_real port inherits its dimension of 2 and its “Signal type” type is also inherited as real-type.
5. For the Subsystem:
    a. the gain block Gain_K is set to the variable value K
    b. the gain block Gain_L is set to the variable value L
    c. the Discrete-Time Integrator has an inherited sample time
    d. the Discrete-Time Integrator_Tsx100 has a sample time of 100*Ts
6. The MATLAB function contains the following

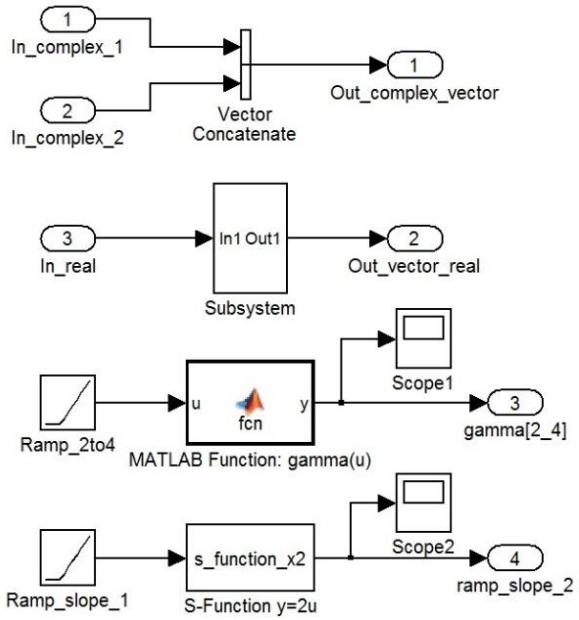
    ```
    function y = fcn(u)
    %#codegen
    y = gamma(u);
    ```

    In this case the %#codegen directive (pragma) is used to indicate that you intend to generate code for the MATLAB algorithm. Adding this directive instructs the MATLAB code analyzer to help you diagnose and fix violations that would result in errors during code generation.

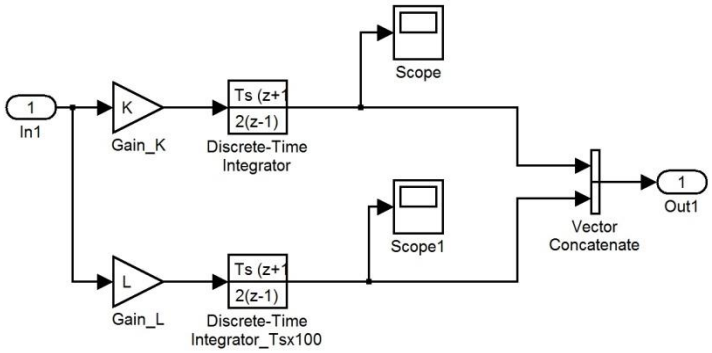
Since this example contains variables in some of its blocks, it is necessary to initialize those variables in the MATLAB work space. In this case it is necessary to enter:

```
>> K = 2;
>> L = 3;
>> Ts = 10e-6;
```

Or to run the provided m-file init_data.m.



a) Top level model



b) Subsystem

Figure 1 Sample Simulink diagram

3.2.2 Code generation steps

The steps are as follows:

1. Select the configuration parameters using the Simulink menu “Simulation>Model Configuration Parameters” and click on Code Generation.
    a. It is necessary to locate and select the “System target file” EMTP.tlc using the Browse button as shown in Figure 2.
        i. In the shown example, there is no referenced model, but it is allowed to use such models by setting exactly the same target file (EMTP.tlc) and the same configuration parameters in all references models. It is not allowed to use tunable parameters when referenced models are used.
    b. Under the “Code Generation” submenu, select “EMTP code generation options” (see Figure 3):
        i. Delete source files after code compilation: this option can be used to delete source code for confidentiality reasons.
        ii. Simulink Model is multi time step compatible: this option is used to generate a DLL that is compatible with multiple time steps. See Section 4 for more details on this option.
2. All files related to this model (design) must be located in the same folder as the model file.
3. Select Solver (see Figure 4) and verify that the Fixed-step size is set to the variable Ts.
4. In this optional step it is possible to select tunable parameters. The tunable parameters are parameters that can be modified from EMTPWorks without recompiling (regenerating) the Simulink DLL.
    a. In the menu Optimization>Signals and Parameters, click on the Configure button (when “Inline parameters” is checked).
    b. Select the “Source list” being “Referenced workspace variables”.
    c. Select the tunable parameters, and click on “Add to table”, as shown in Figure 5. The tunable parameters can be of type real, logical or integer. Complex numbers, vectors and matrices are not allowed. Some parameters (like Ts) cannot be tunable.
    d. Click OK to proceed.
5. At this stage the setup is ready to generate the DLL by clicking on the Build button of Figure 2.
    If there are no error messages in MATLAB, then the build is successful and two new folders will appear in the same folder as the compiled model file folder: slprj and c_EMTP_rtw. In this example the ModelName_EMTP_rtw folder is tests_R2014b_10us_full_EMTP_rtw.
    The final DLL file of the model is also located in the same folder as the model file. Its generic name is ModelName.dll (tests_R2014b_10us_full.dll for this example).

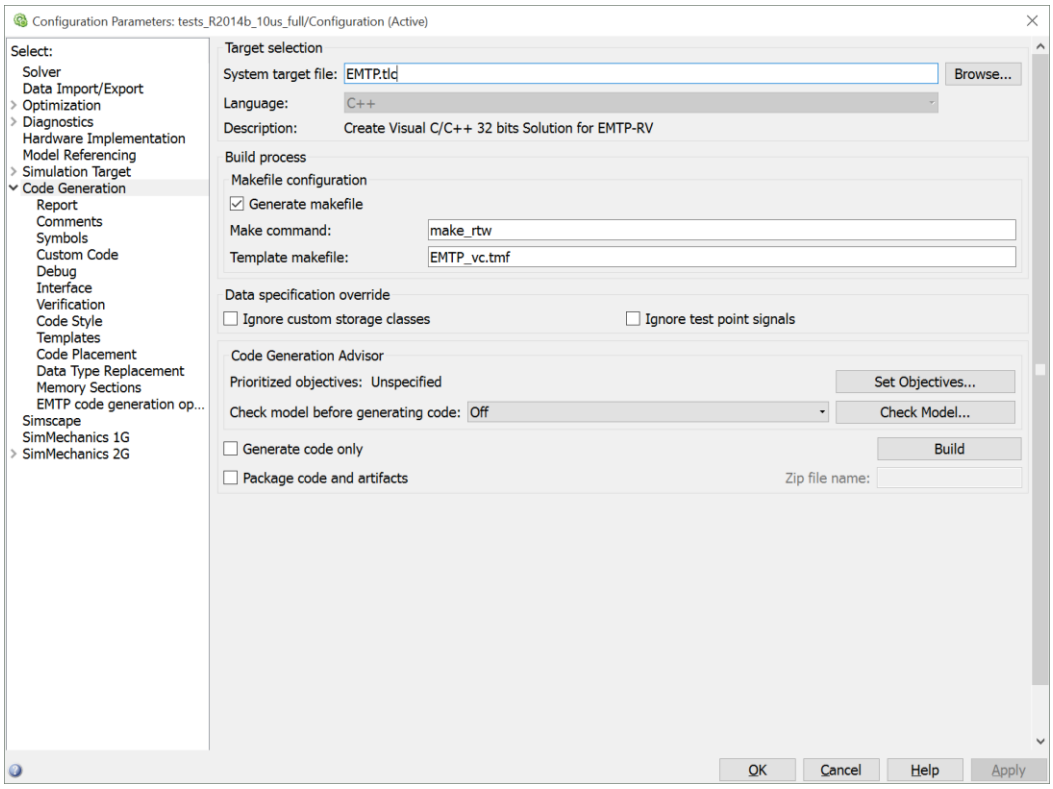


Figure 2 Simulink Model Configuration Parameters, Code Generation.

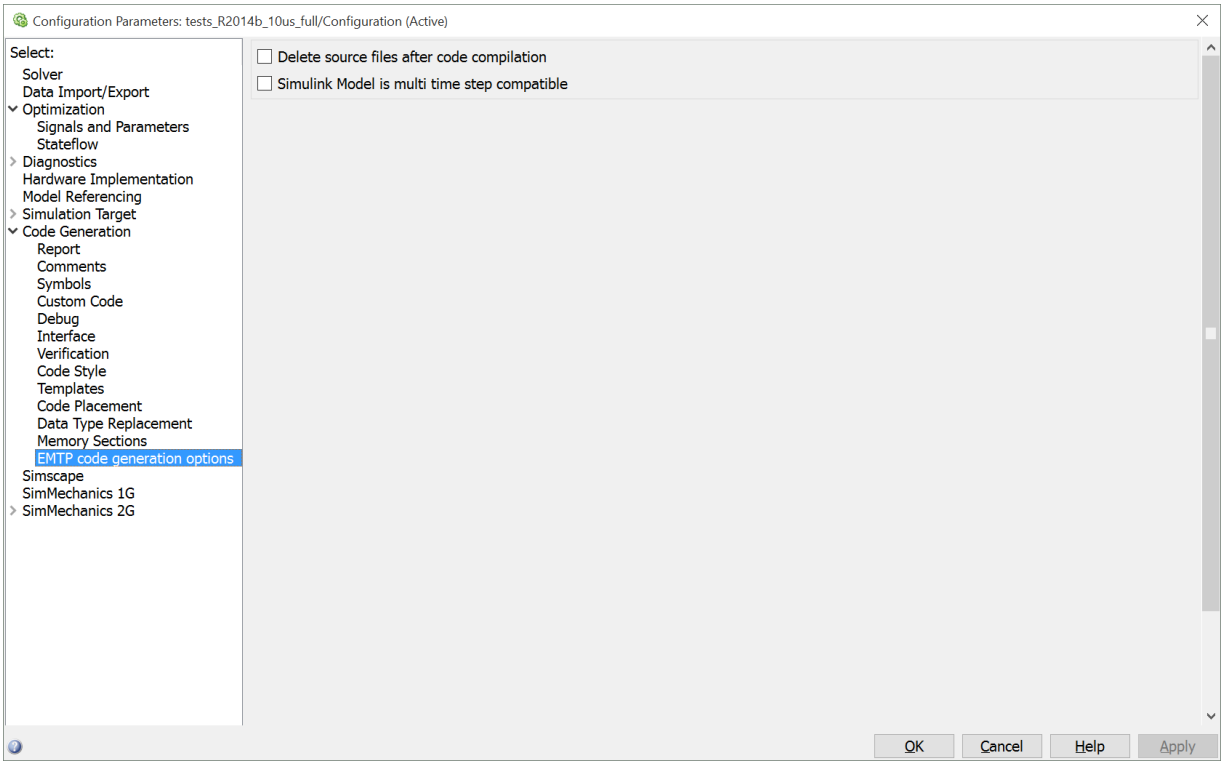


Figure 3 Simulink Model Configuration Parameters, EMTP code generation options.

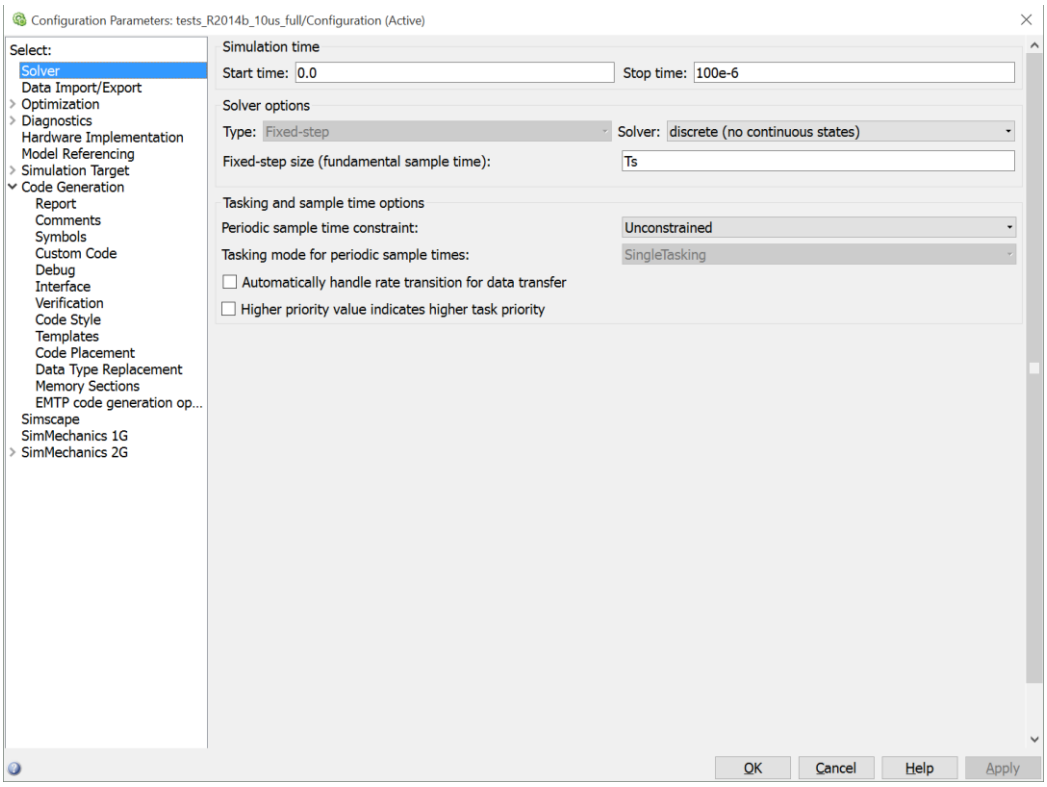


Figure 4 Simulink Model Configuration Parameters, Solver.

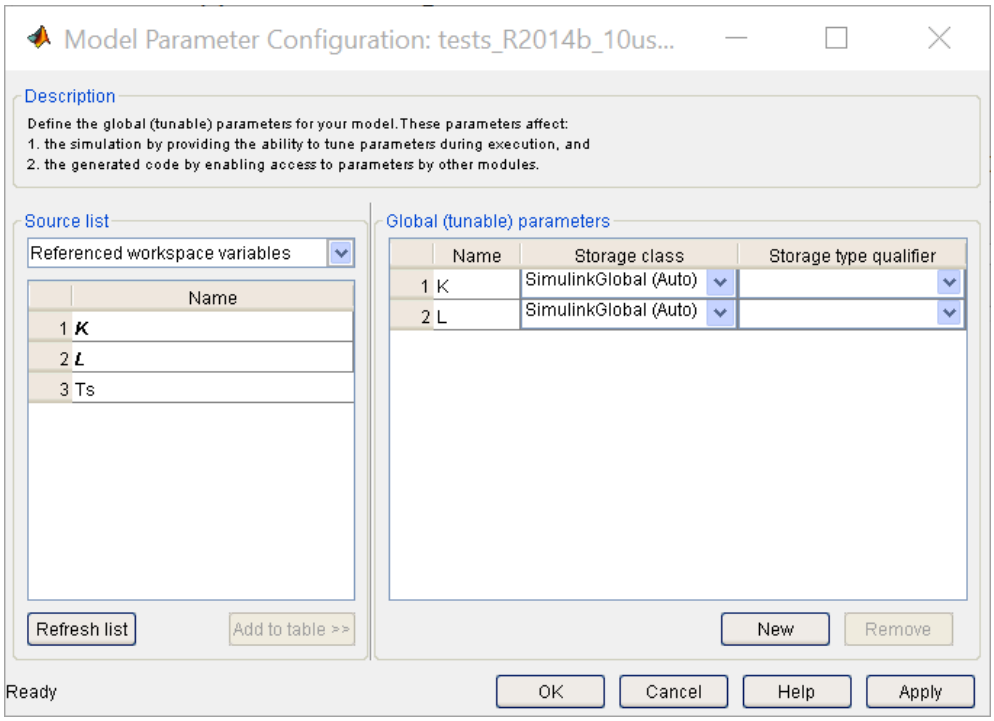
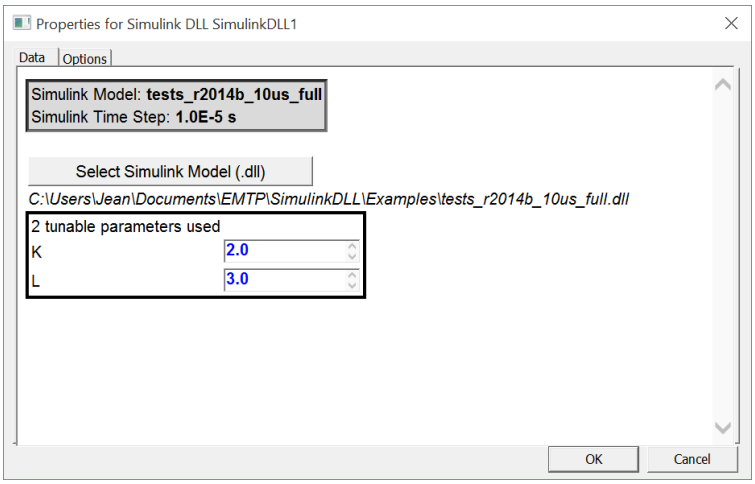


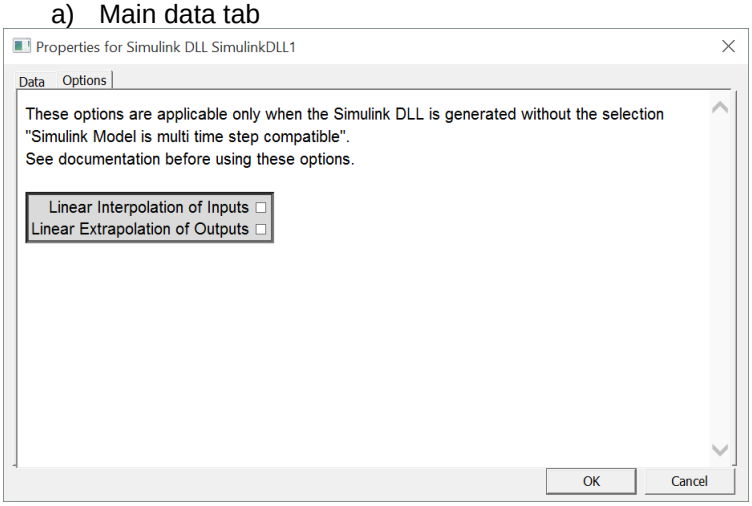
Figure 5 Simulink Model Configuration Parameters, Tunable parameters option.

3.2.3 Importing the model into EMTPWorks

Once the previous DLL code generation steps have been completed, it is possible to import (interface) the DLL into EMTPWorks. The user must drag in the device named “Simulink DLL” into the design and double-click on it for selecting the Simulink model. The “Select Simulink Model” button (see Figure 6) is used to browse and select the DLL file ModelName.dll generated in the above steps. In the example of Figure 1 (without the MATLAB function and the shown S-Function) the ModelName.dll is actually tests_R2014b_10us_full.dll.
Once the DLL selection is completed, the Simulink model time-step and the tunable parameters (if any) are shown (see Figure 6) in the Data tab. The corresponding EMTP file EMTPSimulinkDLL.ecf can be found in the folder ApplicationDir\Toolboxes\SimulinkDLL\Examples\SimpleDemo.
The tunable parameters can become named-values and used in upper level subnetwork masks on EMTPWorks side, if needed.



a) Main data tab



b) Options

Figure 6 EMTPWorks device “Simulink DLL” data tabs.

When the OK button of Figure 6 is clicked, the EMTPWorks interface block (see Figure 7) is automatically drawn with all the appropriate input and output pins and pin types. These are all control (Input or Output) pins.

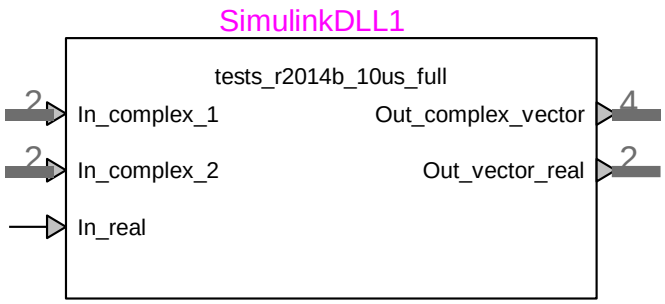
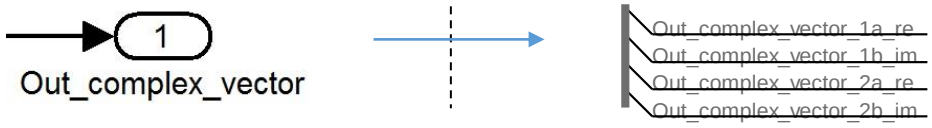


Figure 7 EMTPWorks interface block with Simulink DLL.

The interface block of Figure 7 is related to the selected version of the Simulink DLL. If the DLL is recompiled (regenerated) the interface will not be automatically updated with changes in interfacing pins, time-step and tunable parameters on Simulink side.
It is apparent from Figure 7 that vector type ports (pins) on Simulink side become bundle pins on EMTPWorks side. The number of pins in each bundle is show on the bundle pin.
It is noticed also that a complex signal on Simulink size is interpreted as a bundle on EMTWorks side. This is illustrated for the Out_complex_vector pin in Figure 8.



a) Simulink side b) EMTPWorks side

Figure 8 Transformation of a complex Simulink vector (see Figure 1) into EMTPWorks bundle.

3.2.4 Simulation results

It is expected that simulations results in EMTP will be delayed by the simulation time-step. At startup all Simulink DLL output signals are initialized to zero.

3.2.5 Rules and limitations

The complexity of Simulink may create various incompatibilities and issues for code generation and interfacing with EMTP. Here is a set of known rules that the user must follow.

1. The top level model (design) in Simulink must contain all Input and Output ports for interfacing with EMTP.
2. Tunable parameters
    a. A tunable parameter is a parameter that can be modified in Simulink during simulation.
    b. Tunable parameter settings are ignored when referenced models are used in Simulink.
    c. In some cases the tunable parameters may cause problems by losing tunability. In such cases MATLAB may not necessarily send an error message or the error message may be generic. See the following section for debugging such conditions.
3. S-Function
    a. There are two types of S-Functions in Simulink: standard S-functions based on C or C++ and inlined S-functions based on Simulink TLC files.
    b. To be able to use two or more Simulink models (DLLs) on EMTP side, it is necessary to use only inlined S-functions. See also MATLAB documentation “Write S-Functions that Support Code Reuse” for other issues, such as non-static variables in S-function codes.
    c. The multi-instance compatibility problems can be avoided by using two independent DLLs (different names) that can be generated using two different Simulink model file names for the exact same model.

4. Algebraic loops
    a. Simulink uses an iterative method for solving algebraic loops. A typical algebraic loop is shown in Figure 9.
    b. It is not possible in Simulink to generate code that can solve algebraic loops. In such cases, the user must artificially break the loops by inserting delays and validate that the block diagram maintains desired functionalities.

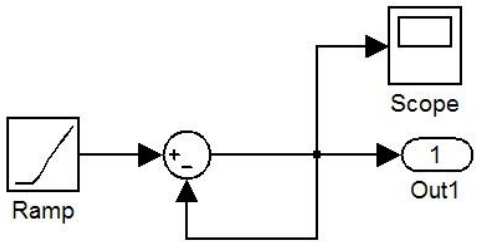


Figure 9 Algebraic loop condition in Simulink

3.2.6 Searching for tunable parameter problems on Simulink side

The safe approach for avoiding tunable parameter problems is to test them on Simulink side before establishing the interface with EMTP.

The user can search for all blocks using tunable parameters and test the tunable functionality by changing the parameter to observe its impact on simulation results. The “Inline parameters” option in “Optimization of Signals and Parameters” (see Figure 3) must be unchecked for performing the following tests.
If L is a tunable parameter, it can located in a model using the CTRL+F keyboard command. The window of Figure 10 with the selected configurations is used in the example of this document. The search results are listed at the bottom of the window. The user can open the dialog box of a listed block, make a change and validate that the parameter is tunable during the simulation by pressing Apply.

The user can search for blocks with masks and validate that tunable parameters are not used in the Initialization tab. A sample case is shown Figure 12. The variable L (see Figure 12a) is used to define the variable A in a subsystem mask (see Figure 12b) and the variable A is used for initialization in see Figure 12c. In this case the variable B in see Figure 12c is not tunable, which means that changing L will affect A, but not B!
In this example, if the initialization tabs are not using the variable A, the user must continue searching by replacing the searched variable L by the variable A since this variable has inherited tunability.

Further details can be found in MATLAB documentation (see Tunable Expression in Subsystem Mask Dialog Box).

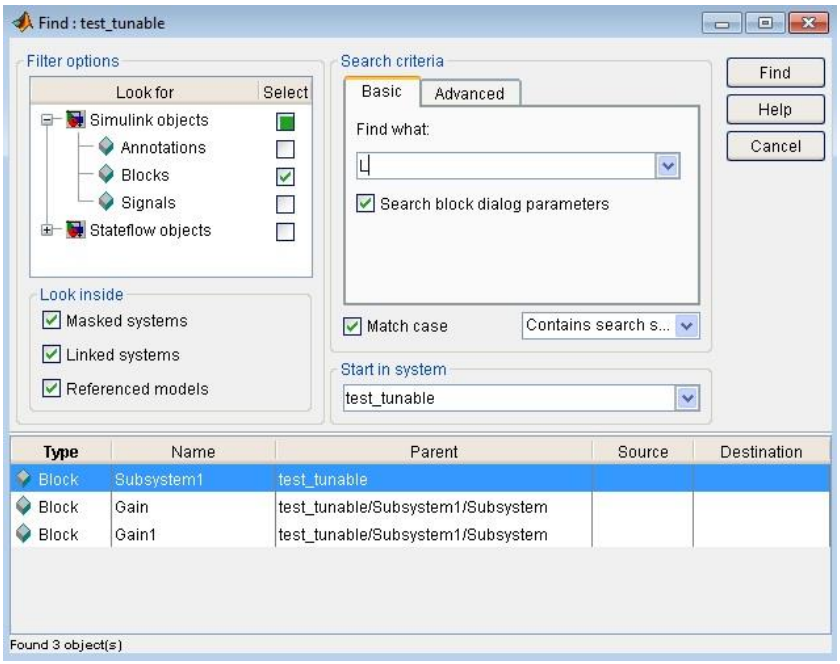


Figure 10 Search window for tunable parameter in Simulink.

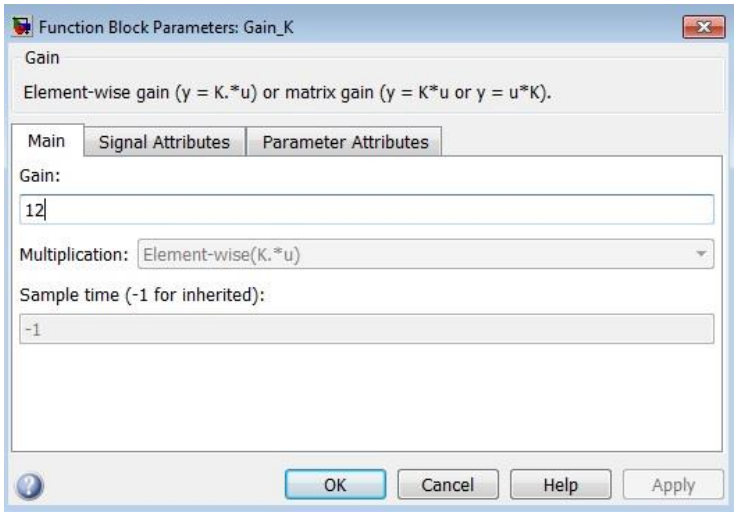
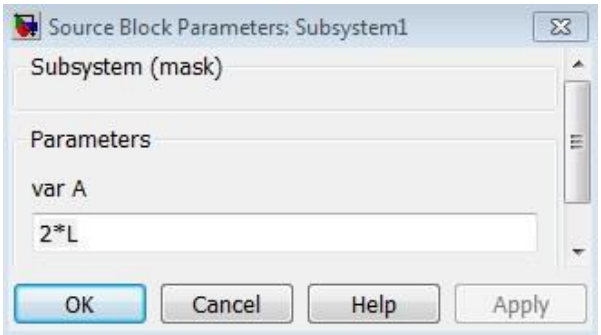
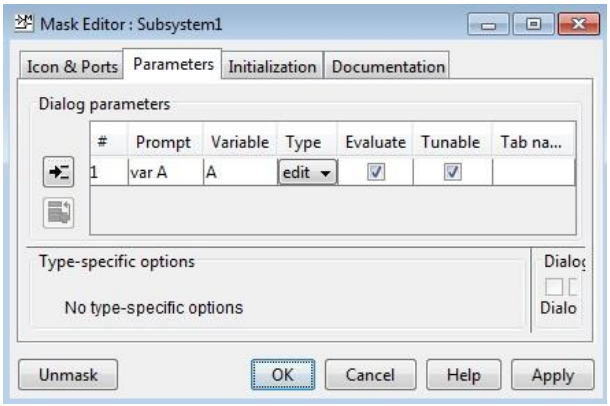


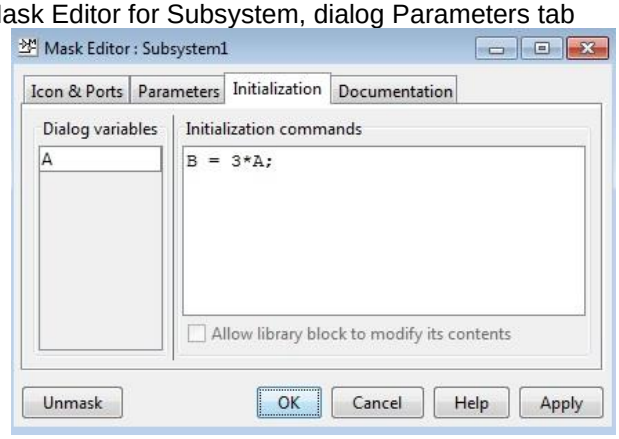
Figure 11 The Gain_K block, the tunable parameter can be replaced (see Gain field) and Applied.



a) Subsystem mask definition with tunable parameter L



b) Mask Editor for Subsystem, dialog Parameters tab



c) Mask Editor for Subsystem, Initialization tab

Figure 12 Simulink Mask with tunable parameter

4 Multi time-step compatibility

The EMTP numerical integration time-step is named hereinafter Δt_{EMTP} and the one used in Simulink is referred to as $\Delta t_{Simulink}$.

It is not recommended to have $\Delta t_{EMTP} \neq \Delta t_{Simulink}$, but the setup allows to handle such situations and may provide acceptable results in some cases.

4.1 Simulink model is multi time step compatible

When the option “Simulink Model is multi time step compatible” in Figure 3 is checked:

1. The Simulink DLL is called at all solution time-points in EMTP.
    a. It includes calls at intermediate Backward Euler time-points in EMTP
    b. It includes calls at all “Simultaneous switching” solutions when re-solving control systems as specified.
2. Such a selection means that the Simulink model is time-step independent and can provide accurate response at every call.
3. The selections on Interpolation and Extrapolation shown in Figure 6b are not applicable.

4.2 Simulink model is not multi time step compatible

When the option “Simulink Model is multi time step compatible” in Figure 3 is not checked, then the Simulink model will be called for any Δt_{EMTP} selection considering the following conditions.

1. It is strongly recommended to maintain $\Delta t_{EMTP} = \Delta t_{Simulink}$ and recompile the Simulink model when Δt_{EMTP} changes or modify Δt_{EMTP} to accommodate $\Delta t_{Simulink}$.

2. The Simulink DLL is not called at intermediate Backward Euler time-points.
3. When $\Delta t_{EMTP} = \Delta t_{Simulink}$ there is no applicable Interpolation and Extrapolation option.

Only when $\Delta t_{EMTP} \neq \Delta t_{Simulink}$ then it is possible to use the Interpolation and Extrapolation options selectable in Figure 6b. This must be done with caution and sufficiently good accuracy may not be achieved.

5 Compatibility with EMTP simulation options

The “Statistical Options” are compatible with the Simulink DLL.

When the “Simultaneous switching” option is used, it may request to “Re-solve control system equations”. This means that the control system equations and the Simulink DLL are called more than one times at the same simulation time-point. The Simulink DLL is called only once (first solution time-point) except when the “Simulink Model is multi time step compatible” option in Figure 3 is checked.

There is no steady-state solution with a Simulink DLL.