Productivity improvement tool for configuration of Modelica plant models and integration with Simulink controller models

Emerson Jacob Jeganathan, Anand Pitchaikani, Elavarasan Dharumaseelan
LMS Emmeskay Solutions Private Limited
20, Kannadasan Salai, T. Nagar, Chennai – 600017, INDIA
{Emerson.Jacob, Anand.Pitchaikani, Elavarasan.Dharumaseelan}@lmsintl.com

Abstract

Engineers practicing model based system design in an automotive sub-system supplier often use Modelica for modeling physical plant models and MATLAB/Simulink® for modeling the controller. Model-in-loop (MIL) simulations are performed using the S-function generated from the chosen Modelica plant model, integrated with the appropriate controller model and then simulated in Simulink. These steps are carried out by the engineer manually for the many different plant-controller configurations available in the organization. This repetitive workflow provides significant opportunities to streamline and automate the model based development process and improve productivity.

This paper presents an in-house MATLAB® GUI tool that can be used to configure the plant, select the controller, automatically generate an integrated model with the plant and controller, and simulate the resulting model. The plant model configuration information is passed on to Dymola® (the simulation environment) using the available communication (COM or DDE) to generate the plant model S-function. This tool includes post processing capabilities such as plotting the simulation results and custom plotting of metrics that are generated post-simulation.

Keywords: Automation; MIL; model configuration; simulation management

1 Introduction

Model-in-loop (MIL) simulation in non-real time environments is used to perform verification and validation testing of the controller model by the simulation engineer or test engineer, who will be referred to as "user" throughout this paper. Significant amounts of time need to be spent on configuring the plant models appropriately in Dymola and then combining them with matching controller models in MATLAB®.

Automation of plant model configuration reduces the burden on the user who would otherwise need to spend more time understanding and configuring the plant models. The tool discussed in this paper is used with a vehicle air-conditioning system model and its controller. The development of the plant models for both vehicle and air-conditioning system is extensively described in the previous work [1]. The way in which the plant model is packaged in Modelica helps the tool in configuring the models easily and automatically.

The tool couples two different modeling platforms and, as a result, a communication channel has to be established between them to exchange configuration information. The use of DDE Server as a communication protocol for data exchange and simulation between two simulation applications (MATLAB and Dymola) was described in [2]. The various pros and cons of using various communication protocols (DDE Server, TCP/IP) for communication between MATLAB and Dymola were discussed in depth in [3].

The in-house tool that is described in this paper has communication needs only while configuring the models and not during the simulation stage. Hence DDE Server communication offers a good solution [3] for data exchange between MATLAB and Dymola.

2 Tools

The in-house tool was developed in MATLAB® R2006b version as a graphical user interface (GUI). The plant models were developed using the Modelica language version 2.2.1. The data choices for the plant models (parameters) are made available as Microsoft® Office Excel spreadsheets for ease of use. The Dymola®-Simulink® interface is used to create the S-function of the developed plant models by the

MATLAB GUI Tool. The controller S-function and plant model S-function are integrated by the MATLAB GUI Tool.

3 Plant and Controller Models

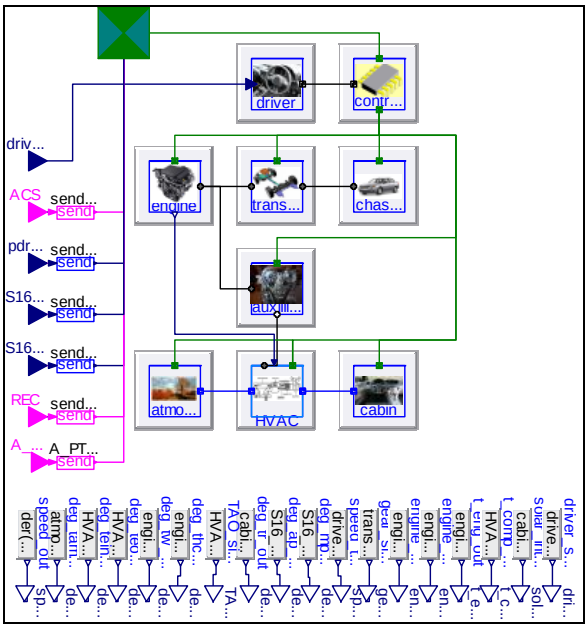


Figure 1: Plant model Architecture

The plant model architecture shown in Figure 1 consisting of the vehicle subsystems (driver, controller, engine, transmission and chassis) and the vehicle air-conditioning subsystems (compressor drive-pad, HVAC systems, cabin and atmosphere) in Modelica is made with all the main subsystems being replaceable. The variants of a particular subsystem are made by extending the same interface.

This scheme provides flexibility to choose different implementations for each subsystem as only their interface models are instantiated as replaceable in the architecture. Thus the same architecture model can be configured to represent different vehicle platforms easily.

The climate system controller model was available as a Simulink S-function. The controller exercises its control action over various components of the HVAC system to maintain the cabin at a set temperature. The controller receives sensor signals from various plant systems as feedback inputs.

A successfully integrated simulation of the developed plant models with the controller model is the main goal of the tool developed. An integrated plant and controller model is shown in Figure 2.

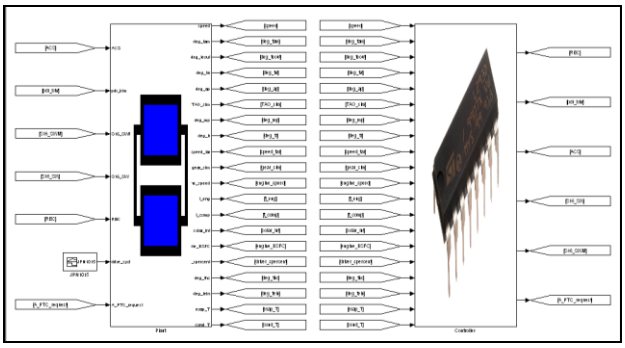


Figure 2: Integrated plant and controller model

4 Manual Process

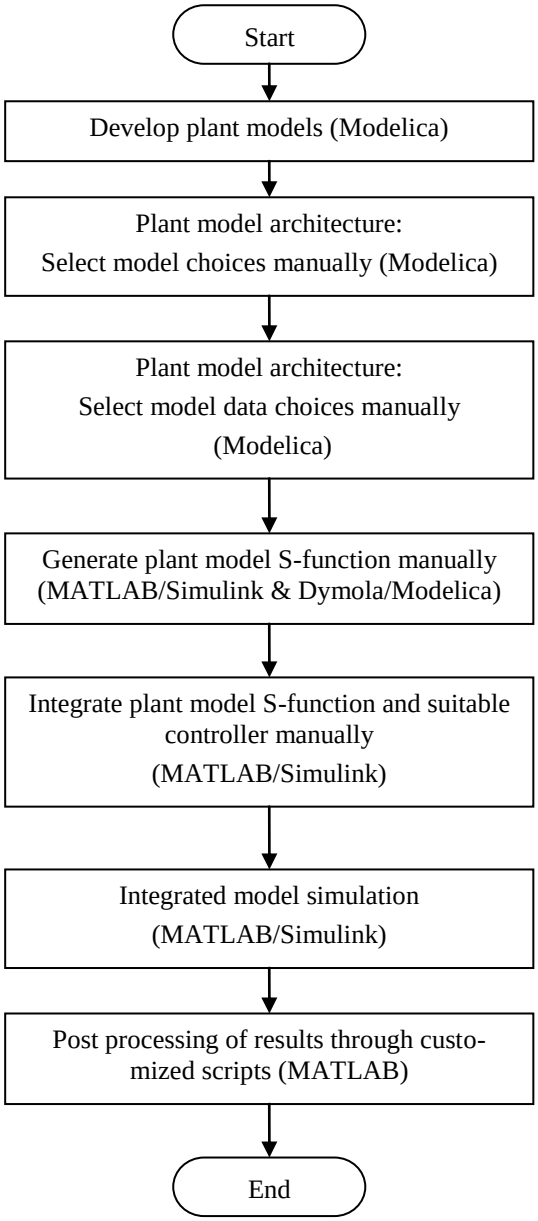


Figure 3: Flow chart - Manual process

The existing process involves configuration of the system model by applying the model choices and data choices for all top-level subsystems in a Modelica code layer by the user manually. Then the S-function is generated for the plant model using Dymola. This S-function is used in MATLAB/Simulink environment along with the controller model. The plant S-function and controller model are integrated manually and simulated in Simulink. Post-processing is done through customized m-scripts for plotting the simulation results. This process flow is depicted in a flow chart in Figure 3.

5 Modeling Guidelines

The power of object-oriented, acausal modeling language Modelica is utilized to the fullest in developing the plant models. The plant model consists primarily of components from the Modelica Standard Library (MSL). Some special needs were satisfied by creating additional custom models in Modelica as explained in [3], [4] and [5].

The plant models used in this work were done as a separate package i.e., library following certain guidelines so that the models can be interpreted and used in an easy and efficient manner by both the MATLAB GUI tool as well as the plant engineer who develops / updates the Modelica plant model.

All the top level subsystems are abstracted and individual interface models that are partial models are made. The complete plant subsystem model is created by extending the respective interface model (5 speed AT as well as 6 speed AT are extended from base class “Transmission”) and filled in with system equations. The package structure of the Modelica plant models is shown in Figure 4 and is discussed in more detail below.

5.1 Interfaces

Interfaces are partial Modelica models that form the base models for components and implementations containing just the interface details. An Interface contains the ports which interact with other models and the icon for identification.

For example, “Transmission” interface will have only two mechanical ports i.e., one for connecting to the engine and the other to the wheels. The interfaces also contain bus connectors to provide interface to the controller model.

5.2 Components

Apart from the Modelica Standard Library (MSL) components for this HVAC system simulation, the special needs were satisfied by creating additional custom models in Modelica. These are used as constituents of the top-level subsystem models.

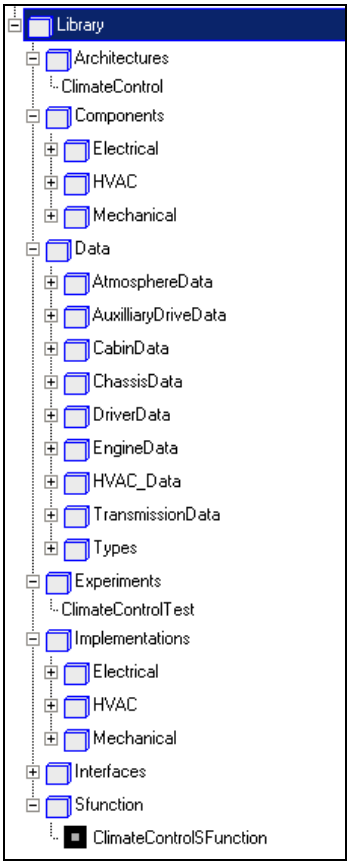


Figure 4: Library package

5.3 Implementations

Implementations are the variants of each subsystem model, which were developed extending the “interfaces” employing the “Components” described above and from the MSL library. These are complete models with respect to equations that represent the actual system.

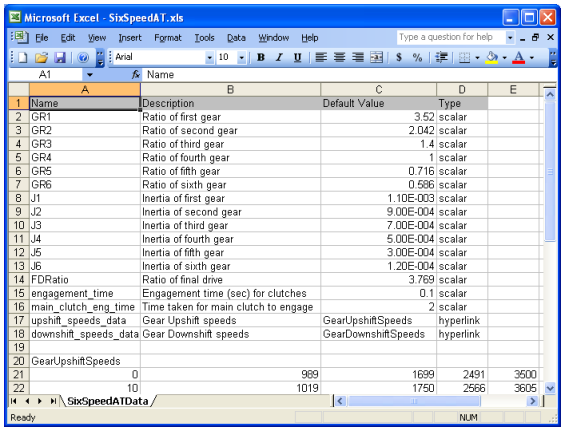
For example “Six speed AT”, “Five speed AT”, “CVT” were variants of transmission i.e., created by extending the “Transmission” interface model. This package is parsed by the SysInit GUI tool shown in Figure 7, to populate the list of various variants available for a particular model.

The GUI bins all the implementations based on the interfaces from which they are extended. As long as new implementations are added to the package following the structure explained above, the tool will automatically add the new variant to the list of choices for that sub-system.

5.4 Data

The data for each model - the set of parameters required by the "Implementations" and "Components" is implemented using the "record" class in Modelica. For example a "Six speed transmission data" record has the gear ratios and engagement time for a six speed transmission.

To facilitate data handling by SysInit GUI tool shown in Figure 7, separate Excel sheets are created in line with the records in Modelica to allow the user to quickly edit/create the data parameters for a particular model. A sample Excel data sheet ("Six Speed AT Data") for a "Six speed AT" model is shown in Figure 5.



	A	B	C	D	E
1	Name	Description	Default Value	Type	
2	GR1	Ratio of first gear	3.52	scalar	
3	GR2	Ratio of second gear	2.042	scalar	
4	GR3	Ratio of third gear	1.4	scalar	
5	GR4	Ratio of fourth gear	1	scalar	
6	GR5	Ratio of fifth gear	0.716	scalar	
7	GR6	Ratio of sixth gear	0.586	scalar	
8	J1	Inertia of first gear	1.10E-003	scalar	
9	J2	Inertia of second gear	9.00E-004	scalar	
10	J3	Inertia of third gear	7.00E-004	scalar	
11	J4	Inertia of fourth gear	5.00E-004	scalar	
12	J5	Inertia of fifth gear	3.00E-004	scalar	
13	J6	Inertia of sixth gear	1.20E-004	scalar	
14	FDRatio	Ratio of final drive	3.769	scalar	
15	engagement_time	Engagement time (sec) for clutches	0.1	scalar	
16	main_clutch_eng_time	Time taken for main clutch to engage	2	scalar	
17	upshift_speeds_data	Gear Upshift speeds	GearUpshiftSpeeds	hyperlink	
18	downshift_speeds_data	Gear Downshift speeds	GearDownshiftSpeeds	hyperlink	
19					
20	GearUpshiftSpeeds				
21	0	989	1699	2491	3500
22	10	1019	1750	2566	3605

Figure 5: Excel data sheet

The tool covert the data in the Excel sheet into appropriate record when writing the S-function model. Consistency in data set has to be ensured between Modelica sub-system model choice's base record and the parameter excel file, which is uniquely identified by having its file name same as the model choice name.

5.5 Architecture

Architecture is the representative assembly of all top-level subsystem partial models with connections between them (conventional vehicle, parallel hybrid vehicle etc.). This has the interconnection of the interface models to create the skeleton of the overall vehicle system i.e., driver, engine, transmission, HVAC, cabin, etc., are connected to form the vehicle system.

The overall plant model architecture (Figure 1) in Modelica is made such that all the main subsystems are replaceable. This architecture provides flexibility to choose different implementations for each subsystem.

In this way, the same model can be configured to model different vehicle systems easily. The tool can use this architecture model as selected by the user as the base model on top of which it applies the model choices and data choices.

5.6 S-function

An "architecture" in which the sub-systems are completely defined (by re-declaring the interface models i.e., "Interfaces" with complete models i.e., "Implementations") with the chosen model and data choices is the S-function model. The model and data choices can be reconfigured by re-declaring them. This model is written by the GUI tool and sent to Dymola for S-function generation.

An example of a Modelica model that would be compiled into an S-function is given below:

```
model ClimateControlSFunction
  "Plant S-function"
  extends Architechture.ClimateControl(
      redeclare Implementations.Mechanical.SixSpeedAT transmission (redeclare Data.TransmissionData.SixSpeedATData transmission_data));
  ...
end ClimateControlSFunction;
```

6 GUI Tool Functionalities

6.1 High level configuration

The top-level GUI which is shown in Figure 6 has various fields for selection of plant system configuration (e.g. Class A Vehicle), controller (Class A Vehicle Controller), drive cycle and fields to indicate the location of the simulation results file and summary file to be written.

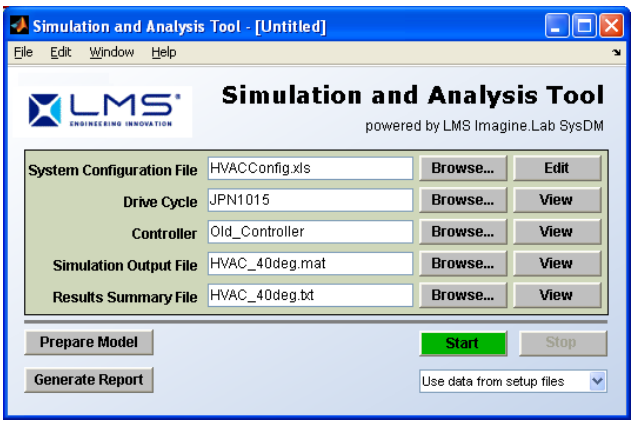


Figure 6: Top-level GUI

Using the "Browse" and "View" buttons the various choices available can be either selected or viewed respectively. The plant and controller models can be integrated and simulated using "Prepare" and "Start/Stop" buttons, while other buttons and menus provide additional functionalities such as report generation, saving the system configuration (plant and controller combination) etc.

The plant model configuration (which is stored as an Excel file) is managed by another GUI (called the SysInit GUI), which configures the plant model with appropriate sub-system model and data choices. This GUI is invoked by the "Edit" button corresponding to the System Configuration File field in the above GUI. The functionality of SysInit GUI is explained in the next section.

6.2 Plant configuration

The vehicle plant model is configured using the SysInit GUI shown in Figure 7. The SysInit GUI has three columns namely Models, Model choices and Data choices.

For each of the top-level vehicle subsystem model (say transmission), the model choices available for that model (5 speed AT, 6 speed AT, CVT etc) are displayed in the middle column and the data choices (Excel data sheets which have parameters like inertia, efficiency, gear ratio) for the chosen model choice are displayed in the last column as drop down menus.

These drop-down menus are automatically populated with model and data choices from the Modelica plant model package as explained in the best practices section.

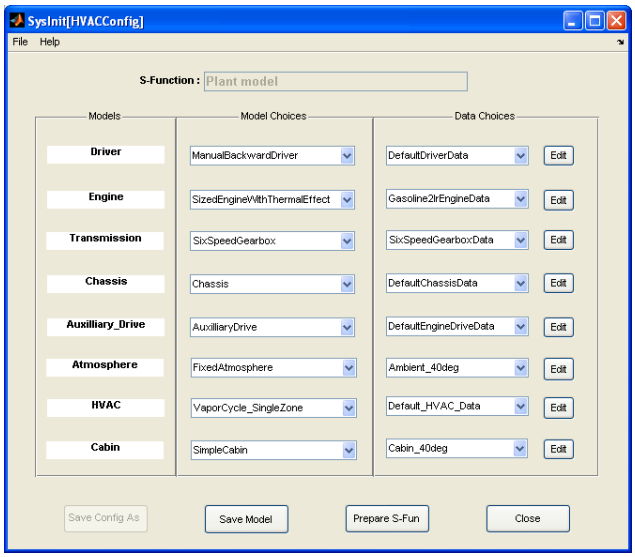


Figure 7: SysInit GUI Tool

The data for populating the model choices in the SysInit GUI is obtained by parsing the Modelica subsystem models that are developed following the "Best Practices" described previously in this paper. The data choices are obtained by parsing the Excel files which are in correspondence with the model choices.

The model and data choice displayed by default are loaded from the system configuration Excel file and can be changed as per the user's needs. The changed configuration can once again be saved as an Excel spreadsheet with appropriate name. Once the desired choices are made for the selected plant model, a Simulink S-function can be generated using the "Prepare S-Fun" button.

The plant model configuration information is written as a Modelica model file using the architecture model specified by the user. This model is passed on to Dymola by establishing a DDE (Dynamic Data Exchange) server communication between MATLAB and Dymola to generate the plant model S-function. This simplified process enables the user to quickly create variants of plant models via different configurations of the sub-system models and parameter data choices.

The plant model Modelica file (.mo file), which may be useful for debugging can be generated using "Save Model" button, while that particular chosen system configuration can be saved to an Excel file for retrieval at a later time.

6.3 Integration with controller

After the S-function is generated for a particular plant model configuration, the top-level GUI tool integrates the chosen controller and drive cycle for simulation with the plant S-function based on matching input and output signals names in them. The drive cycle and controller library are shown in Figure 8. An integrated plant and controller model is shown in Figure 2.

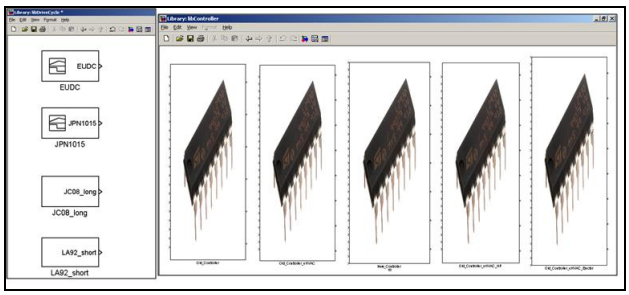


Figure 8: Drive cycle and Controller library

6.4 Simulation and post-processing

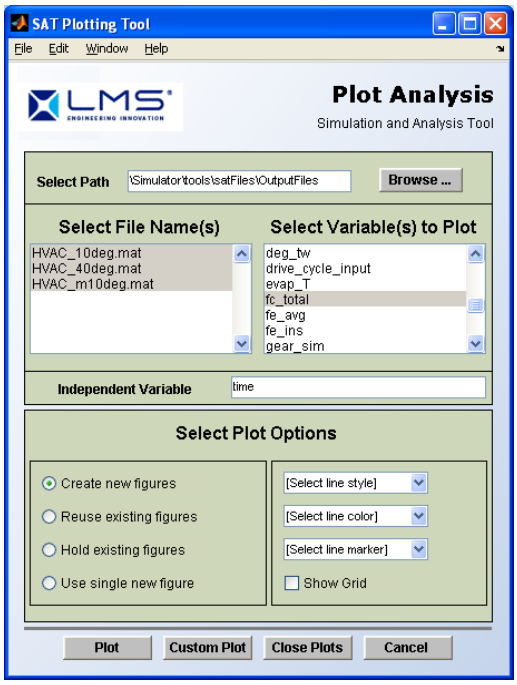


Figure 9: Generic Post Processing GUI

The integrated plant and controller model is simulated from the top-level GUI tool, which stores the result and summary in the specified files once simulation is done. Post processing of the results can be done by two plot GUI's which are shown in Figures 9 and 11.

The plot GUI shown in Figure 9 is a generic one capable of plotting same variables between multiple files and one such sample result is shown in Figure 10.

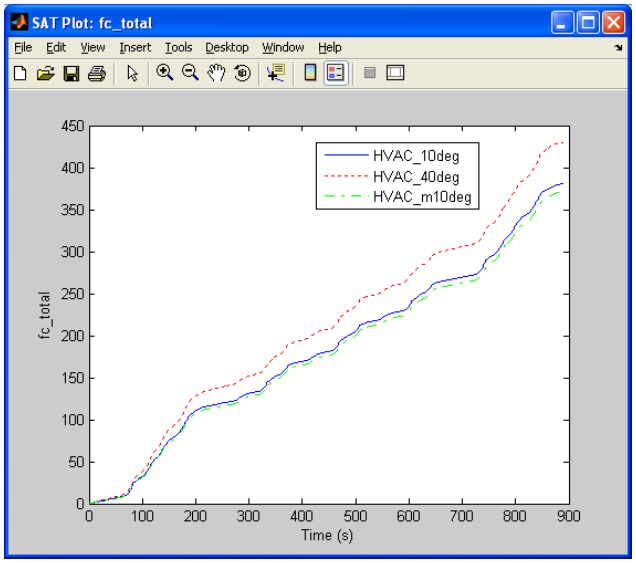


Figure 10: Sample Result (Generic Post Processing GUI)

The plot GUI shown in Figure 11, Custom Post Processing GUI is a specific one which can be used to get "metrics" i.e., some processing of results in different files to get a specific plot. This processing can be done through a MATLAB m-script and applied on the results to get the specific plot.

These post-processing methods can be quite useful for users, say in benchmarking different controllers with the same plant model.

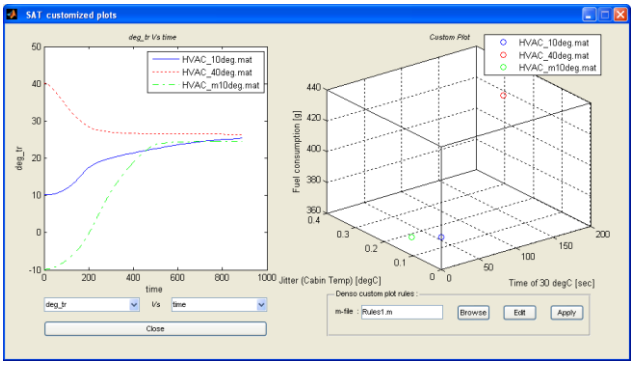


Figure 11: Custom Post Processing GUI

6.5 Flow chart

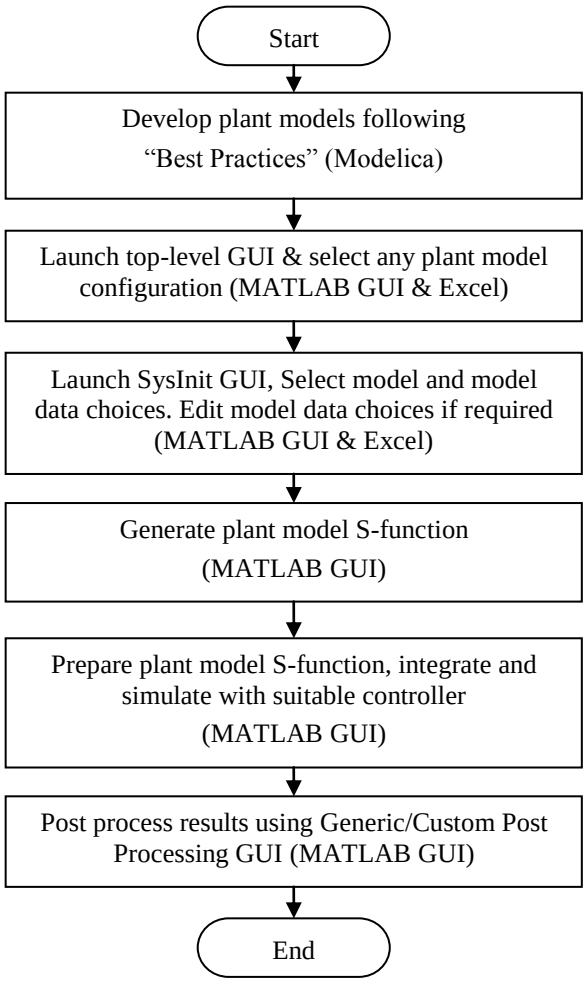


Figure 12: Flow chart – MATLAB GUI Tool process

The flow chart of the processes followed in the in-house developed MATLAB GUI Tool is shown in Figure 12.

7 Practical Implementation - Climate control

The Modelica plant models developed for usage with the MATLAB GUI Tool explained in this paper was related to Climate Controller testing using Modelica plant models [1]. Best practices explained previously were taken care while modeling the plant (vehicle with HVAC system). The climate system controller model was available as a Simulink S-function. The exercise of running the closed loop model establishes the proper compatibility achieved between the developed Modelica plant models with the chosen controller.

The closed loop simulation was very beneficial for validation of control strategies and functionality of control elements related to HVAC like heater, inlet door, etc., comparison of different control strategies and in fuel economy prediction of the vehicle.

8 Future Scope

This GUI tool is developed for a particular plant model architecture, which deals with vehicle with HVAC system simulation. This tool can be quickly reconfigured to work with any plant model architecture of interest for MIL or SIL simulations (Hybrid electric vehicle controller study, etc,). The scope of the tool can also be extended to include processor-in-loop (PIL) and hardware-in-loop (HIL) simulations.

9 Conclusions

This MATLAB based GUI tool enables the user in validating or benchmarking the controller models which are in MATLAB/Simulink against various configurations of plant models which are in Modelica for model-in-loop (MIL) simulations. The most important thing the tool achieves is that it takes the user away from the model development environment so that he remains more objective. The errors that the user can introduce while configuring the plant models in Modelica are eliminated but at a cost that the plant model engineer has to pay by sticking to the guidelines established. It was observed that the time taken by user to configure a new plant model and get the controller tested has drastically reduced by the use of this tool. The tool helps in reducing the interactions between the user and plant engineer with respect to the plant models which helps reduce the development time.

10 Acknowledgments

This work was supported by Dr. Yasunori Yokojima (LMS Japan) and Dr. Shiva Sivashankar (LMS North America). Authors wish to thank Mr. S. A. Sundaresan (LMS Emmeskay Solutions Private Limited, INDIA) for his guidance throughout this work. Authors wish to thank Mr. Bharani Shivakumar (LMS Emmeskay Solutions Private Limited, INDIA) for improving the tool considerably.

References

[1] Anand Pitchaikani et al, Real-time Drive Cycle Simulation of Automotive Climate Control System, pp. 839-846, Proceedings of the 7th International Modelica Conference, Como, Italy, 20-22 September 2009

[2] S.E. Pohl and J. Ungethüm, A Simulation Management Environment for Dymola, pp. 173-176, Proceedings of the 4th International Modelica Conference, Hamburg, 7-8 March, 2005.

[3] C. Schlegel, R. Finsterwalder, H. Olsson, Using Dymola generated C-Code in specialized Client/Server Simulation Environments, Not published, Proceedings of the 4th International Modelica Conference, Hamburg, March 7-8, 2005.

[4] M. Tiller, "Introduction to Physical Modeling with Modelica", Kluwer Academic Publishers, ISBN 0-7923-7367-7.

[5] Dymola. Dynamic Modeling Laboratory, Dynasim AB, Lund, Sweden, http://www.Dynasim.se.

[6] Mathworks, http://www.mathworks.com.

[7] Modelica, http://www.modelica.org.