**PSM V1.0.0**

**USER GUIDE**

Copyright (c) 2017-2018, RTE (<http://www.rte-france.com>). This Document is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

**INDEX**

1.- Definitions 3

2.- PSM Overview 4

3.- Getting started 5

3.1.- Installation requirements 5

3.2.- How to install PSM 5

3.3.- How to run PSM 6

4.- PSM Workflow 6

4.1.- CIM importer 7

4.2.- Loadflow computation 8

4.3.- Modelica file generation 8

4.4.- Dynamic simulation definition 9

4.5.- Time-domain simulation 9

4.6.- User interface 9

5.- Tutorial 10

5.1.- Example 1 10

5.2.- Example 2 18

6.- User interface reference 26

6.1.- Cases section 26

6.2.- DDRs section 27

6.3.- Conversion section 28

6.4.- Simulation section 30

6.5.- Loadflow section 32

6.6.- Validation section 33

7.- Configuration files 35

8.- Command line reference 39

9.- Dynamic Data reference 45

9.1.- Modelica files 45

9.2.- Dynamic Data Repository XML files 46

10.- iPSL Documentation 48

# 1.- Definitions

|  |  |
| --- | --- |
| System | The electric network being modelled. |
| Static system model | Model of the system that only takes into account its steady-state behaviour. The source for this model is CIM[[1]](#footnote-1). |
| Dynamic system model | Model of the system that takes into account dynamic aspects of the system and allows time domain simulation. The source for this model is Modelica[[2]](#footnote-2). |
| Conversion | The process that builds the dynamic system model from the static system model plus additional dynamic modelling information. |
| Element | A piece of electric equipment of the system.  It is defined as an object in the static system model: bus, generator, load, shunt, transmission line, or transformer.  Can be defined as a single model or a set of combined models in the dynamic system model. |
| Case | A steady state of the system. Given by power system state variables and/or steady state hypothesis information, as well as topology information. |
| Event | A model or combination of models in the dynamic system model describing a particular event (fault, value modification, element disconnection, etc.) taking place at a specific time and for a given time duration. |
| Full model initialization | A process that allows obtaining required values for dynamic system simulation from multiple separate dynamic simulations of individual elements. The produced values could be internal parameter derivation or initial values for individual models needed for the system initialization. |

# 2.- PSM Overview

PSM (Power Systems on Modelica) is a software tool that performs the conversion from a static system model received as a set of CIM files to a dynamic system model expressed in the Modelica language, as an automated transformation process, using dynamic data given in a dynamic data repository. The resulting dynamic system model can be used to perform time-domain simulations and study the response to events.

The static system model contains all the objects corresponding to elements of the power system (buses, generators, loads, shunts, transmission lines, transformers, etc.) and how they are linked (also know as topology). The case data contains the demand, power injections and actual topology that correspond to a particular (steady) state of the system. The goal of the automated conversion is to obtain a dynamic system model that will allow users to perform time-domain simulations on the system. To achieve this goal, PSM relies on two fundamental external tools:

* **iPST**: iTesla Power Systems Tools, available as an open source project[[3]](#footnote-3). It is used for importing static networks from files in either CIM or the iTesla Internal Data Model (IIDM) formats, and also for the encapsulation of power flow engines.
* **iPSL**: iTesla Power Systems Library[[4]](#footnote-4), an open source Modelica library that provides dynamic models for a wide variety of power system elements.

The conversion and simulation process can be divided into the following main stages:

1. **Import CIM files** defining the static system model and case to the iPST internal format, IIDM (iTesla Internal Data Model).
2. **Run a power flow** on the IIDM data. This function provides values that are used as parameters of the components of the dynamic system model.
3. Perform a set of individual model simulations for **model initialization** and/or internal parameter derivation for the dynamic models of the elements that require it. This implies building a set of small dynamic models and executing time domain simulations on them. The results of these dynamic simulations provide inputs for the components of the complete dynamic system model.
4. **Build a dynamic system model** where every element in the static system model is replaced by a set of dynamic components defined in the iPSL. Values obtained in previous stages 2 and 3 are used to provide inputs to these components. The user must provide values for the rest of parameters required for the instantiation of the dynamic components.
5. **Apply user-defined events** to the resulting system model.
6. **Run time-domain simulations** using one of the Modelica dynamic simulation engines for which integration is provided (OpenModelica[[5]](#footnote-5) or Dymola[[6]](#footnote-6)).

# 3.- Getting started

This section explains how to install PSM tool and run it.

## 3.1.- Installation requirements

Before installing PSM, the following software should be installed:

* Java version 8[[7]](#footnote-7).
* OpenModelica recommended version v1.11.0,
* or Dymola recommended version 2016.

## 3.2.- How to install PSM

Two zip files are distributed  with the following names:

psm-v-YYYYMMDDhhmm.zip

psm-v-YYYYMMDDhhmm\_validation-data.zip

The second zip correspond to the validation data (see section Validation in User interface reference in this document) which contains large size files and thus is distributed separately.

The user should create a folder where the tool will be installed, for example psm/ (from now called PSM folder) and then proceed with the unzipping.

PSM tool is easily installed by unzipping the first \*.zip inside PSM folder. If the user is going to use the validation data, he/she should unzip the second zip file in the PSM folder. After unzipping, four new folder are created:

* data: containing all information related to cases, i.e. cases directories and Modelica libraries.
* helmflow: containing binaries for HELMTM-Flow loadflow engine.
* hades2LF: binaries for HADES loadflow engine.
* lib: required Java libraries.

The PSM tool is now installed.

## 3.3.- How to run PSM

Once the tool has been fully installed, the user can launch the GUI by using the following command psmgui (in PSM folder) on a terminal.

In Windows OS the tool can be launched by double clicking on psmgui.cmd.

This will start the user interface and the user can start playing with PSM.

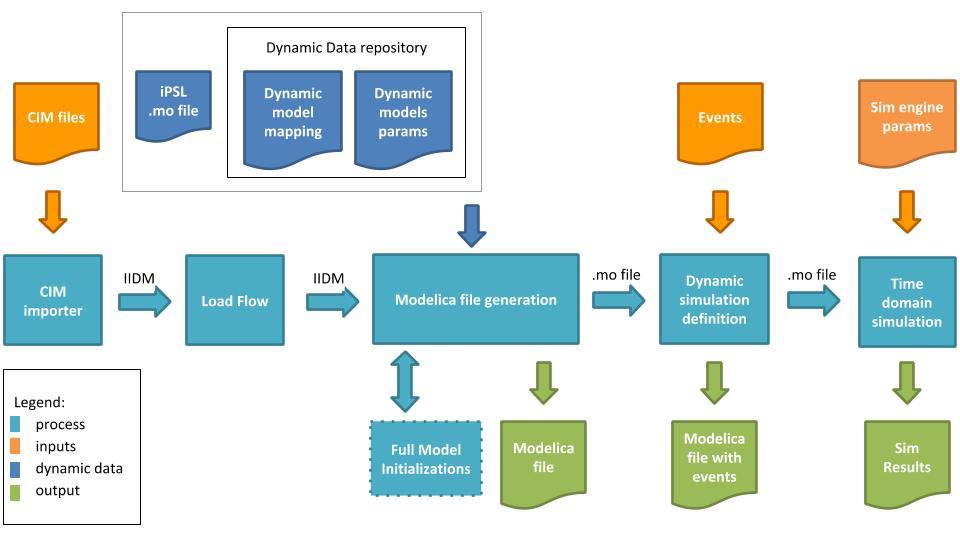
If Dymola will be used as Modelica engine by the PSM tool, the service should be started in the machine where it is installed. For this purpose the user must copy the folder lib and the script dymola\_integration\_service.cmd to the mentioned machine and launch this script.

# 4.- PSM Workflow

The PSM tool is intended to generate power system networks in Modelica (\*.mo files) from CIM files containing static system models and state variables for a given case plus dynamic data given in XML files. A loadflow computation is run over the static system to obtain the steady-state. A Dynamic Data Repository (DDR) is used for defining which dynamic model (from iPSL or user-defined) should be mapped to each element, and which parameters must be used to instantiate those dynamic models. The users also have the possibility to perform time-domain simulations using a Modelica solver engine (either OpenModelica or Dymola).

The tool has been designed to ensure modularity, allowing users to run processes either individually or as a full workflow. Various loadflow and Modelica simulation engines are provided.

The general architecture of the tool is shown in figure 1. The different modules involved in the PSM workflow are: i) CIM importer, ii) loadflow computation, iii) Modelica file generation, iv) Dynamic simulation definition and v) Time-domain simulation.



**Figure 1.**

The tool first converts CIM data files into the iTesla Internal Data Model (IIDM), on which a loadflow is computed. Then another module generates the Modelica file by connecting to the Dynamic Data repository for retrieving dynamic data. The user has the possibility to introduce events to be studied in the dynamic simulation and finally run time-domain simulations with a Modelica engine, selecting the desired simulation parameters. The individual modules are described below.

## 4.1.- CIM importer

The main goal of this module is to import the static system model from a file in CIM format, and convert it to IIDM. The module only supports CIM-compliant files (ENTSO-E CIM Profile 1 V14 in PSM version 1.0.0). The user is free to manually write/update CIM files, as long as the generated file is CIM compliant.

## 4.2.- Loadflow computation

This module is in charge of computing the loadflow on the IIDM network obtained in the previous step. The tool has been designed to work with two different alternative engines: the HELMTM-Flow loadflow engine[[8]](#footnote-8), and the HADES loadflow engine (RTE’s official loadflow engine[[9]](#footnote-9)).

The tool is intended to allow for easy switching between these two engines. The values obtained from the loadflow computation are re-injected into the IIDM network before moving on to the next module, i.e. the Modelica file generation. PSM generates a \*.xiidm file containing the IIDM network with the results from Load Flow (if it has been computed) in the case directory (the engine used is indicated in the file name).

The user also has the possibility to deactivate the loadflow computation and generate the Modelica file using just the existing input values from the CIM file.

## 4.3.- Modelica file generation

This module is responsible for the generation of the Modelica (.mo) file. The tool enables the user to specify parameters for each static element in the given system setup and provide dynamic models (relying on iPSL and/or user-defined models) with default parameters. As depicted in the general architecture of the tool (figure 1), this module connects to the DDR populated with the system dynamic data and all necessary models from the iPSL.

The Dynamic Data Repository is based on a set of XML files that store:

* Which mappings from static elements to dynamic models to be used in dynamic system model building (and its connections).
* The definitions required for building full model initialization simulations of complex dynamic models.
* The parameter sets used for dynamic model instantiations.
* Global declarations and system scope equations.

Full model initialization (in figure 1, the blue box below the Modelica file generation module) refers to the derivation of relevant and coherent initial values for all model variables based on loadflow outputs and external parameters. This is done by performing very short-time simulations on the specific component models using either OpenModelica or Dymola. This initialization is performed only for components models requiring explicit initialization, which is the case of some generator machines and maybe some specific regulation models.

## 4.4.- Dynamic simulation definition

This module is intended to define the simulation scenario. The user can define events to be triggered at given specific times during the simulation, as well as event parameters. This module also allows the user to define load variations, capacitor changes, etc. As a result, a new Modelica file with defined events is generated in addition to the Modelica base case generated in the previous step.

## 4.5.- Time-domain simulation

This module is in charge of running time-domain simulations on the Modelica files generated with the tool, allowing the user to choose between two Modelica simulation engines: OpenModelica or Dymola. PSM fully relies on these engines to run simulations, and thus they must be previously installed. The user also has the possibility of opening the generated Modelica file in any preferred Modelica environment for edition and simulation.

The time-domain simulation module generates output files in the standard Modelica output format (MATLAB binary format \*.mat), provided by both of the simulation solvers, as well as in \*.csv format. This will allow the user to import simulation results into the preferred Modelica environment or other software for plotting and analysis of results.

## 4.6.- User interface

The tool also includes a basic graphical user interface (GUI) with functionalities allowing the user to select which processes to run, on which data sets, displaying progress and logs, without having to manually run command line actions.

# 5.- Tutorial

This section shows how to use the tool with two basic examples that cover most of the features available in the PSM tool.

PSM tool provides a total of nine (9) sample cases referred to as *Reference Cases* in the PSM GUI. The Reference Cases are public cases from IEEE archives, small parts of the European network, and tutorial examples.

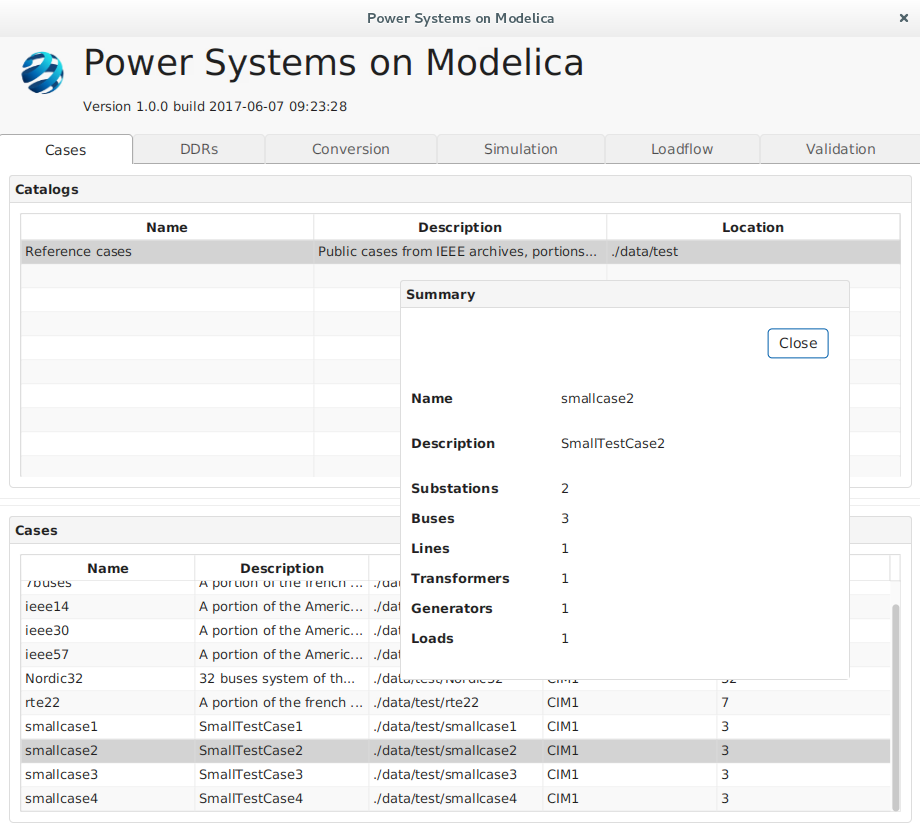
* smallcase1 to smallcase4, which are variants of a small 3-bus system,
* IEEE14, IEEE30 and IEEE57, from the power systems test case archive[[10]](#footnote-10),
* 7buses case, built from a small portion of the French electric power system,
* and the Nordic32 case, from the Swedish transmission network.

For this Tutorial we will use the smallcase2 for Example 1, and IEEE14 for Example 2 below.

## 5.1.- Example 1

In this first example the smallcase2 will be used. The user will first be guided through the PSM tool using this reference case in order to see the tool workflow. Then some modifications will be performed on the case to show the tool’s capabilities.

To start using the smallcase2, the user should run the tool and go to the section **Cases** of the GUI. Once there, go to the Catalogs window and select Reference cases. This will show a list of all Reference Cases available in PSM in the Cases window. By right-clicking on any particular case the user can see a summary with a short description. The following image shows the summary for smallcase2.



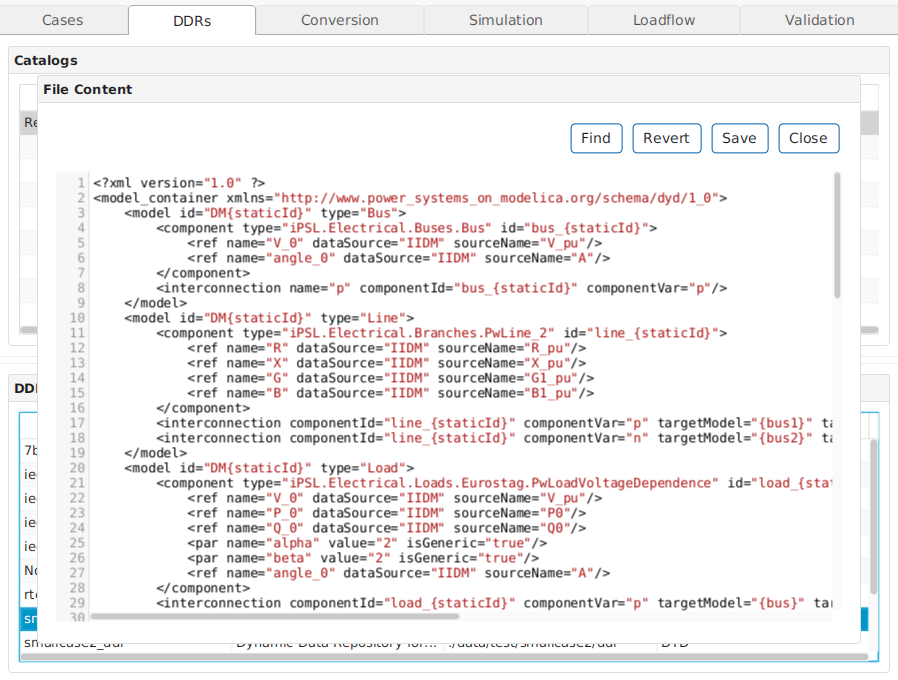
All smallcases have a similar structure: 3 buses, two of them (GEN and GRID) connected by a transformer, and the other two (GRID and INF) connected by a transmission line. In smallcase2, a generator and a load are connected to bus GEN and an infinite bus is connected to bus INF.

The equipment and topology information for each reference case can be seen in the CIM files (\*.xml) located inside the PSM installation folder ./data/test/. In this case, ./data/test/smallcase2/.

The **DDRs** section of the GUI allows the user to inspect DDR files corresponding to each case. By right-clicking on the specific DDR, all \*.dyd and \*.par files found in the case folder are shown. For the Reference Cases these files are model.dyd, params.par, events.dyd, and system.dyd, but the user can modify them and organize the contained information, as long as they are located in the case folder.

The DDR files are located in each Reference Case folder. For the smallcase2, this location is ./data/test/smallcase2/ddr/.

The following image shows the model.dyd file for smallcase2:



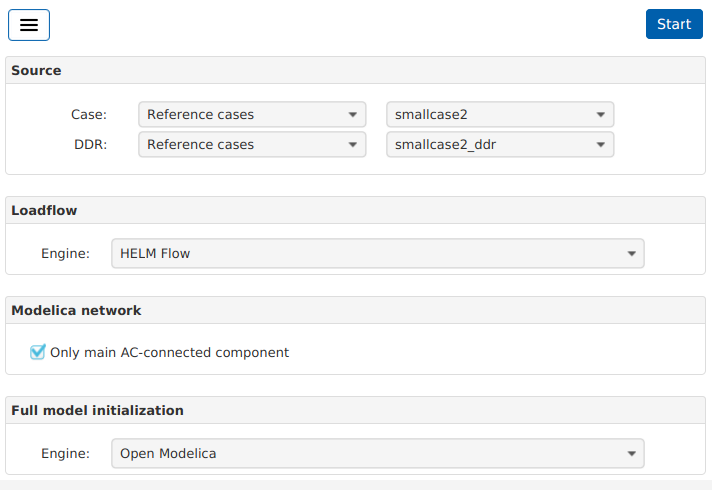
The GUI allows searching any specific element inside a \*.dyd file by using the Findoption when the \*.dyd file is open (see the image above). The user can also modify the \*.dyd file and save it with a new name, or revert all changes to the original form by using Revert option.

In the model.dyd shown above for smallcase2**,** it can be seen how static elements from CIM files identified with staticId are mapped to iPSL dynamic models. For example, elements with staticIds beginning with bus (“bus\_\_{staticId}”) will be mapped to the dynamic model “iPSL.Electrical.Buses.Bus” from the Modelica iPSL (line 4 in the above image). Elements with staticIds beginning with line (“line\_\_{staticId})” will be mapped to the iPSL dynamic model for transmission lines “iPSL.Electrical.Branches.PwLine\_2” (line 11 in the above image). The elements for loads identified as “load\_{statisId}” are mapped to the dynamic model “iPSL.Electrical.Loads.Eurostag.PwLoadVoltageDependence” (see line 21 in the above image).

After these mappings, the parameters needed for each dynamic model in iPSL must be referenced as well as the connection that must be established for each model. The user can review all models available (parameters needed and connections) in the iPSL by inspecting the the iPSL/ directory file located at ./data/library/.

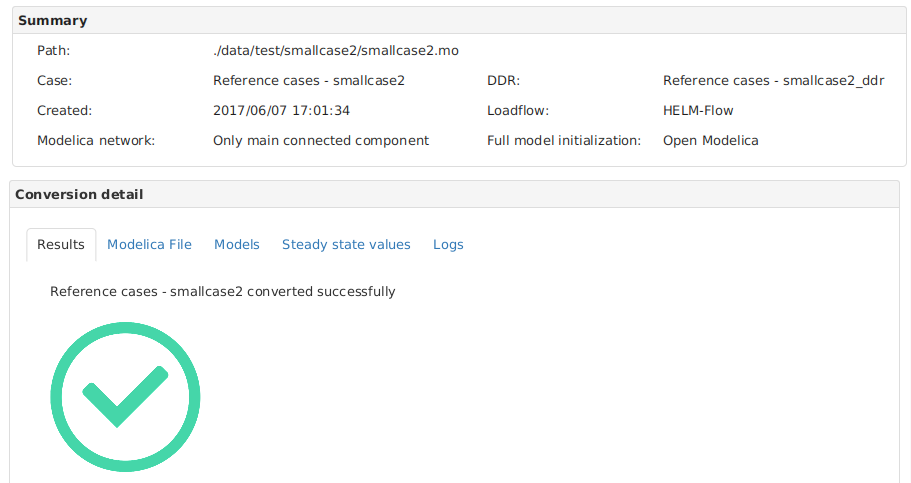
For converting the case to Modelica, the user can go directly to the **Conversion** section; or alternatively, from the **Cases** section, select the Conversion option from the menu that opens up by right-clicking on a specific case.

In the **Conversion** section the user must select the case to be converted, the DDR, the Loadflow engine to be used for loadflow computation, the Modelica engine for full model initialization, and whether or not only the main AC-connected component should be converted.

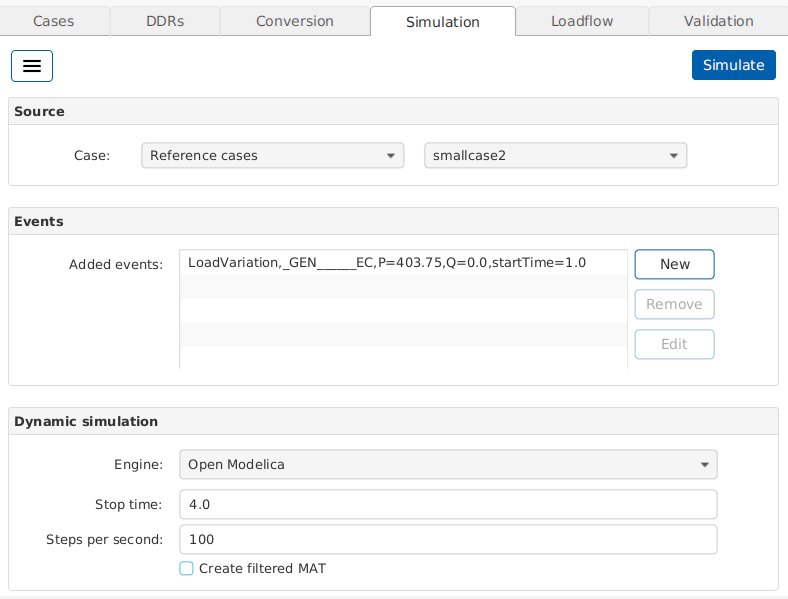


After completing the conversion setup, it can be launched by clicking on the Start button. A new window appears, showing the progress in every step of the conversion.

And once the conversion to Modelica has been correctly completed, the GUI shows a Summary of the case (with the path to the Modelica file and engines used for the conversion). The Conversion detailwindow shows a message indicating if the conversion was correctly completed (see image below) and allows the user to inspect the generated Modelica file, to see the mapping performed between Static Ids and Dynamic Type in Models tab, check the steady state values obtained in the loadflow computation or coming in the CIM file (if no loadflow is computed), and review the logs from conversion.



The next step consists in adding events at specific times and simulate the generated Modelica file to see the results of time-domain simulation. For this, the user has to go to the **Simulation** section, select the case for simulation, add events, select the Modelica engine to be used, the preferred Stop time and Steps per second. For this example, a Load variation event (with P = 403.75 MW and Q = 0.0 MVA) at load GEN\_\_\_\_\_EC is activated at time t = 1.0 s. The Stop time and Steps per second are set to t = 4.0 s. and 100 respectively as shown in the image below.



The simulation generates a \*.mat file (standard format used by Modelica environment tools) with all variable values available, but the user has the option to create a filtered \*.mat file by selecting the variables to print in the file modelicaengine.properties located at ./data/cfg/[[11]](#footnote-11). As can be seen in this file, the default variables to print in this filtered \*.mat are voltages on buses (magnitude and angle), active and reactive power in loads and some generator variables[[12]](#footnote-12).

When the simulation is completed, the GUI shows the results in the Simulation detail window:

* voltage curve(s) at the bus(es) near to the event(s) (in this case bus\_\_GEN\_\_\_\_\_TN),
* the base case Modelica file, the Modelica file with events included,
* as well as the logs generated during the simulation.

In the plot shown in Results tab the user can add curves corresponding to the voltage at any bus in the system.

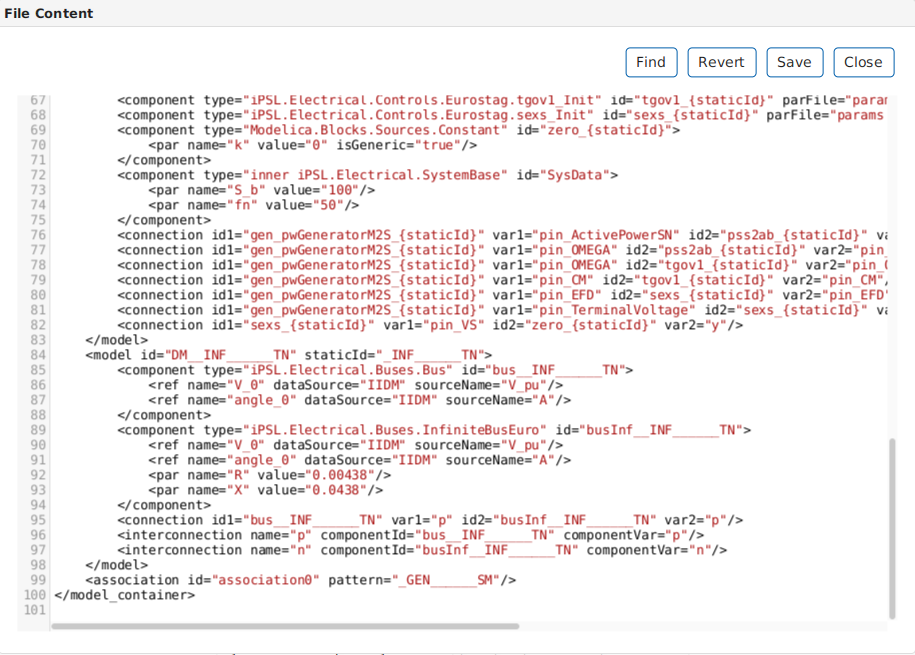
The user can review each one of these results by navigating the tabs in the Simulation detail window.



Now that smallcase2 has been correctly converted and simulated adding a load variation event, the infinite bus will be removed from bus “\_INF\_\_\_\_\_TN”, and all previous steps including conversion and simulation steps will be repeated in order to see the difference in the results. The user may save the generated modelica file smallcase2.mo located at ./data/test/smallcase2/ as smallcase2\_original.mo because this file is rewritten after each conversion and in this example it is interesting to have a copy of the original.

Before doing any modification to the DDR files, it is highly recommended to go to **DDR** section and use the option Duplicate DDR that appears when right-clicking on a specific case. A new ddr folder named ./ddr\_example1/ should be created at the same level of the original ./ddr/ (the GUI will automatically identify this folder in the DDR section once the user refreshes the window by clicking on any tab of the main window and comes back to the DDR tab).

To remove the infinite Bus model, the user should open the models.dyd file in ./ddr\_example1/ and remove all reference to the infinite bus shown in the image below from line 84 to line 98.



Once removed, the user should select the Check DDR option in the menu that appears by right-clicking on the case DDR to ensure that the information in the DDR is coherent and that there are no syntax errors. The smallcase2 must be converted again using the ./ddr\_example1/ as DDR.

Once the conversion is completed, the user can open the Modelica file and search for “InfiniteBusEuro” and verify that there is no reference to this model. This file can also be compared to the original Modelica file (previously saved) in order see the differences (model, connection and annotation from the infinite bus are no longer in the system).

Now the simulation of the recently converted smallcase2 must be repeated by using the same configuration and event (load variation) previously used.

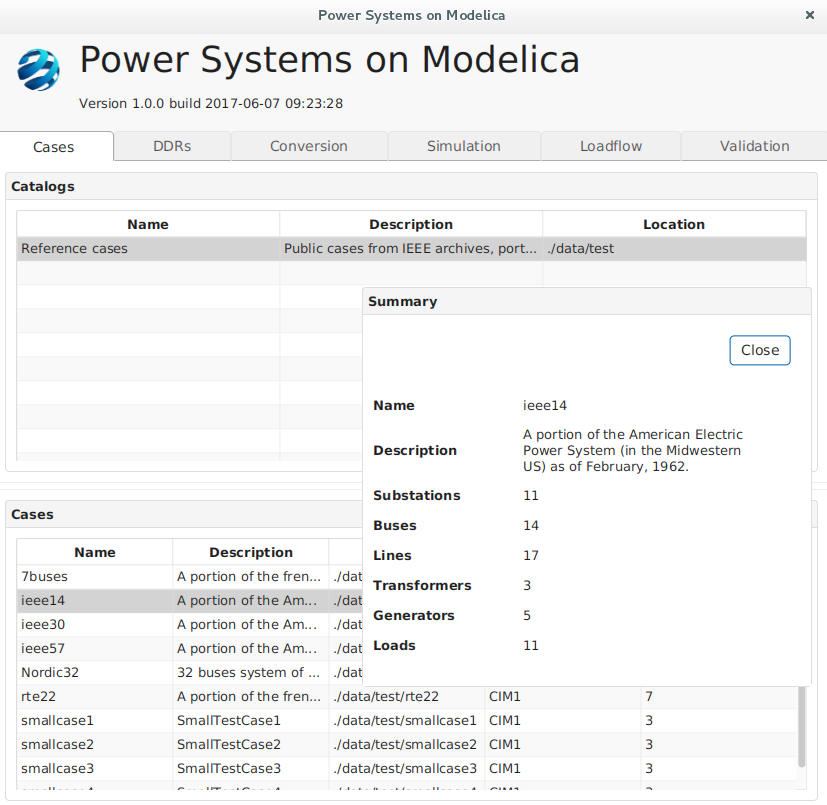
The following image shows the results obtained simulating the smallcase2 with a load variation at t=1.0 second and no infinite bus in the system.



## 5.2.- Example 2

In this second example the IEEE14 case will be used with the PSM tool. As in example 1, the user will be guided through the tool’s workflow using IEEE14 and then some modifications will be performed on the case in order to show the tool’s capabilities.

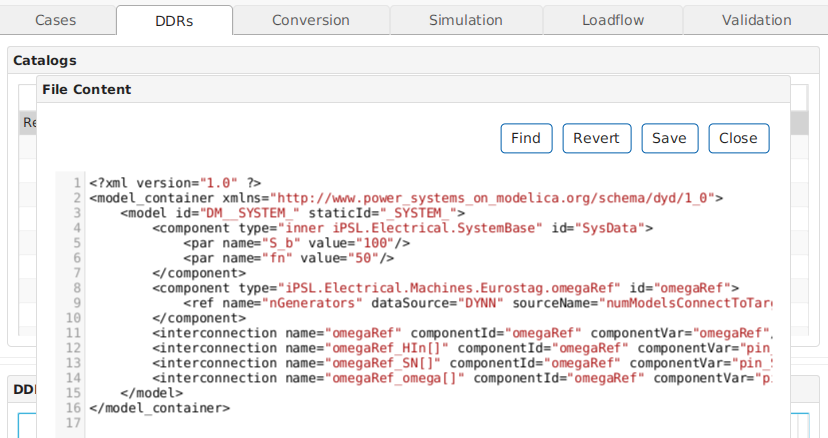
In the **Cases** section from the GUI, select ieee14 from Reference cases, and right-click on the case to open the menu. Select **Summary** option to obtain a description of the case and the elements composing it.



From the **DDR** section, all \*.dyd files and \*.par files associated to the ieee14 DDR can be inspected. On the same menu that opens up by right-clicking on any specific case, the user also has the possibility to check whether the syntax is correct in all files and whether there is any repeated orincoherent information, by using the Check option. This prevents, for example, having two files containing a mapping for the same static Id.

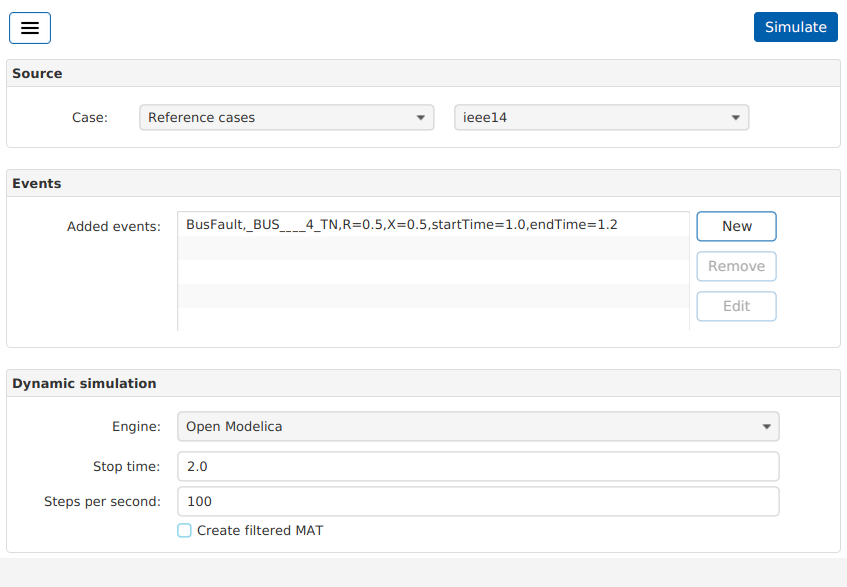
As previously commented, in this same menu, there is also a Duplicate DDR option which allows making an exact copy of the content of the DDR folder, where the user can start modifying and renaming files at will, without losing the originals.

The following image shows the system.dyd file for the ieee14 case, where the user can see, for example, the value used as system base (line 4, SNREF = 100).



For the conversion of the ieee14 case, the same setup used in Example 1 for smallcase2 will be used, except for the case itself and the DDR files, of course. The user should launch the conversion and inspect the generated Modelica file in order to get familiar with it (more details are given in the Modelica files subsection of the **Dynamic Data reference** section of this document).

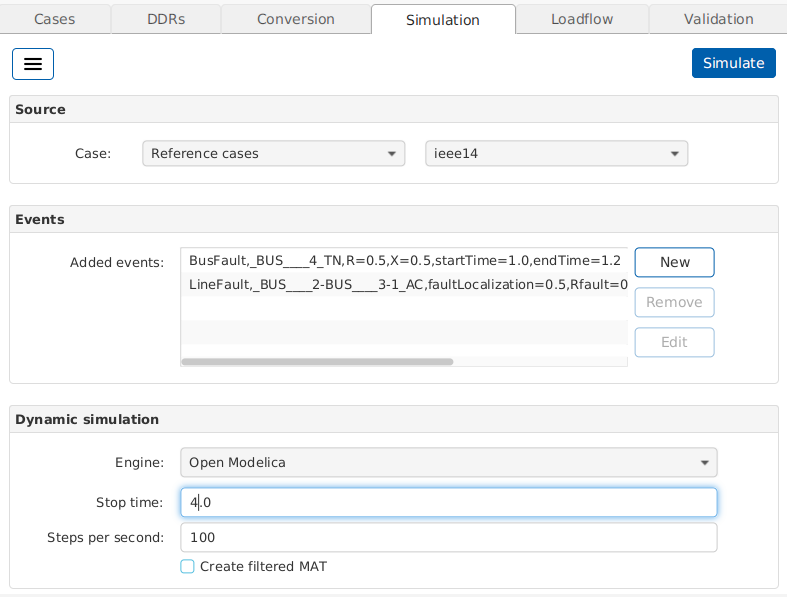
For the simulation of this ieee14 example, a Bus Fault event is first introduced at Bus\_\_\_\_4\_TN (R=0.5 p.u. and X = 0.5 p.u.) between time t = 1.0 s and t = 1.2 s. The simulation Stop time is set to 2 seconds and the Steps per seconds to 100.



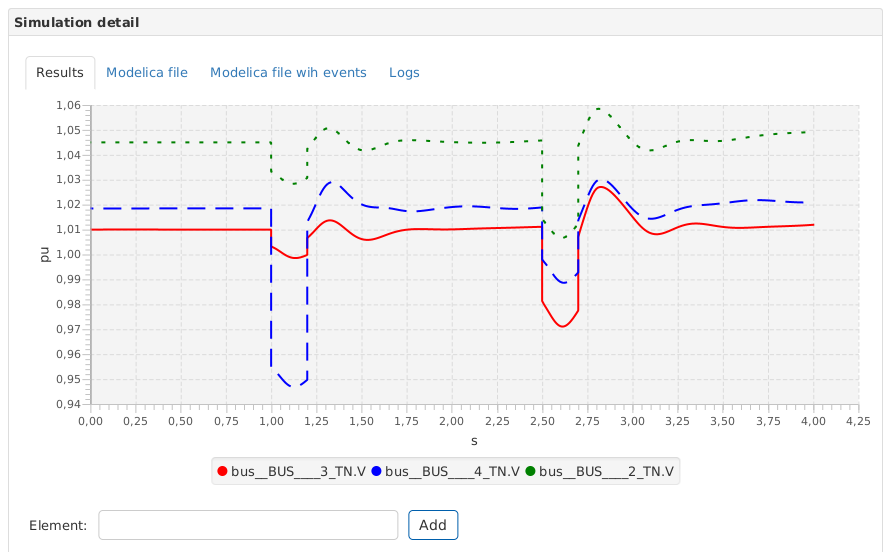
The results obtained with this scenario, once the simulation is completed, are shown in the following figure.



Now a second event is introduced in the simulation, by clicking on New Simulation. The event is a Line Fault in line BUS\_\_\_\_2-BUS\_\_\_\_3-1\_AC (faultLocalization=0.5, R = 0.2 p.u. and X = 0.2 p.u.) between time t = 2.50 seconds and t = 2.70 seconds. The simulation Stop time is set to 4 seconds and the Steps per seconds to 100.



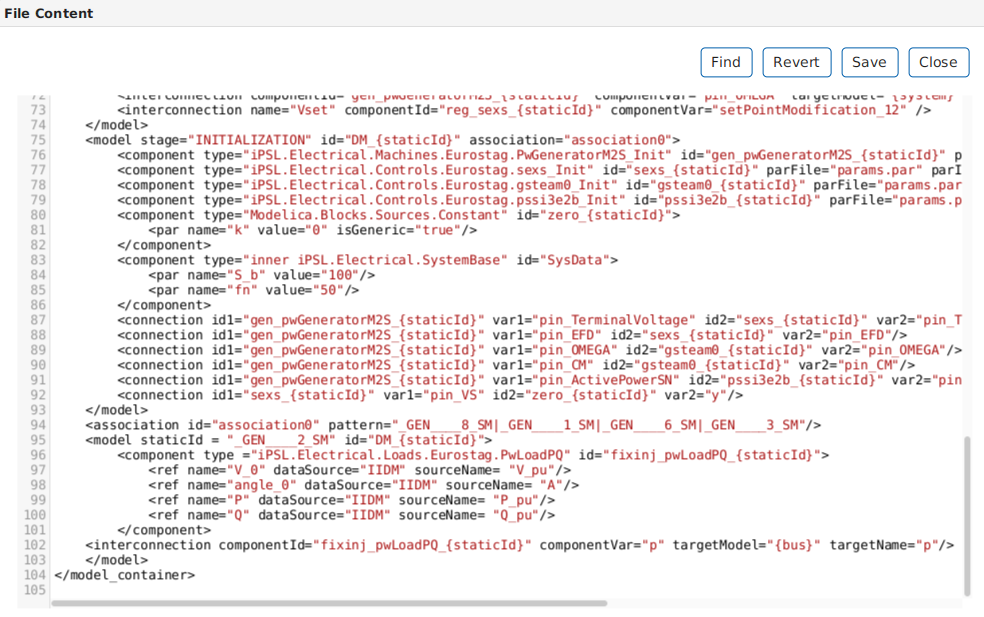
The following figure shows the curve obtained for the ieee14 case with a bus fault event and a line fault introduced successively. The curves shown correspond to the voltage at buses BUS\_\_\_\_4\_TN, BUS\_\_\_\_2\_TN and BUS\_\_\_\_3\_TN which are the buses where the bus fault occurs and the buses at the two ends of the line where the line fault occurs.



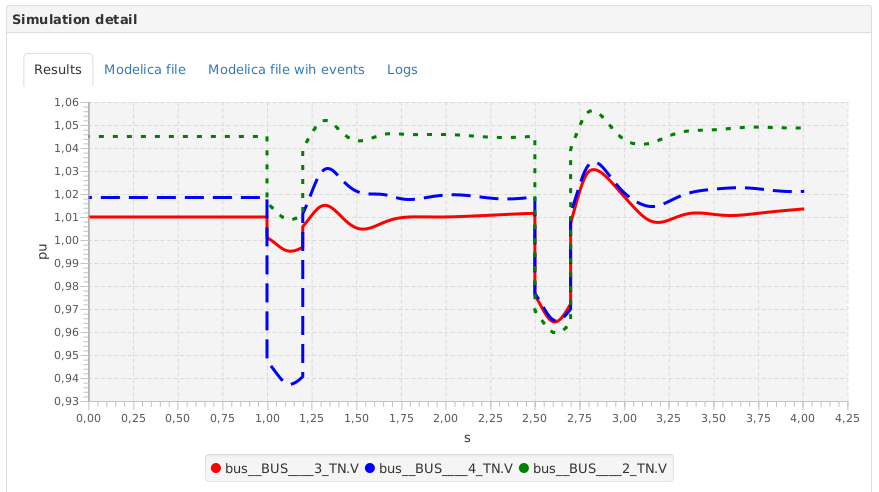
In this second example of the tutorial, the ieee14 case will be modified by removing dynamic data associated to a given generator in order to see how PSM deals with this scenario. As commented before it is highly recommended to use the Duplicate DDR option that appears by clicking on the case in the DDR section. The new folder can be named ./ddr\_example2/. Now, the user can open the model.dyd from the new ieee14 DDR and perform all the desired modifications.

The idea is to remove the generator that will be modelled as a fixed injection from the list of “association0”, and include a new mapping for this generator to the dynamic model “iPSL.Electrical.Loads.Eurostag.PwLoadPQ”[[13]](#footnote-13). In the file param.par the user can remove all parameters associated to the generator that will be modelled as a fixed injection, but since it will not be referenced in the mapping this step is not strictly mandatory; the Modelica file generated without removing parameters will still be correct. As commented at the beginning of this example, the user should check if the syntax of all files in DDR folder is correct and that there is no duplicate information, by using the Check DDR option.

The next screenshot shows the model.dyd for ./ddr\_example\_2/ folder, where the Generator identified with staticId = “\_GEN\_\_\_\_2\_SM” has been removed from association0 in line 94, and it has been modelled as a fixed injection in lines 95 to 103.



Now the user must convert the ieee14 case again using the ./ddr\_example2/ folder as DDR, in the **Conversion** section. Then repeat the simulation introducing the same two events (Bus fault and Line fault) with the same parameters. The time-domain simulation for this modified ieee14 is shown in the curves below, where it can be seen how the response to the line fault event changes, with respect to the previous one, because it occurs near the generator that has been replaced by a fixed injection.



# 6.- User interface reference

The Graphical User Interface (GUI) of the PSM tool is divided in six main sections[[14]](#footnote-14): **Cases, DDRs, Conversion, Simulation, Loadflow,** and **Validation**; that can be accessed by navigating the tabs of the main window. In the following, each of these sections is described in detail.

## 6.1.- Cases section

The **Cases** section gives access to all cases available in the tool and new cases created by the user. These files are located below the PSM installation folder ./data/test/.

The Reference Cases are public cases from IEEE archives, portions of the European network, and small tutorial examples. There are 9 Reference Cases in total:

* smallcase1 to smallcase4, which are variants of a small 3-bus system,
* IEEE14, IEEE30 and IEEE57, from the power systems test case archive[[15]](#footnote-15),
* Nordic32 case, from the Swedish transmission network,
* 7buses case, built from a small portion of the French electric power system,
* and the RTE-22 case also from the French electric power system.

By right-clicking on a specific case, the user has access to a menu with several options: Summary, Conversion, Modelica file (only active when a conversion has been completed), Simulation, and Loadflow.

The Summary option opens a popup window with a description of the case and the number of substations, buses, lines, transformers, generators and loads in the system.

The Conversion option redirects the user directly to the **Conversion** section.

The Modelica file option is activated once a conversion has been successfully performed in the specific case being consulted. It redirects the user to the **Conversion** section showing the last Modelica file generated.

The Simulation option redirects the user directly to the **Simulation** section. This option is also activated once a conversion has been performed in the specific case being consulted, since the simulation is performed in the Modelica file generated after conversion.

The Loadflow[[16]](#footnote-16) option redirects the user directly to the **Loadflow** section in case the user is interested in running a loadflow computation with one the two engines available (HELMTM-Flow and Hades), without performing the full conversion.

If the user wants to convert new cases it is important to save all necessary files in a folder located at the same level as all other cases directories. This way the PSM tool will recognize the new case and list it with all other available cases.

## 6.2.- DDRs section

This section gives access to all the files that make up the Dynamic Data Repository (DDR) for each case available in PSM, either created by the user or provided as reference cases with PSM. The files making up the DDR are located in a folder called ./ddr/ inside each case folder.

Within the DDR section, the user must select a case in the DDRs window and by right-clicking on a particular case he/she can explore each of the files making up the **DDR** (i.e., system.dyd, events.dyd, params.dyd, and models.dyd for Reference cases) by selecting them from the menu that opens up automatically. The specific information given in each of these files is explained in the Dynamic Data Repository section of this document.

There are also two other options in this menu that can be applied to DDR files: Check DDRand Duplicate DDR.

The Check DDR option allows to check the syntax in all \*.dyd files corresponding to the selected DDR, and verify whether there is any duplicate and/or incoherent information in the DDR (for example, mapping information for a given static element being repeated in more than one file).

The Duplicate DDR optionallows the user to copy the content of the ./ddr/ (\*.dyd and \*.par files) in order to give the user the possibility of modifying these files at will while keeping a copy of the originals.

**Note**: the IEE14 case contains an additional DDR called ddr\_fixed\_injections/ that corresponds to the IEEE14 case where all generators have been replaced by a fixed injection. This DDR is provided as an example for the user to see how this replacement can be implemented.

## 6.3.- Conversion section

This section allows the user to select the case to be converted, the loadflow engine to be used for loadflow computation, and the preferred Modelica engine (OpenModelica or Dymola) for performing full model initialization. Once all the necessary items are selected, the conversion can be launched by clicking on Start button.

The menu located at the top left corner of the GUI contains the following options:

* Load: allows loading a previously saved configuration (performed with the Save option) for the conversion of a case.
* Save: allows saving a desired configuration to be used in the future.
* Clean: removes all selections done for the Case and DDR to be used.
* Check: verifies whether all static models have a dynamic model associated, i.e., checks the mapping. In case the mapping is not complete or incorrect, the Check option will throw a message and the user should review the Models and Logs tabs of the Conversion detail window.

When the conversion is running, the progress status of each step involved in the conversion is shown. These steps are:

* Static network importer (import from CIM)
* Loadflow computation (once computed an \*.xiidm[[17]](#footnote-17) file is generated in the case directory).
* Modelica network builder, which includes the check of DDR and the full model initialization. The Check function is equivalent to the one available in the menu located at the top left corner of the Conversion section.
* Modelica exporter (generation of the \*.mo file)

Once the conversion is completed, the GUI shows the following information:

* **Summary** window:
* information on the path location where the Modelica file has been created,
* the static system and the DDR used,
* the date and time of conversion,
* the loadflow engine used,
* the type of Modelica network ,
* and the Modelica engine used for full model initialization.
* **Conversion** detail: five (5) different tabs are available displaying the following information:
* Results tab: showing if the conversion has been completed successfully.
* Modelica file tab: showing the Modelica file generated in the conversion.
* Models tab: the mapping between static IDs and Dynamic IDs for each element.
* Steady state values: four (4) plots can be reviewed, corresponding to Voltage, Phase, Active Power, and Reactive Power on each bus of the system.
* Logs: logs from the conversion process. If an error occurs during conversion, the user should check this tab.

Once a conversion has been completed, the menu located at the top left corner of the GUI shows the option New conversion.

## 6.4.- Simulation section

In this section the user selects the Modelica base case to be simulated with the preferred engine (OpenModelica or Dymola) and the events that will be included in the time-domain simulation.

As in the **Conversion** section, the **Simulation** section also includes a menu located at the top left corner of the GUI, containing the following options:

* **Load**
* **Save**
* **Clean**
* **Check**
* **Verify**

The Load option allows loading a previously saved configuration (performed with the Save option) for the simulation of a case. The Save option allows saving a desired configuration to be used in the future. The Clean option removes all selections done for the Case, and the selected events.

The Check option in this **Simulation** section performs the standard Check Model option from all Modelica environments used (in this case OpenModelica or Dymola), which checks a model and returns a number of variables and equations.

The Verify option performs a very short simulation (stop time = 0.001 s) that allows one to verify if the system from the Modelica file can be correctly translated, initialized, and simulated with the selected Modelica engine.

To introduce events the user must select the option New and a new window will be opened in order to define the events that will be included. Then a Type of event must be selected and a list appears indicating the Elements in the network where the event can be applied, as well as the necessary parameters to activate the event. As of PSM V1.0.0, these are the events that are available for simulation:

* Bank Modification
* Bus Fault
* Generator Vsetpoint modification
* Line Fault
* Line opening on both sides
* Line opening on receiving side
* Line opening on sending side
* Load variation

Beside the events and the Modelica engine, the simulation setup requires that the user selects the desired stop time for the simulation, and the steps per seconds. When the configuration is completed the simulation is launched by clicking on Simulate button on top right corner of the GUI. The simulation can be stopped at any time by clicking on Stop button.

When the simulation is running, the progress status of each step involved in it is shown. These steps are:

* Static network importer (import from CIM)
* Modelica parser (generates the Modelica file with events)
* Modelica Dynamic Simulation (run simulation on the selected Modelica engine simulator)
* Modelica Dynamic Simulation Results (get simulation results to be shown in GUI)

Once the conversion is completed, the GUI will show the following information:

* **Summary** window:
* the static system and the DDR used,
* the date and time of simulation,
* and the Dynamic simulator used.
* Conversion detail: four (4) different tabs are available displaying the following information:
* Results tab: showing a plot with the voltage at the buses nearest to the event. In case an error occurs during simulation, a message will appear in this tab.
* Modelica file tab: showing the Modelica file generated in the conversion.
* Modelica file with events tab: showing the Modelica file with the events included.
* Logs: logs from the simulation process. If an error occurs during simulation, the user should check this tab.

## 6.5.- Loadflow section

This section enables the user to perform a loadflow computation on the static network without performing any conversion. The user must select the case to be used from the cases available in the Catalog, and run the loadflow computation with the preferred engine: HELMTM-Flow or Hades. Since Hades engine is only available under Linux, the user can only use this engine on Linux version of PSM.

This section appears inactive to users when PSM is installed. In order to activate it, the user must go to the GUI configuration file located in the folder .../data/cfg/, open the file gui.properties and change the parameter menu.disableLoadFlowview from true to false.

The results of the loadflow computation are shown in the Loadflow results window. The values obtained for voltages, phases, active power and reactive power are shown in plots and in tables. Logs are also available if an error occurs during the loadflow computation.

This section also allow to perform a comparison between loadflow computation performed with the two available engines, by selecting Comparison between HELM-Flow and Hades in the Engine parameter.

The results of the comparison are shown in different curves and tables, separated by type of variable: Voltages, Angles, Active Power, and Reactive Power. These can be accessed by navigating the different tabs available when the loadflow comparison is completed.

In the Loadflows comparison window, for each variable type, the PSM GUI displays the results in three different manners:

* A plot showing the differences in results obtained with the two engines, with the values for the average and maximum differences found.
* A plot showing the actual values obtained with each loadflow engine.
* A table with the list of static elements in the network, the loadflow values obtained with each engine, and the absolute difference.

## 6.6.- Validation section

This section allows the user to perform the validation of cases converted to Modelica using PSM, against other (proprietary) software for dynamic power system simulations, such as Eurostag[[18]](#footnote-18), PSS/E[[19]](#footnote-19) or others. The external software used to compare results in the validation is referred as the Reference Software.

For the validation of cases three \*.csv files are needed:

* Mapping: mapping between variables to be compared in the validation.
* Expected: the results from the simulation performed in the Reference software with a fixed time step.
* Case: the results from the case simulated with PSM with the same time step used in the reference software.

The Step size used in the simulation performed with both software must be also indicated in the GUI (it is important to use the step size in both software). The user must verify that this parameter is correctly set in order to ensure a correct validation. In case the user selects an step size smaller than the one used in the Reference software, PSM tool will take the latter for computing the validation and a message will be given in the GUI.

All Reference Cases provided with PSM has been built in Eurostag and simulated with a fixed time step, in order to perform the validation against this proprietary software. Inside each case folder (located at ./data/test/) the user will find a ./validation/ folder where all necessary \*.csv files for validation have been saved: Mapping is given in NamesMapping.csv, Expected results are saved in ReferenceData.csv and Case results in [Case\_Name]\_res.csv.

**Important Note**: Most of the simulation files provided with References Cases have been run with a step size = 0.0001 seconds, except for the smallcase3 that has been run with a step size of 0.00001, and the IEEE30 and IEEE57 that have been run with 0.005 and 0.001 respectively[[20]](#footnote-20). The Nordic32 is the only Reference case that does not include validation files because running this case with a small step takes very large times to simulate in Modelica due to its size and complexity and the generated files are extremely large in size (the Nordic32 case has been validated outside PSM). For the RTE-22 case, small differences (<1e-3) still remain between Eurostag and PSM simulation results, and thus a full validation has not been completed yet.

If the user wants to change the step size used for the validation of a given case, he/she must modify the parameter numOfIntervalsPerSecond in modelicaengine.properties file located at ./data/cfg/.

If the user wants to validate new cases, the first steps to follow would be to:

* build the equivalent system in the reference software,
* run the system with a fixed time step,
* and save the simulation results in \*.csv format.

Then, the new case converted to Modelica with PSM must be simulated using PSM and indicating the values to be printed in the \*.csv file generated after each simulation (the \*.csv file is located on the case folder).

By default, PSM prints voltages on buses (magnitude and angle), active and reactive power in loads and some generator variables (efd, cm, lambdad, lambdaf, lambdaq1 and lambdaq2). The values to be printed can be modified in the file modelicaengine.properties located at ./data/cfg/, by changing the following expression:

resultVariables=[a-zA-Z0-9\_]\*((TN.(V|angle))|(EC.(P|Q))|(SM.(efd|cm|lambdad|lambdaf|lambdaq1|lambdaq2)))

It is very important to make sure that both software are printing all variables to be compared in the \*.csv files and that they are included in the mapping with the corresponding IDs. The \*.csv files should be reviewed before launching the validation in order to confirm that all compared variables are correctly printed.

To validate a new system, it is highly recommendable to use the option Clean in the menu located at the up left corner of the GUI before introducing the new files.

Once the validation is completed, the GUI shows a window called Validation where a table is displayed with four (4) columns:

* A first column showing the variables compared.
* RMSE: the Root Mean Square Error obtained comparing the software results.
* Absolute Difference: the Absolute Difference between the two software results.
* Relative Difference: the Relative Difference between the two software results.

The first row show the Tolerance in percentage (%) for the RMSE, Absolute difference and Relative Difference used for accepting the system as validated. These Tolerance values can be changed by the user and they are used to check if the Total percentage (calculated summing the % of each variables) is below tolerance (the cell is coloured in green) or above tolerance (the cell coloured in red).

The percentage for each variable is calculated by seeing how many points of the RMSE, Relative Difference and Absolute Difference are below a given threshold with respect to the total number of points compared (including all variables). The user can check the result for each variable by double clicking on a variable and reviewing the table that opens-up.

Fox example, by clicking on voltages V, the pop-up table will show:

* all elements considered,
* the RMSE obtained for each element in the time series considered (the cell is coloured in green if the RMSE > threshold 1.0E-03),
* the % of point with relative difference RD above the threshold 1.1E-02,
* and the % of point with relative difference AD above the threshold 1.1E-03,

The threshold used in the validation are set in the file case-validation.properties located in ./data/cfg/ folder. The user can also see the thresholds as well as the RMSE used by placing the mouse on the headers RMSE, Absolute differences > threshold and Relative Differences > threshold.

Inside the table for a given variable (V, angle, P, Q, etc.) the user can select one specific element (i.e. bus\_[ID].V) and see the curves corresponding to the Absolute and Relative differences, and the time series for the variable in the element selected.

# 

# 7.- Configuration files

This section describes the different configuration files included in PSM V1.0.0. and located at the directory ./data/cfg/. The files are listed below:

* cases.properties
* ddrs.properties
* conversion.properties
* iidmNames.properties
* hades2.properties
* modelicaengine.properties
* case-validation.properties
* simulation.properties
* gui.properties
* import.properties
* javaScriptPostProcessor.properties

In the following each one of this files is briefly described indicating the configuration parameters that the user can modify for an advanced use of PSM.

1. **cases.properties**

Contains a set of configuration parameters describing the available catalogs and cases. In order to add catalogs to PSM the user should change the number of catalogs and add the corresponding description in the file.

* *catalogs.num*: indicates the number of cases catalogs.
* *catalogs.X.name*: name of the cases catalog number X.
* *catalogs.X.description*: description of the cases catalog number X.
* *catalogs.X.location*: path to the location of the cases catalog number X.

1. **ddrs.properties**

This file contains a set of configuration parameters describing the DDRs available with the tool. These parameters are listed below:

* *catalogs.num*: indicates the number of DDR catalogs.
* *catalogs.X.name*: name of the DDR catalog number X.
* *catalogs.X.description*: description of the DDR catalog number X.
* *catalogs.X.location*: path to the location of the DDR catalog number X.

1. **hades2.properties**

Written automatically by the application every time it is started. Contains the path to Hades binaries.

1. **modelicaengine.properties**

The set of parameters listed in this configuration file describe the needed configuration for dynamic simulation engines. Parameters are listed below:

* *modelicaEngineWorkingDir*: path indicating the working directory for the dynamic simulation data. If this parameter is not set, a temporary folder in ./data/ will be used.
* *libraryDir*: this parameter indicates the path(s) to the folder(s) containing the library or models used for the dynamic simulation. If user should add many paths they should be separated by “,”.
* *resultVariables*: regular expression indicating the variables that the user wants to filter. If this parameter is empty, all variables will be give from the dynamic simulation results.
* *startTime*: indicates the start time for the dynamic simulation. By default is 0.
* *stopTime*: indicates the stop time for the dynamic simulation. By default is 1, but user should change it if needed.
* *tolerance:* this parameter refers to the integration method tolerance. By default is 0.0001.
* *numOfIntervalsPerSecond*: number of intervals per second for the simulation.
* *createFilteredMat*: this is a boolean parameter (true or false) that indicates if the tool will create a .MAT file with a list of filtered variable. The list of variables should be defined in the parameter *resultVariables.*
* *depth*: indicates the task that will be run by the dynamic simulator engines. Three options are available: 0 - Simulate, 1 - Check and 2 - Verify. Option 0 performs the dynamic simulation, option 1 only checks the case and option 2 checks the case and runs a short simulation in order to quickly test if the case simulates or has compilation/translation errors.
* *solver*: this parameter is a string that indicates the solver that PSM will use to simulate.
* *simFlags*: string defining a list of extra parameters that OpenModelica[[21]](#footnote-21) engine will use for the simulation. This parameter is not used by Dymola.
* *webService*: indicates the Dymola server url that should be set if Dymola is used for dynamic simulations. If the Dymola server is running locally this parameter should http://localhost:8888/dymservice?wsdl.

1. **case-validation.properties**

The set of parameters listed in this configuration file describe the needed configuration for case validation. Parameters are listed below:

* *stepSize*: indicates the step size used in seconds.
* *toleranceTh*: indicates the tolerance value that is applied to determine if the input data is too small. If the input value is < *absThreshold\*toleranceTh,* then the relative differences are replaced by absolute differences.
* *absThreshold*: indicates the threshold to determine absolute differences that are classified as erroneous.
* *relThreshold*: indicates the threshold to determine relative differences that are classified as erroneous.
* *variables*: indicates the number of variables to validate.
* *variables.variableX*: indicates the variable to validate. The index X ranges between 1 and the total number of variable to validate (parameter *variables*).
* *writeFile*: this is a boolean parameter (true or false) that indicates if the case validation engine should write two files with the relative differences and absolute differences.
* *pathOutput*: indicates the path where the case validation engine write the files whenever *writeFile* parameter is enabled. This parameter should be included by the user when needed.

1. **gui.properties**

The set of parameters listed in this configuration file describe the default configuration for GUI options. Parameters are listed below:

* *menu.disableLoadflowView*: disable the loadflow tab in the main window.
* *menu.disableSwtoswValidationView*: disable the case validation tab in the main window.
* *conversion.loadflow.engine*: default engine selected in the conversion view.
* *conversion.modelicaNetwork.onlyMainConnectedComponent*: this is a boolean parameter (true or false) that indicates if the conversion engine uses only main AC-connected components.
* conversion.fullModelInitialization.engine: default modelica engine selected in the conversion view.
* *simulation.dynamicSimulation.engine*: default modelica engine selected in the simulation view.
* *simulation.dynamicSimulation.stopTime*: indicates the stop time for the dynamic simulation.
* *simulation.dynamicSimulation.stepsPerSecond*: number of intervals per second for the simulation.
* *simulation.dynamicSimulation.createFilteredMAT*: this is a boolean parameter (true or false) that indicates if the tool will create a .MAT file with a list of filtered variable.
* *compareLoadflows.loadflow.enforceGeneratorsReactiveLimits*: this is a boolean parameter (true or false) that indicates if the loadflow engine should enforce generators reactive limits.
* *swtoswValidation.source.stepSize*: indicates the step in seconds used to validate.
* *swtoswValidation.validation.thrmse*: Indicates the limit for highlighting the RMSE value. The values below the thresholds are shown in green in the GUI and those above it in red.
* *swtoswValidation.validation.thrd*: Indicates the limit for highlighting the Relative Difference value. The values below the thresholds are shown in green in the GUI and those above it in red.
* *swtoswValidation.validation.thad*: Indicates the limit for highlighting the Absolute Difference value, uses the green color for values below the limit and red for values above. Same colors as before.
* *swtoswValidation.chart.mode.activeEquidistantMode*: this is a boolean parameter (true or false) that indicates if the curves use an equidistant algorithm to select which values are showed.
* *swtoswValidation.chart.equidistantValues.maxValuesDisplayed*: indicates the maximum number of values to show in the curves.
* *swtoswValidation.chart.differenceValues.tolerance*: indicates the minimum difference between the last showed value and the candidate value to be selected.

1. **import.properties**

Configuration files for iPST static network import module[[22]](#footnote-22).

Configuration files for iPST static network import module.

# 8.- Command line reference

This section describes the command line tools available in PSM that allow the users to run the main tasks from the command line.

Running the script psm (in the PSM directory) the list of available commands and a small description is given.

Usage: psm COMMAND [ARGS]

Available commands are:

Case Validation:

casevalidation Validate a Modelica file

Data conversion:

iidm2mo Build a Modelica file from an IIDM network and a Dynamic Data Repository

Loadflow:

loadflow Loadflow computation from IIDM files.

Events Adder:

addevents Add events to a given case in a Modelica file building a new Modelica file with events.

Modelica dynamic simulation:

dynamicsim Run a dynamic simulation in a Modelica file using the specified Modelica dynamic simulation engine.

Static Network:

impnetwork Import an IIDM network from CIM files.

Details about these commands are described below. Each of these commands can be run using the script **psm** before the command, as shown in the following descriptions.

The command **impnetwork** allows the user to import an IIDM network from the CIM files. The syntax for this command is:

psm impnetwork --i-cim <CIM\_FILE> --o <IIDM\_FILE> [--help]

--i-cim <CIM\_FILE> Input: CIM file

--o <IIDM\_FILE> Output: IIDM file

The **loadflow** command allows to compute a loadflow on the given IIDM network and has the following syntax:

psm loadflow --engine <ENGINE> [--help] --i-iidm <IIDM\_INPUT\_FILE> -- <IIDM\_OUTPUT\_FILE>

--engine <ENGINE> Loadflow engine

--i-iidm <IIDM\_INPUT\_FILE> Input: IIDM file

--o <IIDM\_OUTPUT\_FILE> Output: IIDM file

The user can convert to Modelica a given IIDM network with dynamic data in a given Dynamic Data Repository using the command **iidm2mo**.

psm iidm2mo [--all-ac-connected-components] --i-ddr <DDR\_PATH> --engine <ENGINE> [--help] --i-iidm <IIDM\_FILE> --o <MODELICA\_FILE>

--i-ddr <DDR\_PATH> Input: Dynamic Data Repository path

--engine <ENGINE> Modelica dynamic simulation engine

--i-iidm <IIDM\_FILE> Input: IIDM file

--all-ac-connected-components All AC-connected components

--o <MODELICA\_FILE> Output: Modelica file

Where ENGINE is one of [Fake, Dymola, OpenModelica]

Adding events to a given Modelica file is also available from the command line, using the command **addevents**, with the following syntax:

psm addevents --i-ddr <DDR\_PATH> --i-events <EVENTS\_FILE> [--help] --i-iidm <IIDM\_FILE> --i-mo <MODELICA\_FILE> --o <MODELICA\_FILE>

--i-ddr <DDR\_PATH> Input: Dynamic Data Repository path

--i-events <EVENT\_FILE> Input: Events file

--i-iidm <IIDM\_FILE> Input: IIDM file

--i-mo <MODELICA\_FILE> Input: Modelica file

--o <MODELICA\_FILE> Output: Modelica file

The command **dynamicsim** performs a dynamic simulation in a given a Modelica file containing a dynamic system model:

psm dynamicsim --engine <ENGINE> [--filtered-mat] [--help] --i-mo <MODELICA\_FILE> --o <DIRECTORY\_PATH> [--steps-per-second <STEPS\_PER\_SECOND>] --stop-time <STOP\_TIME>

--engine <ENGINE> Modelica simulator engine

--filtered-mat Create a .MAT file with filtered variables

--i-mo <MODELICA\_FILE> Modelica file

--o <DIR\_PATH> Output: Directory path for the dynamic simulation results

--steps-per-second <STEPS\_PER\_SECOND> Number of steps per second

--stop-time <STOP\_TIME> Simulation stop time in seconds

Where ENGINE is one of [Dymola, OpenModelica]

The last command available is casevalidation that allows users to perform the validation of a given case.

psm casevalidation [--help] --i-mapping <MAPPING\_FILE> --i-mo <MODELICA\_FILE> --i-reference <REFERENCE\_FILE> [--o <DIR\_PATH>] [--step-size <STEP\_SIZE>] [--write-values <BOOLEAN>]  
 --help display the help and quit  
 --i-mapping <MAPPING\_FILE> Input: Mapping file  
 --i-mo <MODELICA\_FILE> Input: Modelica file  
 --i-reference <REFERENCE\_FILE> Input: Reference file  
 --o <DIR\_PATH> Output: Directory path for results  
 --step-size <STEP\_SIZE> Step size  
 --write-values <BOOLEAN> Write diff files

These commands allow the user to generate a power system network in Modelica from CIM files containing static data and a DDR containing dynamic data. A loadflow and a dynamic simulation, including events, can be prrformed.

In order to show how the command line tools work, a use case using the IEEE14 will be shown below. In the following, all commands are written as if the user is located on the first level of PSM directory, thus the path to ieee14 case directory ./data/test/ieee14/ is provided in all of them.

The steps to follow are:

1. **Import the IEEE14 case from CIM to IIDM.**

psm impnetwork --i-cim ./data/test/ieee14/ieee14bus\_EQ.xml --o ./data/test/ieee14/ieee14.xiidm

This command creates an xiidm file with the imported IIDM network.

1. **Compute a loadflow in the IIDM case.**

psm loadflow -i-iidm ./data/test/ieee14/ieee14.xiidm --o ./data/test/ieee14/ieee14\_with\_lf.xiidm --engine HELM-Flow

In case that the loadflow computation has been computed correctly this command generates an IIDM file with the Network that contains the loadflow results.

1. **Convert the IIDM with a DDR to Modelica.**

psm iidm2mo --i-ddr ./data/test/ieee14/ddr -i-iidm ./data/test/ieee14/ieee14\_with\_lf.xiidm --engine OpenModelica --all-ac-connected-components --o ./data/test/ieee14/ieee14.mo

After running this command the user will find a Modelica file in the ieee14/ case directory, from the conversion performed using imported IIDM network and the dynamic data from the DDR.

1. **Add events to the Modelica system.**

For adding a bus fault in the bus Bus\_\_\_\_4\_TN (R=0.5 p.u. and X = 0.5 p.u.) between time t = 1.0 s and t = 1.2 s, the user should create a \*.txt file (in this cale the file is named event\_1.txt) with the information for events. The event that will be added is defined in the file event\_1.txt as shown below:

BusFault,\_BUS\_\_\_\_4\_TN,R=0.5,X=0.5,endTime=1.2,startTime=1.0

To add this events to the Modelica base case the following command should be run:

psm addevents --i-ddr ./data/test/ieee14/ddr --i-events ./data/test/ieee14/event\_1.txt --i-iidm ./data/test/ieee14/ieee14\_with\_lf.xiidm --i-mo ./data/test/ieee14/ieee14.mo -o ./data/test/ieee14/ieee14\_event\_1.mo

Using the above command the user could add events to any Modelica file, i.e., to a base case Modelica network or to a Modelica file with other events added previously. For adding a line fault to the Modelica file generated with the command above, first the event added is defined in the file event\_2.txt:

LineFault,\_BUS\_\_\_\_2-BUS\_\_\_\_3-1\_AC,faultLocalization=0.5,Rfault=0.2,Xfault=0.2,startTime=2.5,endTime=2.7

Then, the user can use the same command used before for adding events, putting as input the Modelica file ieee14\_event\_1.mo.

psm addevents --i-ddr ./data/test/ieee14/ddr --i-events ./data/test/ieee14/event\_2.txt --i-iidm ./data/test/ieee14/ieee14\_with\_lf.xiidm --i-mo ./data/test/ieee14/ieee14\_event\_1.mo -o ./data/test/ieee14/ieee14\_event\_1\_2.mo

The file ieee14\_event\_1\_2.mo contains the Modelica network with a bus fault and a line fault.

1. **Run a dynamic simulation in Modelica.**

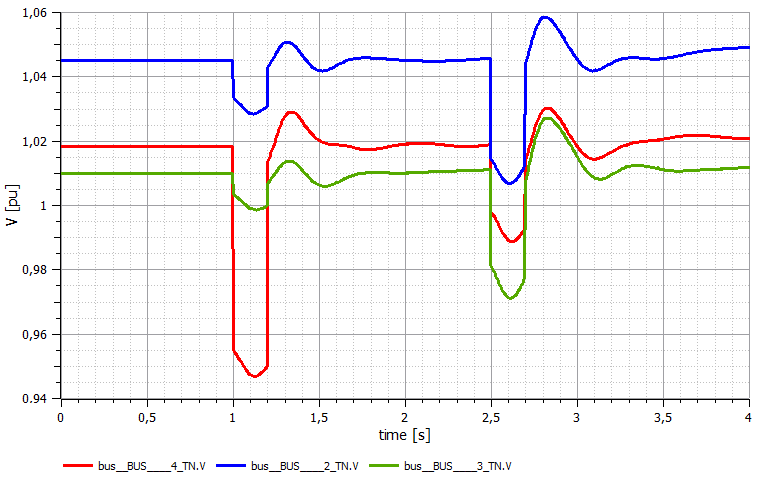
The file ieee14\_event\_1\_2.mo contains a Modelica network with two events included: a bus fault and a line fault. A dynamic simulation can be run on this netwrok using the following command:

psm dynamicsim --engine OpenModelica --filtered-mat --i-mo ./data/test/ieee14/ieee14\_event\_1\_2.mo --stop-time 4 --steps-per-second 100 --o ./data/test/ieee14/output\_data/

This command generates in the directory path set as output a set of files with the dynamic simulation results, listed below:

* *ieee14bus\_withEvents\_withEvents.mo* → The Modelica file that has been simulated.
* *ieee14bus\_withEvents\_withEvents.log* → Log file with the log of the dynamic simulation.
* *ieee14bus\_withEvents\_withEvents\_res.mat* → MAT file with all the simulation results.
* *ieee14bus\_withEvents\_withEvents\_res.csv* → A CSV file that contains all the simulation results in the MAT file.
* *ieee14bus\_withEvents\_withEvents\_res\_filtered.mat* → This file contains a set of the results variables (defined by the user in the configuration files) and it is generated only if the option --filtered-mat is set. This file has only sense if the user want to filter the list of variables from the simulation results.
* *iPSL.mo* → the library used to simulate the case.

Opening the filtered MAT file using OpenModelica or Dymola, as shown in the figure below, the curves match with curves in the Example 2 of the Tutorial section.



# 9.- Dynamic Data reference

A PSM DDR (Dynamic Data Repository) is a container for the dynamic modelling data required to perform the conversion of a certain static system. Dynamic modelling data can be grouped into the following sets:

* A library of predefined power system components in Modelica. Such a library can be the iPSL available with PSM tool, as well as a library or individual models provided by the user.
* Configuration data describing how the objects of the static system model should be mapped to components of the dynamic models library. In more detail:
  + A mapping between static network model objects and dynamic system model components.
  + A definition of the connections that must be created between dynamic system model components.
  + The parameters that must be used to instantiate all the dynamic system model components.
* Configuration data on which full model initialization (individual model simulations) must be run and how its results must feed the complete dynamic system model.
* Configuration data on how static network equipment attributes and power flow results should feed the dynamic system model components.

The next section presents the structure of the Modelica documents that will allow describing dynamic system models. After that, details of DDRs implemented using XML files are described.

## 9.1.- Modelica files

The only Modelica objects that are created for building the dynamic system model output are models, which refer to components defined in the iPSL or other user-defined models, and connections, that link these models between them according to the current topology of the network. The data model for this information is very simple:

* A model has:
  + A type name (the name of a component),
  + an identifier,
  + a set of instantiation arguments, and
  + a set of (public) variables.
* A connection has:
* Exactly two model identifiers and two variable names, one for each model.

The Modelica output that contains the dynamic system model will be a Modelica text file with a well-defined structure:

|  |
| --- |
| **within**;  **model** system\_model\_id  Model1Type model1\_id (  model1\_arg1\_name = model1\_arg1\_value,  model1\_arg2\_name = model1\_arg2\_value,  ...  );  ...  ModeljType modelj\_id (  modelj\_arg1\_name = modelj\_arg1\_value,  modelj\_arg2\_name = modelj\_arg2\_value,  ...  );  ModelkType modelk\_id (  modelk\_arg1\_name = modelk\_arg1\_value,  modelk\_arg2\_name = modelk\_arg2\_value,  ...  );  ...  **equation**  **connect** (model1\_id.model1\_varj1\_name, modelj\_id.modelj\_var11\_name);  **connect** (model1\_id.model1\_varj2\_name, modelj\_id.modelj\_var12\_name);  ...  **connect** (model1\_id.model1\_vark1\_name, modelk\_id.modelk\_var11\_name);  **connect** (model1\_id.model1\_vark2\_name, modelk\_id.modelk\_var12\_name);  ...  **end** system\_model\_id; |

## 9.2.- Dynamic Data Repository XML files

PSM defines its Dynamic Data Repositories using XML files. Two types of XML files are used: dynamic definitions files (DYD files) contain mapping definitions; and parameter files (PAR files) contain sets of parameters for dynamic models.

XML files with DYD extension define the mapping between static objects and dynamic model components. They also define which dynamic models should be used to introduce events in the dynamic system.

* The static model of a given piece of equipment is mapped to a dynamic model built from a set of dynamic components available in the library. Identifiers for static model objects and dynamic model components are given in the mapping.
* Connections between dynamic model components are defined explicitly using the dynamic model identifiers and its variables.
* Different mappings can be specified for dynamic system model simulation and for full model initialization.
* Dynamic model components require a set of parameters to be instantiated. They can be provided inline in the model mapping definition or as a reference to an external parameters file[[23]](#footnote-23).
* Points of interconnection between dynamic models must be specified so that the whole system dynamic model can be built based on the topology of the case being converted.
* Events specify a dynamic model and how it should be injected in the dynamic system model: either replacing the dynamic model of the affected equipment, or adding new components to its dynamic model.

XML files with PAR extension contain parameter sets for the instantiation of dynamic models.

* Every PAR file can contain multiple parameter sets.
* Each parameter set is given a unique identifier to be referenced from the DYD files.
* A parameter set contains a list of parameters. A parameter can be a variable or a reference. A variable has a name and a value. A reference has a name and a pointer to a data item available during the conversion process: an attribute of the static network model, a result from the power flow computation, or a result from the full model initialization.

A dynamic model mapping can be specified at three different levels. From more specific to more generic:

* A dynamic model mapping is defined for a specific static identifier.
* A dynamic model mapping is defined for a group of similar static equipment through an association. Example: all generators of a given set. Currently, groups must be defined based on static identifiers of the elements that should be included.
* A dynamic model is defined for all the elements of a static type. Example: all transmission lines are mapped to the same dynamic model.

The most specific dynamic model will be selected for a given static network equipment, if multiple options are available (PSM first look if a mapping exists for the static ID, then it checks if the equipment is included in some association, and then it checks the default to type-based mapping). The process to obtain the specific Modelica model corresponding to a given static element would be:

1. Look-up for a specific mapping defined for static identifier,
2. If not found, look-up for a group where the element is included,
3. If not found, look-up for a mapping defined for the static type of the element.

# 10.- iPSL Documentation

PSM version 1.0.0 has been built with the iTesla Power System Library release v1.1.1 published on iPSL GitHub repository:

<https://github.com/itesla/ipsl/releases>

All information regarding the Modelica library iPSL can be found at this repository.

As previously commented, the library is contained in PSM installation folder located at ./data/library/iPSL/

1. Common Information Model (CIM): <https://www.entsoe.eu/major-projects/common-information-model-cim/Pages/default.aspx> [↑](#footnote-ref-1)
2. Modelica language and the Modelica Association: <https://www.modelica.org/> [↑](#footnote-ref-2)
3. iTesla Power Systems Tools (iPST) project: <http://www.itesla-pst.org/> [↑](#footnote-ref-3)
4. iTesla Power Systems Library on GitHUb: <https://github.com/itesla/ipsl> [↑](#footnote-ref-4)
5. OpenModelica is an open source Modelica environment developed by the Modelica Association

   <https://www.openmodelica.org/> [↑](#footnote-ref-5)
6. Dymola is a commercial tool developed by Dassault Systèmes

   <https://www.3ds.com/products-services/catia/products/dymola/latest-release/> [↑](#footnote-ref-6)
7. PSM needs JRE 1.8 and JavaFX. In Java Oracle, the JavaFX comes integrated with JRE but if another Java is used, the JavaFX must be downloaded separately. [↑](#footnote-ref-7)
8. More info on HELMTM-Flow product at <http://elequant.com/products-and-solutions/helm-flow/> [↑](#footnote-ref-8)
9. RTE Hades loadflow <http://www.rte.itesla-pst.org/>, is only available for linux. [↑](#footnote-ref-9)
10. Power Systems Test Case Archive from the University of Washington <http://www2.ee.washington.edu/research/pstca/> [↑](#footnote-ref-10)
11. Details about configuration files can be found in the Configuration files section. [↑](#footnote-ref-11)
12. The variables filtered by default are the required ones for the case validation (see Validation section in User interface reference of this document) [↑](#footnote-ref-12)
13. IEEE14 case contains an additional DDR called ddr\_fixed\_injections/ which corresponds to a scenario of the IEEE14 case where all generators have been replaced by a fixed injection. This is provided as an example for the user to see how this replacement can be implemented in all generators.. [↑](#footnote-ref-13)
14. In PSM windows version the loadflow comparison section is deactivated because Hades loadflow is not available for windows. [↑](#footnote-ref-14)
15. Power Systems Test Case Archive from the University of Washington

    <http://www2.ee.washington.edu/research/pstca/> [↑](#footnote-ref-15)
16. Only available for PSM version for Linux. [↑](#footnote-ref-16)
17. This \*.xiidm file contains the IIDM networks with the results from Load Flow computation, If now Load flow is computed this file contains the initial values from CIM. [↑](#footnote-ref-17)
18. Eurostag is a commercial tool from Tractebel Engineering <http://www.eurostag.be/> [↑](#footnote-ref-18)
19. PSS/E is commercial tool from SIEMENS <http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/software-solutions/> [↑](#footnote-ref-19)
20. Different step sizes must be chosen in order to make sure that the system runs correctly with a fixed time step in the reference software. [↑](#footnote-ref-20)
21. A short overview of simulation flags for OpenModelica engine can be found at

    <https://www.openmodelica.org/doc/OpenModelicaUsersGuide/latest/simulationflags.html> [↑](#footnote-ref-21)
22. More info at iPST GitHub repository: <https://github.com/itesla/ipst> [↑](#footnote-ref-22)
23. **Important Note**: all parameters for a given dynamic model should be defined in one single file, either in the models file or in the parameter file. If some parameters are defined in the models file and others in the parameters file, PSM will not take the complete list of needed parameters. [↑](#footnote-ref-23)