Web Energy Services and Tools

Web Energy Services and Tools (WEST) allow creation of an FMU server to deliver the FMU in all confidentiality and security. The adaptive interface provides the possibility to do simulation with FMU both in model exchange and co-simulation mode. Upon finalization of installation, user can deposit new FMU into server and can get access to them via web service. The tool now supports FMI 1.0. A new version supporting FMI 2.0 is under development. The tool is open source and can be downloaded from:

- https://sourceforge.net/projects/westsandbox (Contributors need to be added by an admin).
- https://erigrid.eu/wp-content/uploads/2018/10/python_django_west_webservices.rar

I. Introduction WEST – FMU as a service

The Functional Mock-Up Interface (FMI) allows the creation of interoperable complied C code of non-interoperable modelling tools. The Functional Mock-Up Unit (FMU), simulation "black box" that works independently of the simulation environment or language, is the core element of FMI. For instance, the utilization of FMU is not straight-forward and requires some coding knowledge. Moreover, not all popular simulation tools support FMU.

The WEST tool allows the deliverance of power system FMU library for co-simulation and model-exchange in a Software-as-a-Service (SaaS) architecture.

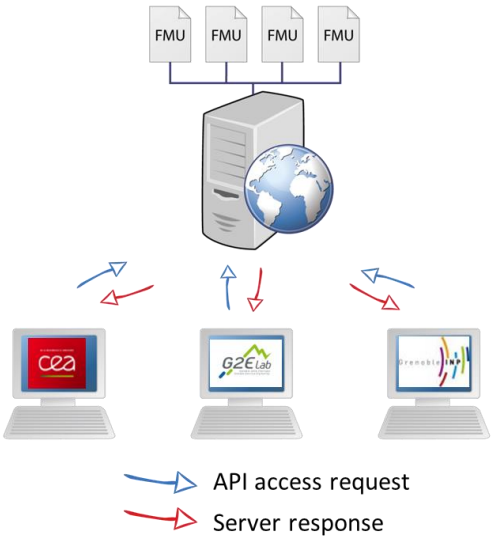


Figure 1 : FMMU-as-a-service approach in WEST.

The interest of this approach is:

- Remote Co-simulation via API.
- User-friendly GUI
- The FMU is executed on server. No installation is needed on client side, thus, no coding skill is required from the user. The architecture also provides better user experience as the server offers better performance and the possibility to instantiation and running multiple instance of FMU without worrying about performance.
- The FMU is located on server and the user has only access to the interface. Therefore, the FMU owner can share their models in all security and confidentiality (in terms of intellectual property).

The tool is actually developed on Django. Using PyFMI as solver, the application is compatible with FMI 1.0 and FMU CS. It is required that the FMU consists of available compatible platform information in order to proceed.

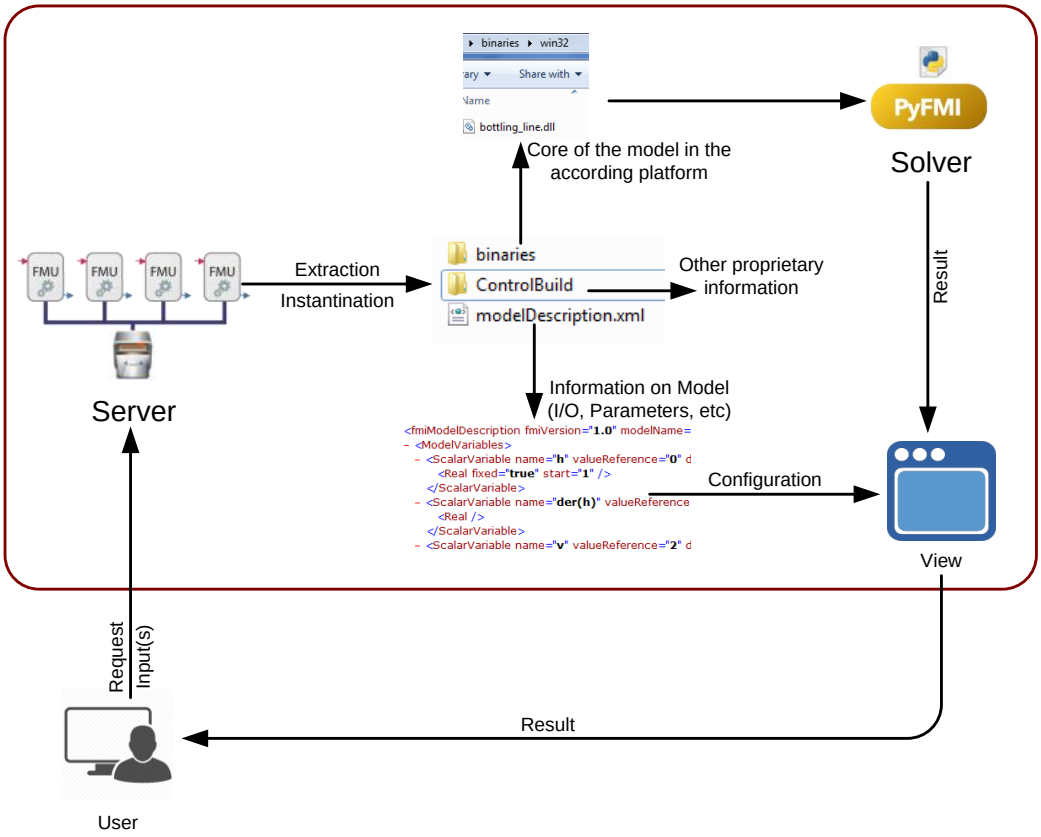


Figure 2 : The diagram of FMU processing in the WEST application

II. Installation

The WEST tool runs in server client mode. The FMU library and the core processing modules are located on the server side. The client accesses to the server via a web browser (Internet Explorer, Google Chrome or Mozilla Firefox).

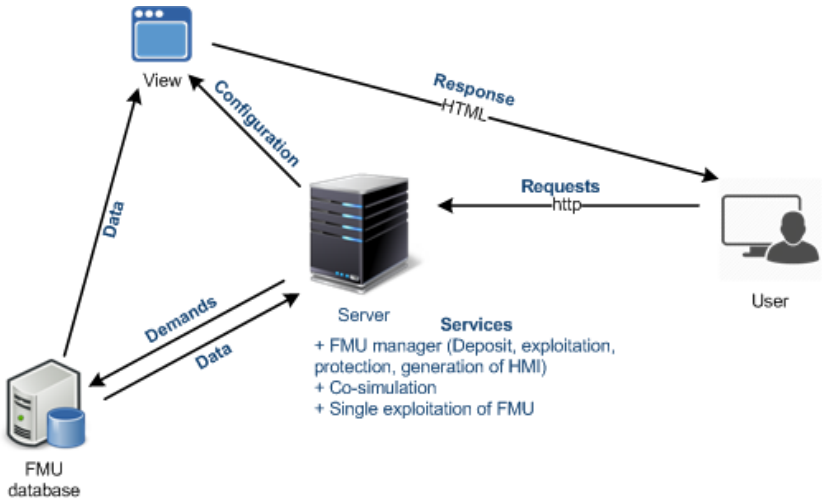


Figure 3: Structure of the application

The downloaded file should contain the following important elements:

- __init__.py indicates that the folder is a python module. This file is empty and must not be modified
- Views.py composed of functions that process user requests, modify the GUI and return server responses
- Models.py describing the classes that will format the models. Each attribute is a field in the database. Access and modification of the fields is then done by the instance of the class.
- The template folder.
- Forms.py contains forms generated by Django. In a similar way to models, a form is represented by a class and each attribute corresponds to a field. This makes it possible to secure the data sent by the user and automate their treatment
- Urls.py allows rooting queries by linking URLs to views
- Admin.py allows to manage the administration interface
- Settings.py which defines all the important settings of the administration
- Wsgi.py implements a global object that will allow the server to execute Python web applications. This file should not be modified

On server side, Python (2.7) and the following packages are required for the functionality of the WEST tool:

- PyFMI 2.3.1 for python 2.7.
- Assimulo 2.8
- Numpy 1.11.2
- FMI_Library 2.0.1
- Cmake
- Django 1.10.5
- Scipy 0.18.1
- Lxml 3.7.2
- PyCrypto 2.6
- Sundial
- Cython 0.25.2
- Optional: wxPython, matplotlib

Upon finishing the download and installing all these prerequisites, the user can start the server via the following methods:

- In **Window**: Run the script runserver.bat
- In **Linux**: from the python_django_west_webservices folder, execute the command: python manage.py runserver [IP addresse] : [port]

By default, the server is located at 127.0.0.1:8000. The user can get access to the server via this address. In order to grant access to other users to the server, the admin needs to deploy the server to a dedicated host (instead of just running the default configuration) and to allow http connection through the defined port. This requires however editing the manage.py file via a python editor.

The WEST application involves two main important aspects: the services accessible via the API and the services accessible via the web interface. These services are linked to the same FMU processing modules and same FMU solver, so they should give the same simulation results. In the following, we introduce these services.

III. Web interface

The web interface provides the users with a list of FMU and their available services: Execution, Getting variable and co-simulation, etc.

The user can run an available FMU in different settings and initial conditions, get and plot results via the web interface. The simulation interface groups together all the services available for the simulation of energy systems. The interface reacts dynamically with the server to have an answer instantly.

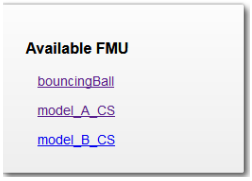


Figure 4: List of available FMU

To get access to available services for one FMU, the user clicks on the according FMU on the list. An instance of the said FMU is created.

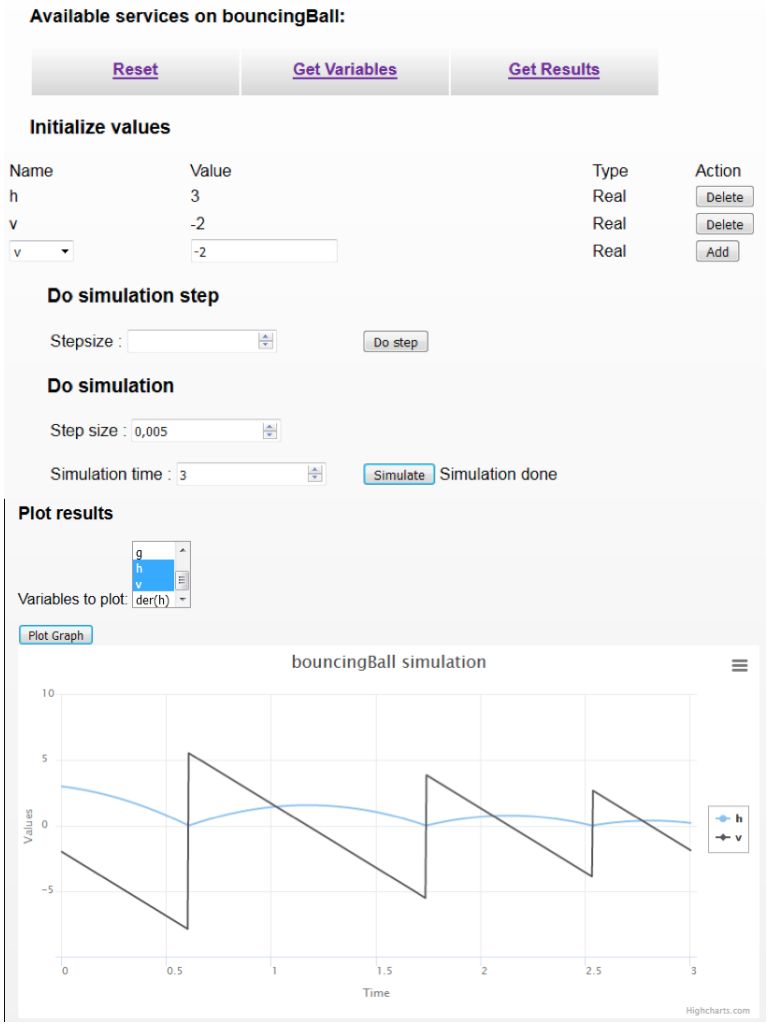
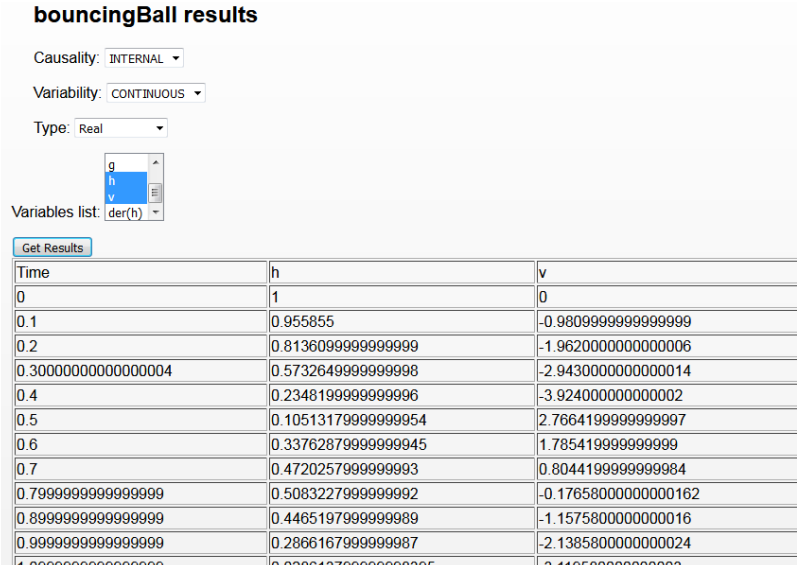


Figure 5: Simulation interface

The simulation can be done in two methods, for a defined time period or step by step. The user can reset the defined variables via the button Reset.

The simulation results can be gotten in form of tables or can be plotted.



Time	h	v
0	1	0
0.1	0.955855	-0.9809999999999999
0.2	0.8136099999999999	-1.9620000000000006
0.30000000000000004	0.5732649999999998	-2.9430000000000014
0.4	0.2348199999999996	-3.924000000000002
0.5	0.10513179999999954	2.7664199999999997
0.6	0.33762879999999945	1.785419999999999
0.7	0.4720257999999993	0.8044199999999984
0.7999999999999999	0.5083227999999992	-0.17658000000000162
0.8999999999999999	0.4465197999999989	-1.1575800000000016
0.9999999999999999	0.2866167999999987	-2.1385800000000024

Figure 6: Simulation results.

User can also set up a co-simulation framework of several FMU via the co-simulation interface, by defining the link (inter-dependency) among these models. The defined inter-dependencies are executed in top-down order, to avoid cyclic-dependency.

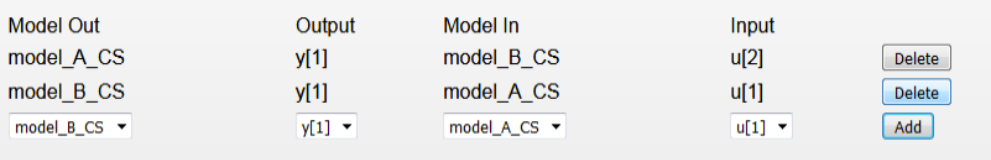


Figure 7: Setting up dependency in co-simulation interface.

After defining the interdependency, the co-simulation framework can be executed and the simulation results can be accessed from the according FMU's interface.

The application also provides the possibility to manage the FMU library (adding new FMU, executing and deleting). These services are available after user authentication.

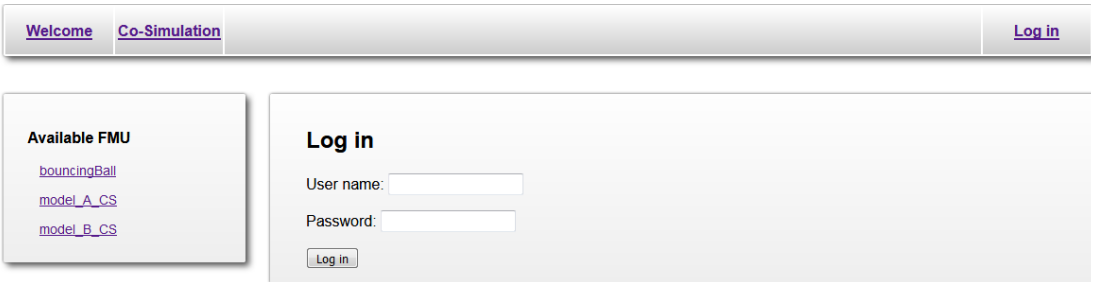


Figure 8: Authentication Interface

From the new Profile link, the user can add and delete their FMU to/from the server.

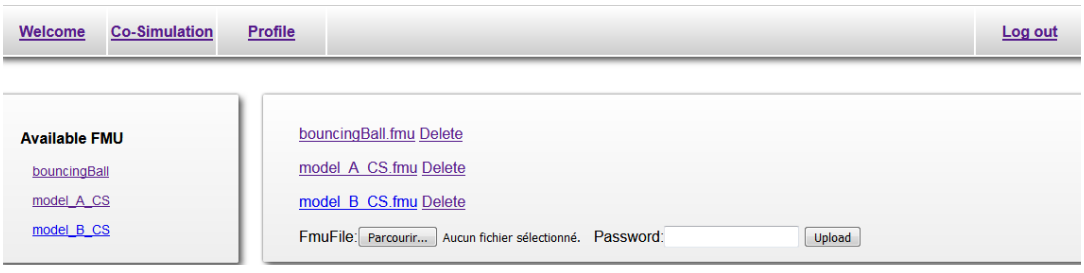


Figure 9: FMU management interface

IV. The API

Besides accessing the services via the graphical interface, the user can also choose to execute the services via the following URI:

- FMU Listing: http://localhost:8000/fmuWebService/{model-name}/getVariables/html
- Variable exploration: http://localhost:8000/fmuWebService/{model-name}/getVariables/html
- Set Initial values: http://localhost:8000/fmuWebService/{model-name}/initializeVariable?initValues=values
- Setting up values: http://localhost:8000/fmuWebService/{model-name}/setVectorVariables?setValues=values
- Do step: http://localhost:8000/fmuWebService/{model-name}/doStep?stepSize=value
- Simulation of a duration: http://localhost:8000/fmuWebService/{model-name}/simulate?simulationTime=value&stepSize=value
- Reset Model: http://localhost:8000/fmuWebService/{model-name}/reset or : http://localhost:8000/fmuWebService/resetAll
- FMU Linking for co-simulation: http://localhost:8000/fmuWebService/setCosimLink/?modelIN={model-name1}&variableIn={variable1}&modelOut={model-name2}&variableOut={variable2}
- Delete exiting FMU Link: http://localhost:8000/fmuWebService/delCosimLink/?modelIN={model-name1}&variableIn={variable1}&modelOut={model-name2}&variableOut={variable2}
- Get Co-sim links: http://localhost:8000/fmuWebService/getCosimLink/
- Do step Co-simulation: http://localhost:8000/fmuWebService/doCosimStep/?stepSize=value
- Co-simulation in a duration: http://localhost:8000/fmuWebService/cosimulate/?simulationTime=value&stepSize=value
- Get results: http://localhost:8000/fmuWebService/{model-name}/getAllScalarOutputs/html
- Plot results: http://localhost:8000/fmuWebService/{model-name}/plotResult?variableNames=values
- Vectorial input with API: http://localhost:8000/fmuWebService/bouncingBall /setInputVariables?setValues={"h":[3,4,5],"v":[1,0,-1]}

These URI are very useful in case the user would like to set up an automatic connection to the server or a co-simulation framework with a model which is not on the server.