MLE+: Tutorial

Willy Bernal Heredia - National Renewable Energy Laboratory

September 18, 2016

Contents

1 Introduction

MLE+ is an open-source Matlab/Simulink toolbox for building energy simulation, analysis, optimization and control design. At the core of MLE+ are co-simulation interfaces with multiple building energy simulation programs such as EnergyPlus. MLE+ provides a Simulink library with a S-Function block for easy drag-and-drop implementation. Figure 1 illustrates the overall structure of MLE+.

Note: This version of MLE+ does not provide a graphical interface.

./figs/overview.pdf

Figure 1: MLE+ interfaces control system toolboxes with building models and systems.

MLE+ is designed for engineers and researchers who are familiar with Matlab and Simulink and want to use these software tools in building energy research. MLE+ is particularly useful for:

1. **Simulation configuration:** The MLE+ front-end streamlines the configuration process of linking the building model and the controllers by abstracting the necessary parameters from the co-simulation. This

reduces setup time and configuration problems.

2. **Controller design:** MLE+ provides a control development workflow as well as graphical front-ends for designing advanced control strategies for buildings, in which the building simulation is carried out by dedicated simulation software, such as EnergyPlus, while the controllers are implemented in Matlab or Simulink.

3. **Simulation-based optimization:** MLE+ can be used to find optimal parameters or control sequences for building system designs, using simulation-based nonlinear optimization.

4. **Data analysis:** Simulation output data from EnergyPlus can be aggregated, analyzed and visualized in Matlab.

5. **Building Energy Management System Interface:** MLE+ provides a BACnet interface to develop and implement control methods for real building equipment.

6. **Matlab/Simulink environment:** MLE+ allows complete access to the Matlab environment and toolboxes such as Global Optimization Toolbox, System Identification Toolbox and Model Predictive Control Toolbox. The user can step through the code for debugging and pause the co-simulation at any time.

2 System Requirements

- Windows Operating System. Currently, MLE+ is only supported in Windows. However, we are working in making MLE+ compatible in the Linux and Mac OS platforms.

- MLE+ requires Matlab and/or Simulink of recent versions. MLE+ uses the GUI Layout Toolbox. This is included in the MLE+ distribution. MLE+ has been tested in Matlab 2011a and 2012a versions[1].

- Java must be enabled in Matlab. By default, Java is already enabled in Matlab, so no further action is required. The Java socket library is used by MLE+ for communication with EnergyPlus.

[1] The GUI Layout Toolbox requires 2010a or future version of Matlab. If you find any problems, please contact the authors for further assistance

- EnergyPlus version 7.1.0 (latest). MLE+ should work well with previous versions of EnergyPlus: versions 7.0.0 and 6.0.0. However, it has not been tested thoroughly. We strongly recommend to download EnergyPlus 7.1.0 as the example files correspond to this version.

3 Installation

1. Download the latest version from `https://github.com/mlab/mlep_v1.1/zipball/master` or clone the repository `https://github.com/mlab/mlep_v1.1`
2. Extract all files to a directory in your computer. Call this folder `/mlep`.
3. Open Matlab and change the current directory to the `/mlep/MLE+` folder that has just been created. In Matlab, run the installation script `installMep.m` and follow the instructions.

3.1 WINDOWS

1. This will install the GUI Layout Toolbox and add the necessary paths to the Matlab environment automatically. After that, the installation screen in Figure 2 will appear. Here you need to specify the paths to EnergyPlus main Directory and the path to the folder with Java binaries. Also, this will replace your `RunEPlus.batch` file (in Windows).

3.2 MAC

1. This will install the GUI Layout Toolbox and add the necessary paths to the Matlab environment automatically. After that, the installation screen in Figure 3 will appear. Here you need to specify the paths to EnergyPlus main Directory.
2. In MAC distributions, the orginal `runenergyplus` file in your EnergyPlus Distribution (e.g. `\Applications\EnergyPlus\runenergyplus`) makes sure the EnergyPlus result files are written to the `\Output` folder and `.mat` files are not deleted.

figs/installation.png

Figure 2: Windows MLE+ Installation Screen.

3.3 MAnual Installation

1. Depending on your Matlab distribution, you might not be able to use the GUILayout

tool required to open the MLE+ front-end. If you run into some problems installing try using the manual installation. You would need to follow this instructions .

4 Tutorial: Shading Control Example

In the following tutorial example, we will walk you through the steps to set up a co-simulation session with EnergyPlus from Matlab. We will then figs/installationMAC.pdf

Figure 3: MAC OS MLE+ Installation Screen.

design a controller in MLE+ for actuating the window blinds of a building simulated in EnergyPlus.

4.1 The Building

A single-storied building shown in Figure 4 consists of three zones with a total floor area of 130 $\{\mathrm{m}^2\}$ $\#130\,\mathrm{m}^2$. The West zone of the building consists of a large window equipped with blinds/shades and is subject to strong solar radiation during the day. The goal is to control the window shade deployment of the West zone such that the transmitted solar radiation (through the window) never exceeds a certain threshold. The window blinds can be controlled using two EnergyPlus variables:

- `Shading_Deployment_Status` controls whether the blinds are deployed or not;
- `ShadeAngle_Schedule` controls the slat angle so it is perpendicular to the incident solar radiation whenever the blinds are deployed.

`figs/buildingcad.jpg`

Figure 4: EnergyPlus window shading control model.

We will design a controller in MLE+ which monitors the angle and intensity of the solar radiation incident on the West zone window. If the incident solar radiation exceeds a certain threshold, the blinds will be deployed and the shade angle will be set to reduce the possibility of glare.

4.2 The MLE+ Control Design Workflow

The control design workflow of MLE+ defines a sequence of steps for designing a controller in Matlab for a building model simulated by EnergyPlus. A graphical front-end is provided to support this workflow. To start the front-end, execute the command `mlep` in Matlab. This will open a graphical interface as shown in Figure 5.

figs/start.png

Figure 5: Graphical front-end for the MLE+ control design workflow.

4.3 Set Up EnergyPlus Simulation Model

First, we need to specify the EnergyPlus building model and the weather profile to be used for simulation (Figure 5).

- Click the button **Select IDF file** and select the file `EMSWindowShadeControl.idf` located in the folder `/ShadingProject`.
- Click the button **Select weather file** and select the weather file `USA_IL_Chicago-OHare.Intl.AP.725300_TMY3.epw`. We will use the weather profile of Chicago for our simulation.

4.4 Configure Input and Output Variables Between EnergyPlus and Matlab

We will set up the input and output variables to be exchanged between EnergyPlus and Matlab for co-simulation. An input variable serves as an input to EnergyPlus at each step of the co-simulation, while output variables are those which can be repeatedly read from EnergyPlus to monitor its internal state.

1. Select the Control Tab (Figure 6)
2. In the Control Tab, push the **Variable** button to open the Variable Configuration Window (Figure 7).
3. Load the `.idf` file by pushing the **Load IDF** button. This will list the available `ExternalInterface:Schedule`, `ExternalInterface:Actuator` and `ExternalInterface:Variable` objects from the idf file. It will also list the available `Output:Variable` objects.
4. Add the necessary inputs and outputs to have the settings specified in (Figure 8) and (Figure 9), respectively. In this example, we specify `Shading_Deployment_Status` and `ShadeAngle_Schedule` as the inputs to EnergyPlus as these are the variables that we will control through MLE+. Make sure your configuration is exactly the same as the one shown in (Figure 8) and (Figure 9).
5. Once the input and output variables had been set, push the green button **Write Variables.cfg**. This file will create a file with the communication configuration between Matlab and EnergyPlus. It should be printed in the Matlab command line.
6. Close the Variable Configuration Window. Either click on the **Close Screen** or the **X**.

In MLE+, an alias can be specified for each of the variables (Figure 8 and Figure 9). The alias allows the user to reference a variable with a more figs/control.png

Figure 6: MLE+ control design tab.

intuitive name and avoid the intricate names specified by EnergyPlus. For instance, the EnergyPlus variable `Zn001_Wall001_Win001_Shading_Deployment_Status` can be assigned a more intuitive name as `ShadeStatus`.

4.5 Design a Shading Controller

In the control tab, we will specify the building controller, implemented in Matlab, for our building model (Figure 10).

1. Push the button **Load Control File** and select the file `control_file_blind_angle.m`. This file contains the Matlab code for the shading controller.

2. View and edit this file by clicking the button **Edit Control File.** You can also create a template file for your own feedback loop by clicking

figs/variables.png

Figure 7: Variable configuration window.

on **Create Control File.** This creates the file `controlFile.m`.

The input and output variables specified by the user are referred to by their aliases throughout the control file as shown in Figure 11. In the code snippet shown in Figure 11 the value of the incident solar radiation is compared against the threshold ($100\,\mathrm{W\,m^{-2}}$) to determine if the shades will be deployed.

4.6 Simulation and Assessment

Once a control design has been completed, we can run the simulation or step through it using the Matlab debugging environment.

1. Click on the tab **Simulate** then click on button **Run Simulation**

`figs/variableInput.png`

Figure 8: Configuration of input variables to EnergyPlus.

This will call EnergyPlus to run the building energy simulation with the parameters we have specified.

2. A Windows command window will open and will show the progress of the simulation.

3. After the co-simulation has finished, MLE+ extracts and parses all output variables generated by EnergyPlus, then lists them in a listbox (see Figure 12). Select one or multiple variables, then click the button **Plot** to display them on the screen.

4. You can also save the data to the Matlab workspace by clicking the buttons **Save all** or **Save Selected**

figs/variableOutput.png

Figure 9: Configuration of output variables to EnergyPlus.

5. The building geometry is visualized in tab **Building.**

Note that MLE+ decouples the simulation engine and the controller implementation. This way we can tune the control scheme in Matlab, then assess its performance by running multiple simulations without the need of modifying the EnergyPlus file.

4.7 Load, Save and Reset Project Data

At the bottom of the window, you can find buttons to load a control design project from a file, save a project to a file, and reset the current project data. A project file has the extension `.prj` and contains all essential information of a control design project.

figs/control1.png

Figure 10: MLE+ control design tab.

1. **Load Project:** open previously saved projects.
2. **Save Project:** save all the configuration settings which have been entered so far to a file. Note that this does not save your controller file, or your `.idf` file.
3. **Reset Project:** empty all fields in the graphical front-end. Note that this will not erase the current project file, but only reset the configuration settings in the graphical front-end.
4. **Exit:** exit the program.

For your convenience, a project file for the tutorial example is included in the distribution. You can load the project file `ShadingProject.prj` and figs/code.pdf

Figure 11: Matlab code snippet of the shading controller (notice alias variables).

switch directly to the tab **Simulate** to run a simulation of the control design.

5 Other Examples

5.1 Legacy Example

This folder contains the original example distributed with MLE+ Legacy. This example does not make use of the MLE+ front end. You can run this example by executing runsimple.m in Matlab. This example sets the Temperature Setpoints for a small building.

figs/simulate.png

Figure 12: Plot and analyze simulation results of EnergyPlus.

5.2 Green Scheduling vs. Uncoordinated Control

Here we compared two different binary (ON/OFF) controls for keeping the temperature of a small building inside the comfort level. Green Scheduling is a control scheme designed to reduce peak power consumption while satisfying the temperature conditions. You can load these projects by using the **Load Project** button.

5.3 MPC vs. Proportional Control

These two cases implement continuous control schemes. The first control is a Model Predictive controller using built-in functions in Matlab. This is compared against a very simple proportional feedback loop. The model for the first strategy was generated using the **System Identification tab** in MLE+. This tab allows you to design the disturbances you feed your model for SYSID. Then, you can import this model directly into the Matlab's built-in system identification toolbox.