D6.2 – User guide for TradeRES models and tools (D6.2.1)

Deliverable number: *D6.2*

Work Package: *WP6*

Lead Beneficiary: *ISEP*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 864276

Author(s) information (alphabetical)		
Name	**Organisation**	**Email**
Christoph Schimeczek	DLR	christoph.schimeczek@dlr.de
Filipe Sousa	ISEP	ffeso@isep.ipp.pt
Gabriel Santos	ISEP	gajls@isep.ipp.pt
Hugo Algarvio	LNEG	hugo.algarvio@lneg.pt
Ingrid Sanchez Jimenez	TUDelf	i.j.sanchezjimenez@tudelft.nl
Niina Helistö	VTT	niina.helisto@vtt.fi
Ricardo Hernandez Serna	TNO	ricardo.hernan-dezserna@tno.nl
Tiago Pinto	ISEP	tcp@isep.ipp.pt

Acknowledgements/Contributions		
Name	**Organisation**	**Email**
Débora de São José	ISEP	drj@isep.ipp.pt

Document information			
Version	**Date**	**Dissemina-tion Level**	**Description**
1.0	28.02.2021	Public	User guide and information to support users when executing the tools developed by the project.

Review and approval		
Prepared by	**Reviewed by**	**Approved by**
Gabriel Santos (ISEP)	Russel Wolff (bitYoga)	Ana Estanqueiro (LNEG)

Disclaimer

The views expressed in this document are the sole responsibility of the authors and do not necessarily reflect the views or position of the European Commission or the Innovation and Network Executive Agency. Neither the authors nor the TradeRES consortium are responsible for the use which might be made of the information contained in here.

Executive Summary

The present deliverable, as part of task 6.2, aims to provide a guide for users to easily access the tools that are being used and developed in the scope of the project. Considering that stakeholders' engagement is a crucial point in this project as an enabler for suitable testing, validation and refinement of the models developed by the project, it is essential that the necessary means to support this engagement are created.

The relevant stakeholders identified in Task 6.1 and described in D6.1 [1] have different levels of expertise regarding the topics addressed by project. Consequently, a set of materials will be provided to facilitate the engagement of different types of players, considering their characteristics and levels of expertise; namely: video tutorials, webinars, meetings (such as conference and seminars) and user guides.

The user guides developed support users when executing the tools developed by TradeRES project. These are presented in this deliverable as part of a report that includes a brief overview of the models for which user guides that have been created.

Table of Contents

Acronyms

ABM agent-based model

AMIRIS - Agent-based Market model for the Investigation of Renewable and In-tegrated energy Systems

BRPs - Balance Responsible parties

CAPEX Capital Expenditure

CM - Capacity Mechanisms

COMPETES-EMLab - COMPetition in Electric Transmission and Energy Simula-tor & Energy Modelling Laboratory

EM – Electricity Market

EMLab - Energy Modelling Laboratorys

EU – European Union

GAMS - General Algebraic Modeling System

MASCEM - Multi-Agent Simulator for Competitive Electricity Markets

MIBEL - Iberian Electricity Market

MPT - Marginal Pricing Theory

NTC - Net Transfer Capacities

OPEX Operational Expenses

O&M Costs Operation and Maintenance Costs

PT - Portuguese

RES - Renewable Energy Sources

RESTrade - Multi-agent Trading of Renewable Energy Sources

SP – Spanish or Portuguese

WP – Work Package

1. Introduction

This report is a collection of user guides for the AMIRIS, Backbone, EMLab, COMPETES-EMLab, MASCEM and RESTrade models, a group of innovative electricity market designs that are being developed/improved and tested in the TradeRES project.

The goal is to engage different stakeholders in testing the open-access tools developed in WP4 and collect feedbacks, information, and recommendations for ~100% RES market designs [2]. However, the relevant stakeholders identified in Task 6.1 and described in D6.1 [1] have different levels of expertise in the topics covered by the project. On this basis, a set of materials are provided as part of Task 6.2 to facilitate their participation, considering their characteristics and level of expertise. The materials will be made available in deliverables D6.2 and D6.3, and consists of video tutorials, webinars, conferences (such as conferences and seminars) and user guide. All the materials are publicly available in the project's website [3].

The user guides are designed to facilitate stakeholders' involvement in shap-ing market design and testing and validating the open-access market tools, as their feedback and suggestions will be used to improve the project models and tools, as well as to define recommendations for market design with ~100% RES [2].

All the user guides are organized as follows: firstly, an overview of the model is presented, then the necessary inputs, followed by the outputs that can be ob-tained, then steps required to run the model and, finally, a section with links to learn more about the model and the main contacts.

This report presents an overview of the six models, and due to the need to continuously update the user guides, these are independent documents and, therefore, presented in Annex 1-6.

2. TradeRES Models

This report provides stakeholders easy access to the tools developed in WP4 and the findings of WP5 through the provision of six user guides for the models described in the following sections.

2.1 AMIRIS

The open **A**gent-based **M**arket model for the **I**nvestigation of **R**enewable and **I**ntegrated energy **S**ystems is a model that computes electricity prices endogenously based on the simulation of strategic bidding behavior of prototyped market actors. Simulations with AMIRIS thus enable the investigation of the influence of political framework conditions on the behavior and profitability of energy market actors, considering different marketing paths, as well as the quantification of the influence of uncertainties and socio-economic decision aspects of individual actors on energy markets [4]. AMIRIS user guide is available in **Erro! A origem da referência não foi encontrada.**.

2.2 Backbone

Backbone is a generic energy network optimization tool that was designed to be highly adaptable in different dimensions: temporal, spatial, technology representation and market design. Backbone can represent stochastics with a model predictive control method, with short-term forecasts and longer-term statistical uncertainties. It can also support multiple different models due to the modifiable temporal structure and varying lengths of the time steps [5]. Backbone user guide is available in **Erro! A origem da referência não foi encontrada.**.

2.3 EMLab

Energy **M**odelling **Lab**oratorys is an agent-based model with the purpose of investigating the long-term effects of climate and energy policies. The agents are power companies with limited information about the future system and make imperfect investment decisions. EMLab can also simulate a CO_2 market and capacity mechanisms [6]. EMLab user guide is available in **Erro! A origem da referência não foi encontrada.**.

2.4 COMPETES-EMLab

COMPetition in **E**lectric **T**ransmission and **E**nergy **S**imulator & **E**nergy **M**odelling **Lab**oratory model covers 28 EU Member States and some non-EU countries. It is a power system optimization and optimal dispatch model that seeks to minimize the total power system costs of the European power market. It can be used to perform simulations for two least-cost capacity expansion to optimise generation and transmission capacity additions and day-ahead markets, through least-cost planning and dispatch of generation and demand [7]. COMPETES-EMLab user guide is available in **Erro! A origem da referência não foi encontrada.**.

2.5 MASCEM

The **M**ulti-**A**gent **S**imulator for **C**ompetitive **E**lectricity **M**arkets is a modelling and simulation tool designed to study complex restructured electricity market operations by modeling the complex dynamic market players, including their interactions and the collection of medium/long-term data and experience, to support participants in making decisions based on to their characteristics and goals [8]. MASCEM user guide is available in **Erro! A origem da referência não foi encontrada.**.

2.6 RESTrade

The Multi-agent Trading of Renewable Energy Sources includes different models development in the TradeRES project, the traditional and power and energy reserve markets. The module supports traditional dispatchable power plants, variable renewables and demand actors to participate in system balancing, i.e. automatic and manual frequency recovery reserve markets. Additionally, it uses both marginal pricing theory and pay-as-bid schemes to define prices in these markets. RESTrade user guide is available in **Erro! A origem da referência não foi encontrada.**.

3. Conclusions

This deliverable presented a small description of the different models and pro-vided a user guide for each one of them, which are available in Annex 1-6. The goal was to provide stakeholders easy access and to engage different stakeholders in testing the open-access tools from WP4 and collect feedbacks, infor-mation, and recommendations to im-prove the project models and tools.

The user guides are part of the Task 6.2, from WP6, and will be followed by de-liverable D6.3 which corresponds to a set of internet video tutorials and webinars that will be provided to complement and enrich the information already available in the user guides. The user guides will undergo progressive updates as the models are further developed. New guides will also be created as new flows, linking different models, are developed. User guides and support manuals for several of the tools presented, are already available online, and, these materials will be used for stakeholders' engagement and to support the specific tutorials and webinars to be organized under T6.2.

References

[1] TradeRES project consurtium, Deliverable 6.1 “D6.1 – Creating the basis for stakeholders engagement” 2021.

[2] The Grant Agreement nº 864276 and all its annexes — TradeRES

[3] TradeRES - New Markets Design & Models for 100% Renewable Power Systems. Available in https://traderes.eu. Accessed in February 7th 2022.

[4] AMIRIS - The open agent-based electricity market model. Available in https://dlr-ve.gitlab.io/esy/amiris/home/. Accessed in February 7th 2022.

[5] Backbone. Available in https://gitlab.vtt.fi/backbone/backbone. Accessed in February 7th 2022.

[6] EMLab - Energy Modelling Laboratory. Available in http://emlab.tudelft.nl/. Accessed in February 7th 2022.

[7] COMPETES - COMPetition in Electric Transmission and Energy Simulator. Available in https://repository.tudelft.nl/islandora/object/uuid:63691862-9a26-4df3-b6dd-57bed6c9d8a5?collection=education. Accessed in February 7th 2022.

[8] MASCEM - Multi-Agent Simulator for Competitive Electricity Markets. Available in http://www.mascem.gecad.isep.ipp.pt/overview.php/. Accessed in February 7th 2022.

Annex 1 – AMIRIS User Guide

AMIRIS

Agent-based Market model for the Investigation of Renewable and Integrated energy Systems

Overview

AMIRIS is a next-generation tool to dissect the complex questions with respect to future energy markets, their market design, and energy-related policy instruments. The model computes electricity prices based on the simulation of strategic bidding behavior of prototyped market actors. This enables AMIRIS to not only consider marginal prices but also support instruments and uncertainties. Figure 1 shows agents and their associated flows of information, energy, and money modeled in AMIRIS.

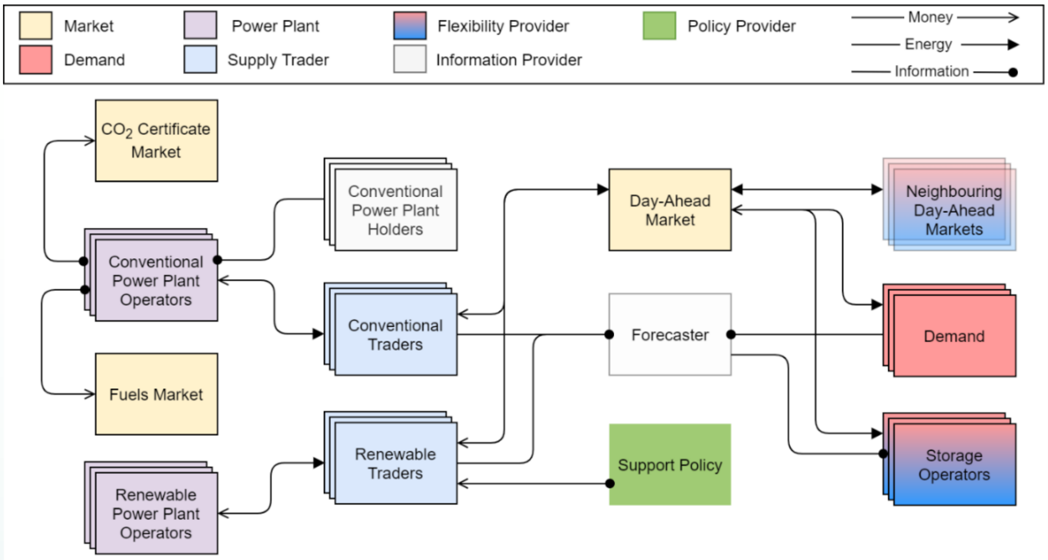

Figure 1 Overview of agents and their interactions in AMIRIS (open version)

Actors are represented as agents and can be roughly divided into six classes: Power plant operators, traders, marketplaces, policies, demand, and flexibility option facilities. Power plant operators provide generation capacities to traders, but do not trade on the markets themselves in the model. Bidding and operation decisions are conducted by traders in pursuit of, e.g., profit maximization strategies. Marketplaces serve as trading platforms and determine prices. Policies define a regulatory framework, which impacts the decisions of other agents. Demand agents as well as flexibility option facilities, e.g., storage facilities, trade directly in the electricity market.

Inputs

AMIRIS is configured via human-readable YAML files: "scenario.yaml" and "fameSetup.yaml". The latter is covered in the section "How to run AMIRIS". The scenario file is split into several sections:

The "Schema"-section specifies which types of agents exist, which attributes they have, which they require and how they can interact with other agents. This section must not be changed. Typically, a separate schema file is provided to describe simulation capabilities and requirements. The special YAML loader of FAME-Io https://gitlab.com/fame-framework/fame-io allows to split and reuse YAML files via the "!include"-command.

The "Agents"-section defines which agents are to be created in the simulation and how they are parameterized. Each agent requires at least a type and unique id. Additional attributes might be required, depending on its type. Attributes can be a single value, a list of values, an externally specified time series in CSV format, a group of sub-attributes or even a list of grouped sub-attributes. The type of each attribute is specified in the Schema section. The full list of attributes cannot be stated in this document. Instead, please refer to the up-to-date pages at the AMIRIS-Wiki https://gitlab.com/dlr-ve/esy/amiris/amiris/-/wikis/Classes/Classes. There, a comprehensive list of all attributes and associated configuration options can be found for all agent types.

The "Contracts"-section configures the interactions between the agents. Each contract comprises a sender and receiver agent (identified by ID), a product type, as well as an initial execution time and execution interval (in seconds). Usually, changes to contracts are only required if agents are added to or deleted from the configuration. For this, groups of agents are defined that allow to add and remove agents conveniently with a change at a single point of the configuration. Please see the AMIRIS-Wiki for a list of all available products per agent type. Similar to the "Schema" section, "Contracts" are often extracted to separate files to keep the scenario file tidied up. FAME-Io's "!include" command can be used to load a multitude of other YAML files. See the AMIRIS-Examples project https://gitlab.com/dlr-ve/esy/amiris/examples for several examples of techniques to keep the configuration files neat and organized.

Contract configuration for AMIRIS is not trivial, since almost all actions within AMIRIS are controlled via contracts. Thus, one needs to know what actions create which data, which input is required by what action and which agent can provide such input. Therefore, instead of starting from scratch, please refer to the AMIRIS-Examples project to see several working examples of agents and related contracts.

Once the configuration is completed, “FAME-Io”, a Python tool, is used to convert configuration files into a single binary input file for AMIRIS. Please see the FAME-Io documentation for further instructions on its installation, execution, and command-line options.

Outputs

Each execution of AMIRIS creates a singular output file. Name and path to that file can be controlled via the “fameSetup.yaml” (see next section). The binary output file (protobuf) needs to be converted to human-readable form for interpretation. FAME-Io provides a script to perform that task, which will create a folder with CSV files: one file per agent type in the simulation. Figure 2 provides an example output in that format.

```
AgentId;TimeStep;TotalAwardedPowerInMW;ElectricityPriceInEURperMWH;DispatchSystemCostInEUR
1;599184000;52799.0;0.0;382017.32681056164
1;599187600;42766.0;0.0;382017.32681056164
1;599191200;50965.0;0.0;382017.32681056164
1;599194800;50546.0;0.0;382017.32681056164
1;599198400;39285.0;0.0;382017.32681056164
1;599202000;51366.0;0.0;382017.32681056164
```

Figure 2 Sample output of an agent of type “EnergyExchange” in CSV format

Each created output file features the columns “AgentId” and “TimeStep” at least. The first column refers to the ID of the agent as specified in the “scenario.yaml” file (see previous section). All outputs from agents of the same type are combined in the same file, although sorted by “AgentId”. The next column defines the simulation time at which the output was made by that agent. Please see https://gitlab.com/fame-framework/wiki/-/wikis/TimeStamp for a detailed description how FAME measures time. FAME-Io offers a function to convert time steps to time stamps – when using Spine toolbox (see next section), this is done automatically.

Depending on the type of agent, additional columns exist. Typically, the header row tells what value is depicted – including the unit in its name. Note that not all output columns are used in each time step, necessarily. Figure 3 provides an example for that case. There, outputs alternate between specifying offered and awarded power, which occurs at different times within the simulation.

```
AgentId;TimeStep;OfferedPowerInMW;AwardedPowerInMWH;ReceivedMoneyInEUR;CostsInEUR
500;599183998;9135.36;;;
500;599184002;;9135.36;0.0;55285.4799328693
500;599187598;9135.36;;;
500;599187602;;9135.36;0.0;55285.4799328693
500;599191198;9135.36;;;
500;599191202;;9135.36;0.0;55285.4799328693
500;599194798;9135.36;;;
500;599194802;;9135.36;0.0;55285.4799328693
```

Figure 3 Sample output of an agent of type “ConventionalPlantOperator”

A full list of all outputs cannot be given here. Please refer to the AMIRIS-Wiki to learn about all agent types and their outputs. You may also check the source code and inspect what is actually done to create the output. Simply search for uses of the "store(<ColumnName>, <Value>)" method in the particular class for the agent type you are interested in.

How to run it

AMIRIS is based on FAME, the "open Framework for distributed Agent-based Modelling of Energy systems" (see https://gitlab.com/fame-framework). Thus, it requires a Java Development Kit (version 8 or above), a Python installation (version 3.7 or above) and Apache Maven. AMIRIS can be either run standalone or within the Spine toolbox. Therein, all individual steps of input peparation, running and output conversion are joined into a single workflow. An experimental workflow to run AMIRIS within the Spine toolbox is availlable at https://github.com/TradeRES/toolbox-amiris-demo. However, usability and documentation is to be enhanced in the near future. Please see the AMIRIS Readme file https://gitlab.com/dlr-ve/esy/amiris/amiris/-/blob/main/README.md for a description on installing and running AMIRIS without Spine toolbox. The Spine toolbox workflow requires a packaged Java ARchive (JAR) file of AMIRIS including all dependencies. It can be easily obtained using Maven – please follow instructions in the AMIRIS Readme.

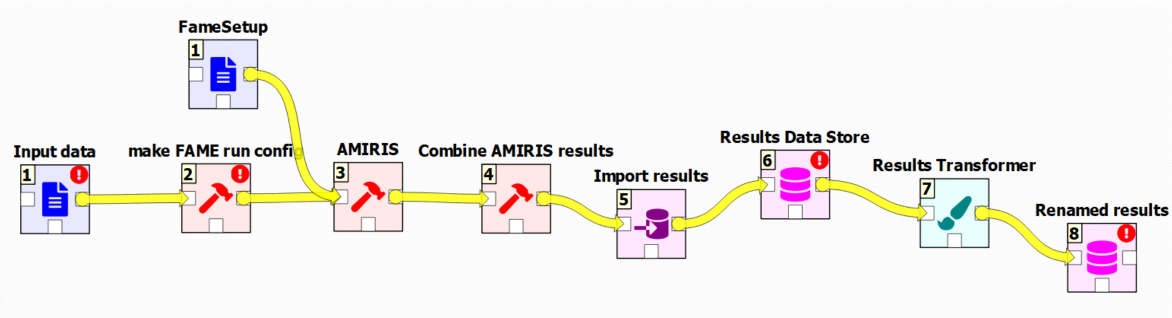

Figure 4 Workflow executing AMIRIS in Spine toolbox

Figure 4 illustrates the workflow steps in the Spine toolbox: two files (1) need to be provided to the workflow: the *scenario* definition and the *fameSetup* – both in YAML-format. Please see the section Inputs for a description of the scenario file. The workflow automatically calls FAME-Io (2) to translate the scenario into a single binary input file in protobuf format to be used by AMIRIS. When AMIRIS is run (3), the *fameSetup.yaml* file is read by FAME-Core. It defines file output parameters (see Table 1). It is best not to change the provided file. After AMIRIS is run, the result file is read (4), extracted into .csv files and imported (5) into the local SQL database (6). To comply with the TradeRES naming standards and assigning of time stamps to the results, these data are then transformed (7) and saved into another database section (8).

Table 1 Parameters in fameSetup.yaml

Parameter	Description
outputPath	Relative or absolute path to create the output file at
outputFilePrefix	Name of the output file
outputFileTimeStamp	True on default; if true, a time stamp is prepended to the output file
agentPackages	List of Java package names that contain classes derived from "Agent"
messagePackages	List of Java package names that contain classes derived from "DataItem"
portablePackages	List of Java package names that contain classes derived from "Portable"

Find out more

Nitsch, F. and Schimeczek, C. and Bertsch, V. (2021) “Back-testing the agent-based model AMIRIS for the Austrian day-ahead electricity market”. *Working paper*. doi: 10.5281/zenodo.5726738

Nitsch, F. and Deissenroth-Uhrig, M. and Schimeczek, C. and Bertsch, V. (2021) “Economic evaluation of battery storage systems bidding on day-ahead and automatic frequency restoration reserves markets”. *Applied Energy* (298). Elsevier. doi: 10.1016/j.apenergy.2021.117267.

Laura Torralba-Díaz et al. (2020) “Identification of the Efficiency Gap by Coupling a Fundamental Electricity Market Model and an Agent-Based Simulation Model”. *Energies*. Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/en13153920.

Frey, U. and Klein, M. and Nienhaus, K. and Schimeczek, C. (2020) “Self-Reinforcing Electricity Price Dynamics under the Variable Market Premium Scheme”. *Energies*. Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/en13205350.

Deissenroth, M. and Klein, M. and Nienhaus, K. and Reeg, M. (2017) “Assessing the Plurality of Actors and Policy Interactions: Agent-Based Modelling of Renewable Energy Market Integration”. *Complexity*, 2017, 7494313:1-7494313:24, doi: 10.1155/2017/7494313.

Main contacts

AMIRIS Home https://dlr-ve.gitlab.io/esy/amiris/home/

AMIRIS@Gitlab https://gitlab.com/dlr-ve/esy/amiris/amiris

AMIRIS@openMod https://forum.openmod.org/tag/amiris

FAME@Gitlab https://gitlab.com/fame-framework

FAME-Core@Maven https://mvnrepository.com/artifact/de.dlr.gitlab.fame/core

FAME-Io@PyPI https://pypi.org/project/fameio/

TradeRES
New Markets Design & Models for
100% Renewable Power Systems

Annex 2 – Backbone User Guide

Backbone

Overview

Backbone represents a highly adaptable energy systems modelling framework, which can be utilised to create models for studying the design and operation of energy systems, both from investment planning and scheduling perspectives. It includes a wide range of features and constraints, such as stochastic parameters, multiple reserve products, energy storage units, controlled and uncontrolled energy transfers, and, most significantly, multiple energy sectors. Both high-level large-scale systems and fully detailed smaller-scale systems can be appropriately modelled.

The framework has been implemented as the open-source Backbone modelling tool using General Algebraic Modeling System (GAMS). The tool minimizes the total investment and operating costs of the system. The formulation is based on mixed-integer programming and takes into account unit commitment decisions for power plants and other energy conversion facilities.

The adaptability of Backbone extends to several dimensions: temporal, spatial, technology representation and market design. Stochastic inputs can be represented with short-term forecasts and longer-term statistical uncertainties. It is possible to vary time step durations, select representative periods, and define rolling optimization structures. Due to the modifiable node-unit structure, multiple efficiency representation alternatives, and aggregation possibilities, technologies can be modelled with appropriate accuracy. From the market design perspective, Backbone supports, for example, different reserve requirement and provision configurations as well as gate closures.

Inputs

Main input data sets and parameters are given in the inputData.gdx file. Backbone sets and parameters are documented in the files 1b_sets.gms and 1c_parameters.gms, respectively, as well as 1a_definitions.gms, which describes hard-coded sets, such as parameter definitions, in more detail. Some sets and parameters are not in the input data – they are calculated from the input data.

The Backbone model is initialized by the modelsInit.gms file (a model definition file) in the input data folder. This file sets, for example, the temporal structure of the model. It often just points to a specific model version that is described by another file (for example, scheduleInit.gms). This way it is easier to change between different model definition versions. There are templates for different model versions in the Backbone/defModels folder.

Global system parameters to control the solution process are given in 1_options.gms and temporal index ranges are given in timeAndSamples.inc. Templates for these files are available in the main Backbone folder.

The main input data types are tabulated below.

Table 2

Output group	Main output types
Investments	Invested amount of each type of generation or other technology as well as transmission
Unit commitment and dispatch	Start-ups and shutdowns, online status, production or consumption, ramping, and reserve provision of of each unit
Transmission between nodes	Use of transmission links
Emissions	Emissions from fuel consumption
Costs	OPEX: fuel costs, other O&M costs, emission costs, penalty costs from inflexibilities

Outputs

Backbone writes results in the results.gdx file. Result parameters are described in 1d_results.gms. The main output data types a tabulated below.

Output group	Main output types
Investments	Invested amount of each type of generation or other technology as well as transmission
Unit commitment and dispatch	Start-ups and shutdowns, online status, production or consumption, ramping, and reserve provision of of each unit
Transmission between nodes	Use of transmission links
Emissions	Emissions from fuel consumption
Costs	OPEX: fuel costs, other O&M costs, emission costs, penalty costs from inflexibilities
	CAPEX: generation capacity and other technology investments, transmission link investments
Prices	Marginal cost of commodities (electricity, gas, heat, etc.)
Flexibility indicators	Loss of load, reserve shortages, spillage, VRE curtailment

Find out more

Backbone is available at https://gitlab.vtt.fi/backbone/backbone. The README file and Wiki pages https://gitlab.vtt.fi/backbone/backbone/-/wikis/home give more instructions for installing and using the tool. Backbone documentation and other publications are listed at https://gitlab.vtt.fi/backbone/backbone/-/wikis/More-information/List-of-publications.

If you use Backbone in a published work, please cite the following publication, which describes the Backbone energy systems modelling framework:

Helistö, N., Kiviluoma, J., Ikäheimo, J., Rasku, T., Rinne, E., O'Dwyer, C., Li, R., & Flynn, D. (2019). Backbone—An Adaptable Energy Systems Modelling Framework. Energies, 12(17), 3388. https://doi.org/10.3390/en12173388

Main contacts

Juha Kiviluoma juha.kiviluoma@vtt.fi

Annex 3 – EMLab User Guide

EMLab

Energy Modelling Laboratory

Overview

EMLab is an agent-based model (ABM) developed with the purpose of investigating the long-term effects of climate and energy policies. In contrast to optimization models, ABMs are programmed according to decision rules given a certain context. In this ABM, the agents are power companies with limited information about the future system that make imperfect investment decisions. The investment decisions are based on the expected profitability of possible power plants in the future. Hence, it requires clearing the market multiple times in every simulation year. EMLab can simulate a CO_2 market and capacity mechanisms (CM) such as strategic reserve and capacity market.

As flexible resources will characterize the future power system, this research consortium's goal is to consider the elastic demand, storage options, and sector coupling along with some market designs. For this purpose, it is indispensable to simulate an hourly unit commitment. The original EMLab uses a segmented load duration curve and therefore doesn't enable the analysis of added flexibility, so it will be soft-linked with models that have a more detailed short-term market.

The original EMLab was developed in Java as a standalone model. The modules are controlled with a scheduler, and it is not possible to interrupt the calculation to extract values. For this reason, the algorithms are being rewritten into separate Python modules, and these modules will be referred to as EMLab-py.

Workflow

EMLab-py – AMIRIS soft linking

To analyze the role of flexibility, EMLab-py will be soft-linked with AMIRIS. This is an ABM focused on the simulation of short-term markets. In AMIRIS, the market-clearing and the renewable energies support calculations will be executed. Subsequently, the CO_2 market and the capacity mechanism will be triggered in EMLab-py. Besides the capacity market, a strategic reserve and capacity subscriptions will be considered for the capacity mechanisms.

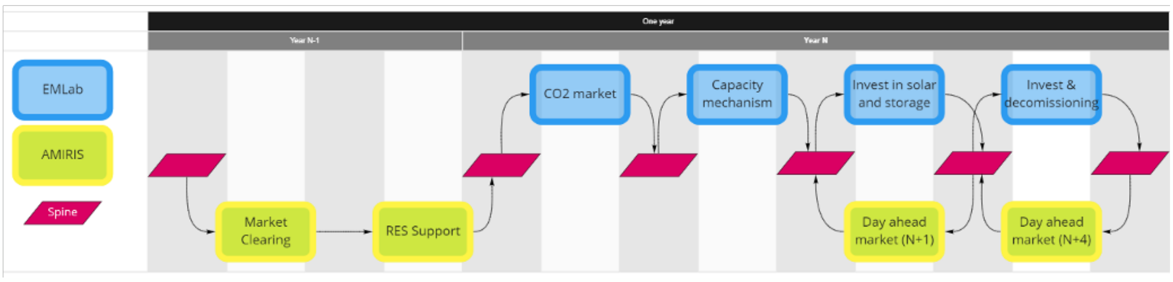

Figure 5 EMLab-py AMIRIS soft coupling workflow

Finally, the short-term investment algorithm of EMLab-py will trigger multiple iterations of AMIRIS market-clearing one year ahead of the simulation year (N+1). This short-term investment will represent the decision of installing technologies that have shorter building times and that can be profitable if these are quickly commissioned. Similarly, a long-term investment will trigger market-clearing iterations four years ahead (N+4). The optimal results will be added in the power mix considering the building time per technology.

EMLab-py – COMPETES soft coupling

See the Competes – EMLab section.

Inputs

This data is limited to the data that will be transferred between both models.

Amiris

- Hourly demand (MWh)
- Fuel Prices (Eur/MWh)
- Fixed O&M (Eur/MW)
- Demand (MWh)
- Yield profiles

EMLab py

- Yearly revenues per power plant (Eur) from spot market and from RES support, one and five years ahead
- Banked allowances (ton CO2)
- Yearly unit emissions per power plant (ton CO2)
- CAPEX (Eur/MW)
- Fuel Prices (Eur/MWh)
- Fixed O&M (Eur/MW)

Outputs

Amiris

- Yearly revenues per power plant (Eur)
- Hourly production per power plant (MW)
- Renewable support per power plant (Eur)
- Hourly nodal price (Eur/MW)

EMLab py

- CO2 prices (Eur/ton CO2)
- Capacity Market Clearing price (Eur/MWh)
- New power plants
- Capacity Mechanisms revenues per power plant (Eur)

How to run it

EMLab-py – AMIRIS

Currently, the EMLab-py – AMIRIS soft-linking is under development and the instructions to run this soft coupling will be reported in the final version of this Deliverable. The latest code can be found in this repository https://github.com/TradeRES/toolbox-amiris-emlab.

Find out more

Chappin, E. J. L. et al. (2017) 'Simulating climate and energy policy with agent-based modelling: The Energy Modelling Laboratory (EMLab)', Environmental Modelling and Software. Elsevier Ltd, 96, pp. 421–431. doi: 10.1016/j.envsoft.2017.07.009.

EMLab general purpose description: http://emlab.tudelft.nl/

Latest EMlab code: https://github.com/ejlchappin/emlab-generation2

EMLab-py – AMIRIS implementation: https://github.com/TradeRES/toolbox-amiris-emlab

Main Contacts

Ingrid Sanchez Jimenez i.j.sanchezjimenez@tudelftnl

Milos Cvetkovic m.cvetkovic@tudelft.nl

Laurens de Vries L.J.deVries@tudelft.nl

Annex 4 – COMPETES-EMLab User Guide

COMPETES-EMLab

COMPetition in Electric Transmission and Energy Simulator & Energy Modelling Laboratory

Overview - COMPETES

COMPETES is a power system optimisation and optimal dispatch model that seeks to minimize the total power system costs of the European power market whilst accounting for the technical constraints of the generation units and transmission constraints between the countries. COMPETES model can be used to perform simulations for two types of purposes:

- Least-cost capacity expansion to optimise generation and transmission capacity additions.
- Day-ahead markets, through least-cost planning and dispatch of generation and demand.

The COMPETES model covers 28 EU Member States and some non-EU countries (i.e., Norway, Switzerland, and the Balkan countries), including a representation of the crossborder transmission capacities interconnecting these European countries. The Balkan and Baltic countries are each aggregated in one node. Denmark is split into two nodes due to its participation in two non-synchronous networks. The model assumes an integrated EU market where the trade flows between countries are constrained by 'Net Transfer Capacities (NTC)'.

EMlab-py – COMPETES soft linking

COMPETES doesn't consider Capacity mechanisms and doesn't model CO2 endogenously, hence in a first attempt, we soft-linked the COMPETES optimization model with the capacity market and CO2 market modules of Emlab-py. Figure 1 shows the workflow of one simulation year. The CO2 market uses the costs and the short-term market results. The CO2 price is used to simulate the Market Clearing and generation expansion in COMPETES. Finally, the capacity market module is done in EMLab-py.

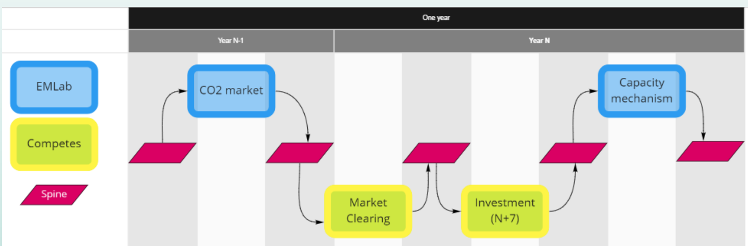

Figure 6 EMLab-py COMPETES soft coupling workflow

Inputs

This list contains the data that is exchanged between COMPETES and EMLab-py.

EMLab-py

- Average Unit Production Costs (€/MWh)
- Fuel prices (€/MW)
- Fixed O&M costs (€/MW)

Data from COMPETES

- Yearly revenues per power plants (€)
- Yearly unit emissions (ton CO_2/MWh)
- Operational plants, Unit capacity (MW)

COMPETES

The input data of COMPETES involves a wide range of generation technologies. There are 14 types of fossil-fuel power plants which can operate with CCS or as a combined heat and power (CHP) plant – nuclear, geothermal, biomass, waste, hydro, wind, and solar technologies. These can be detailed out with unit by unit generation in the Netherlands.

The main inputs for electricity supply can be summarised as:

- Operational and flexibility characteristics per technology per country:
  - Efficiencies
  - Installed power capacities
  - Availabilities (seasonal/hourly)
- Emission factors per fuel/technology.
- Fuel prices per country
- Hourly time series of VRE technologies (wind, solar, etc.).
- Overnight costs for conventional generation (€/MW).
- Transmission capital expenditures (CAPEX; €/MW).
- Fixed O&M costs (€/MW)
- Hourly demand (MWh)

Data from EMLab-py

- Capacity Market Clearing Price (€/MWh) -> Translates to a CAPEX and OPEX reduction in COMPETES input data
- CO_2 prices (€/ton CO_2) and Extrapolated CO_2 price year N + T (€/ton CO_2)

Outputs

Competes

The COMPETES model calculates the following main outputs for the EU28+ as a whole, as well as for the individual EU28+ countries and regions:

- Investments in cross-border transmission (interconnection) capacities (capacity expansion module output).
- Investments in conventional generation capacities (capacity expansion module output).
- The allocation of power generation and cross-border transmission capacity.
- Hourly and annual power generation mix – and related emissions – in each EU28+ country and region.
- The supply of flexibility options, including power generation, power trade, energy storage and VRE curtailments.
- Hourly competitive electricity prices per country/region.
- Power system costs per country/region.

EMLab-py

- CO2 Market Clearing Price (€/MWh)
- Capacity Market Clearing Price (€/MWh)

How to run it

Requirements and Installation Instructions

To run and install the soft-linking, the following software is prerequisite:

- Python
- SpineToolbox
- AIMMS (license required, academic license available)
- COMPETES (obtainable in accordance with TNO)
- Microsoft Access (or Microsoft Access Ready Driver)
- The source code is openly available and can be downloaded through the Github (J. Hommes, Github repository for this project. [Online]. Available: https://github.com/TradeRES/Spine_EMLab_COMPETES).

Once downloaded, the folder can be opened as a Spine-project in SpineToolbox.

Running Instructions

To run a case study, first the input data has to be defined. Under the 'resources/data/' folder, template files have been supplied. These should be filled and the 'template_' should be removed from the file names.

If SpineToolbox is showing red exclamation marks, this indicates that there are problems that need to be fixed before running. Most likely, this will be fixed by the following actions:

1. File references should be checked. The blocks should refer to the correct paths, e.g., the 'EMLAB Init Data'-block should refer to the 'EMLAB Init.xlsx' file.
2. The SpineDB databases have to be initialized. This is done by clicking 'New Spine DB' on the 'DB EMLAB', 'DB COMPETES' and 'Simulation Configuration Parameters' blocks.

Once SpineToolbox shows no red exclamation marks, SpineToolbox is ready to be executed. However, to be able to run the full coupling, COMPETES has to be initialized. This is done by launching AIMMS and loading COMPETES. Once loaded, the procedure 'Setup_RESTAPI' has to be run. This procedure enables the HTTP request module from the AIMMS DataExchange library. This enables SpineToolbox to communicate with AIMMS and is necessary to let SpineToolbox run COMPETES. The soft-linking is now ready to be executed.

The entire loop in SpineToolbox entails one year of execution time. The Spine toolbox workflow shown in Figure 2 represents one simulation year. This implementation was developed before the looping functionality of SpineToolbox was available and thus, as a workaround, the Clock module indexes the data in the Databases and triggers the next year iteration. Note that when running multiple years, the import blocks should not be run again. To run multiple years, the blocks from 'Init EMLAB Clock' should be run.

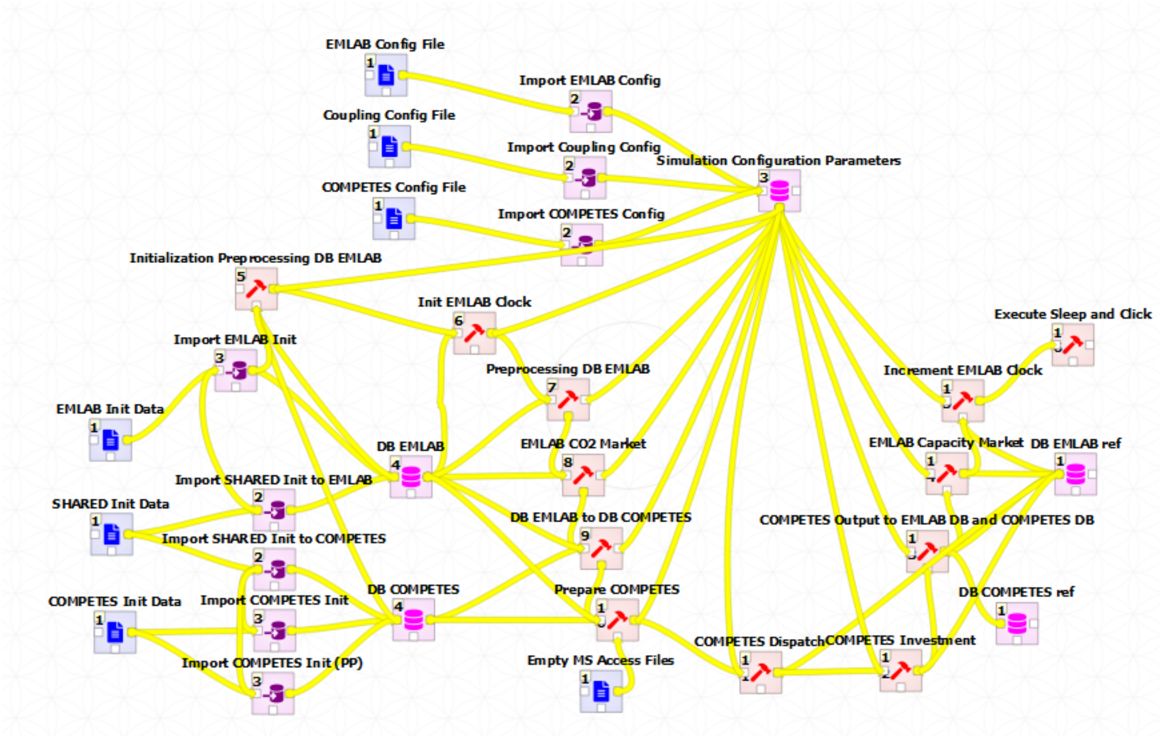

Figure 7 COMPETES implementation in Spinetoolbox

The “EMLAB Config File” and ”COMPETES Config file” contain configurations to specify the model runs. The “Coupling Config File“ specifies the order in which the data has to be imported into the databases.The Init data-blocks (in blue) point to the excel sheets to run the respective models. These resources are intended to be replaced by the TradeRES database. The connected *importer*-blocks (in purple) serve to map the input data to the Spine databases (pink). The translation script ”DB EMLab to DB Competes” collects the CO_2 and capacity market results into COMPETES and similarly the “COMPETES Output to EMLAB“ transfers the dispatch and investment results into EMLab.

After preprocessing the data to be imported to EMLab-py, the CO_2 Market is run, the CO_2 price is then fed to COMPETES which simulate the market clearing of the current year. Then the COMPETES investment block computes an optimal capacity mix seven years ahead. The new power plants are then added to the power mix considering the building times. Finally, the capacity market is simulated in EMLAB-py considering the current year dispatch results.

Find out more

The following publications describe the model in further detail and some examples of its use:

- EMLab general purpose description http://emlab.tudelft.nl/.
- Chappin, E. J. L. *et al.* (2017) 'Simulating climate and energy policy with agent-based modelling: The Energy Modelling Laboratory (EMLab)', *Environmental Modelling and Software*. Elsevier Ltd, 96, pp. 421–431. doi: 10.1016/j.envsoft.2017.07.009.
- COMPETES – EMLab-py first implementation. Master Thesis of Jim Hommes http://resolver.tudelft.nl/uuid:63691862-9a26-4df3-b6dd-57bed6c9d8a5.
- Ö. Özdemir, B.F. Hobbs, M. van Hout, P. Koutstaal (2019), Capacity vs energy subsidies for promoting renewable investment: Benefits and costs for the EU power market, Energy Policy, In Press, Available online 18 December 2019, https://doi.org/10.1016/j.enpol.2019.111166.
- Sijm J., Hout M. van, Özdemir Ö, Stralen J van., Smekens K., Well A. van der, Werner W. van, Musterd M. (2017), The supply of flexibility for the power system in the Netherlands, 2015-2050, ECN-E-17-044, November 2017. https://www.tno.nl/media/12356/e17044-flexnet-the-supply-of-flexibility-for-the-power-system-in-the-netherlands-2015-2050-phase-2.pdf

Main contacts

Main contacts for further assistance on running the model if needed:

Ricardo Hernandez Serna – ricardo.hernandezserna@tno.nl

Ingrid Sanchez Jimenez – I.J.SanchezJimenez@tudelft.nl

German Morales España – german.morales@tno.nl

Milos Cvetkovic – m.cvetkovic@tudelft.nl

Annex 5 – MASCEM User Guide

MASCEM

Multi-Agent Simulator of Competitive Electricity Markets

Overview

MASCEM [1], [2] is a simulation and modeling tool developed to study and simulate electricity market (EM) operation. It models the main market entities and their interactions. Medium/long-term gathering of data and experience is also considered to support players' decisions under their specific characteristics and goals. The main market entities are implemented as software agents, such as market and system operators, buyer and seller agents (consumers, producers, and/or prosumers), and aggregators.

The Market Operator regulates pool negotiations by validating and analyzing the players' bids depending on the type of negotiation. It also determines the market price, the accepted and refused bids, and the economical dispatch that will be sent to the System Operator. The System Operator examines the technical feasibility from the power system point of view and solves congestion problems that may arise. It is also responsible for the system's security as well as ensuring that all conditions are met within the system.

The key elements of EM are Buyer and Seller agents. A Buyer agent may be a consumer or distribution company that participates in the market to buy given amounts of power. On the other hand, a Seller agent may simulate electricity producers or other entities allowed to sell energy in the market. Aggregators represent alliances of small independent players to enable their participation in the wholesale market and compete with big players. Aggregators manage their aggregates' information and are seen from the market's point of view as buyer or seller agents.

The main types of negotiations usually present in EM included in MASCEM are day-ahead and intraday pool (symmetric or asymmetric, with or without complex conditions) markets, bilateral contracts, and intraday markets. By selecting a combination of these market models, it is also possible to perform hybrid simulations. For each scenario, the user must input the market and market type to simulate, the number of simulation days, the number of participating players, and their strategies considering each type of agent. MASCEM accommodates the simulation model of three of the main European EM: MIBEL, EPEX, and Nord Pool.

Inputs

In the scope of the TradeRES project, MASCEM allows the simulation of MIBEL day-ahead and intraday markets, EPEX and Nord Pool day-ahead markets, and the simulation of symmetric and asymmetric pools for a specific period. Inputs are passed using a JSON file for each specific type of simulation. Table 1 presents snippets with examples of the inputs required for MIBEL, EPEX, and Nord Pool.

Table 3 Snippets of input examples for simulating MIBEL, EPEX, and Nord Pool.

MIBEL (schema)	EPEX (schema)	Nord Pool (schema)

{"electricityMarket": {
"marketType": "DAY_AHEAD",
"sessionID": "1",
"numberOfPeriods": 24,
"numberOfOffersPerBid": 1,
"playersBids": [{
"playerID": "Player 1",
"bids": [{
"period": 1,
"transactionType": "SELL",
"offers": [{
"price": 25.0,
"power": 100.0
}, ...
]}, ...
],
"complexConditions": {
"indivisibility": true,
"chargeGradient": 10.0,
"minimumIncome": {
"totalIncome": 100.0,
"valuePerPowerUnit": 5.0,
"scheduledStop": true
}
}}, ...
]
}}
	{"electricityMarket": {
	"sessionID": "1",
	"numberOfPeriods": 24,
	"numberOfOffersPerBid": 1,
	"playersBids": [{
	"playerID": "Player 1",
	"bids": [{
	"period": 1,
	"transactionType": "BUY",
	"offers": [{
	"price": 25.0,
	"power": 100.0
	}, ...
	]}, ...
	],
	"blockOffers": [{
	"power": 100.0,
	"price": 25.0,
	"startPeriod": 1,
	"endPeriod": 3,
	"transactionType": "SELL"
	}, ...
	]}, ...
	]
	}}
		{"electricityMarket": {
		"sessionID": "1",
		"numberOfPeriods": 24,
		"numberOfOffersPerBid": 2,
		"playersBids": [{
		"playerID": "Player 1",
		"bids": [{
		"period": 1,
		"transactionType": "BUY",
		"offers": [{
		"price": 20.0,
		"power": 100.0
		}, ...
		]}, ...
		],
		"blockOrders": [{
		"power": 100.0,
		"price": 25.0,
		"startPeriod": 1,
		"endPeriod": 3,
		"transactionType": "BUY"
		}, ...
		],
		"flexibleOrders": [{
		"price": 25.0,
		"power": 25.0,
		"transactionType": "SELL"
		}, ...
		]}, ...
		]
		}}

The "`marketType`" property of MIBEL accepts one of the constants: "`DAY_AHEAD`" or "`INTRADAY`". The "`complexConditions`" (MIBEL), "`blockOffers`" (EPEX & Nord Pool), and "`flexibleOffers`" (Nord Pool) properties are optional for each respective EM [3], [4]. The user must uniform the "`price`" and "`power`" units. Table 2 shows input examples for the symmetrical and asymmetrical pools respectively. The only difference between them is that, in the asymmetrical pool, only sellers are able to submit prices in their bids [5]. The "`instant`" property allows to order the bids by order of submission when there is no "`price`" or when there are multiple offers with the same "`price`".

Table 4 Snippets of input examples for running asymmetrical and symmetrical pools.

Asymmetrical Pool (schema)	Symmetrical Pool (schema)

{"demandBids": [{
"offerUUID": "Player1Period1",
"energy": 0.0621,
"instant": 1493641594783
}, ...],
"supplyBids": [{
"offerUUID": "Player9Period1",
"energy": 1.987,
"price": 0.0998,
"instant": 1493642533688
}, ...]}
	{"demandBids": [{
	"offerUUID": "Player1Period1",
	"energy": 0.0621,
	"price": 0.1827,
	"instant": 1493641594783
	}, ...],
	"supplyBids": [{
	"offerUUID": "Player9Period1",
	"energy": 1.987,
	"price": 0.0998,
	"instant": 1493642533688
	}, ...]}

After market execution, there's the power flow validation of the electrical grid's technical feasibility. Table 5 presents a snippet of the power flow validation input.

Table 5 Snippet of power flow input example.

Power Flow (schema)

```
{"network": {
    "name": "Spine 1",
    "elements": {
      "buses": [{
          "id": 0,
          "busName": "bus 1",
          "voltage": 12
        }, ... ],
      "lines": [{
          "id": 6,
          "name": "line 1",
          "initialBus": 0,
          "finalBus": 1,
          "length": 21,
          "lineCapacitance": 1.5,
          "lineReactance": 1.4,
          "lineResistance": 1.4,
          "zeroSequenceLineCapacitance": 1.1,
          "zeroSequenceLineReactance": 1.2,
          "zeroSequenceLineResistance": 1.3,
          "maximumThermal": 11
        }, ... ],
      "externalGrids": [{
          "id": 10,
          "name": "external grid 1",
          "bus": 1,
          "controllable": true,
          "degree": 20,
          "inService": true,
          "maxActivePower": 100,
          "maxReactivePower": 100,
          "maxRXRatio": 100,
          "maxShortCircuitApparentPower": 100,
          "minActivePower": 100,
          "minReactivePower": 100,
          "minRXRatio": 100,
          "minShortCircuitApparentPower": 100,
          "voltage": 100,
          "zeroSequenceMaxRXRatio": 100,
          "zeroSequenceMaxX0XRatio": 100
        }]}
  },
  "powerflow": {"balancedAC": {}},
```

```
    "resultType": "json"
  }
```

The mappings between market players and the grid are made here as demonstrated bellow:

Table 6 Snippet of players - busses mappings.

Players Busses Mappings


```
{"playersBuses": [{
    "player": "Player 1",
    "bus": 0
  }, ...
}]}
```

Outputs

MASCEM's outputs, in turn, depending on what we are simulating, may assume two forms, as demonstrated bellow:

Table 7. Snippets of output examples for EM sessions and period pools.

Electricity Market Session Output


```
{"status": "ok",
  "message": {
    "market": "MIBEL",
    "session": {
      "totalDemandByPeriodOffers": 4700.0,
      "totalSupplySatisfied": 4600.0,
      "session": 1,
      "socialWelfare": 2500.0,
      "totalSupplyByPeriodOffers": 4600.0,
      "totalSupply": 4600.0,
      "playersResult": [{
        "playerPeriodsResult": [{
          "transactionType": "buying",
          "costs": 2500.0,
          "period": 1,
          "marketPrice": 25.0,
          "satisfied": 100.0,
          "demand": 100.0
        }, {
          "removalJustification": "Player 1
removed from exchange because of indivisibility
condition!",
          "period": 2
        }, ... ],
        "totalDemandSupply": 2400.0,
        "totalCostsProfits": 60000.0,
        "totalSatisfied": 2400.0,
        "totalCostsProfitsByPeriodOffers":
60000.0,
        "demandSupplyByPeriodOffers":
2400.0,
        "playerId": "Player 1"
      }, ... ],
      "consumptionSurplus": 2500.0,
      "demandSatisfiedByPeriodOffers":
4600.0,
      "averageMarketPrice": 25.0,
      "totalDemand": 4700.0,
      "totalDemandSatisfied": 4600.0,
      "generationSurplus": 0,
      "supplySatisfiedByPeriodOffers":
4600.0,
      "periods": [{
        "period": 1,
        "totalDemandByPeriodOffers": 150.0,
        "totalDemand": 150.0,
        "totalSupplySatisfied": 100.0,
        "socialWelfare": 0,
        "totalSupplyByPeriodOffers": 100.0,
        "totalSupply": 100.0,
        "totalDemandSatisfied": 100.0,
        "generationSurplus": 0,
        "supplySatisfiedByPeriodOffers":
100.0,
        "consumptionSurplus": 0,
        "demandSatisfiedByPeriodOffers":
100.0
      }, ... ]
    },
    "marketType": "SPOT"
```


Period Pool Output


```
{"status": "ok",
  "message": {
    "marketPrice": 0.1745,
    "hasTrading": true,
    "tradingResults": [{
      "offerUUID": "Player1Period1",
      "wasTraded": true,
      "transactionType": "buy",
      "energy": 0.0621,
      "tradedEnergy": 0.0621
    }, ... ],
    "totalDemand": 20.4421,
    "totalSupply": 2.7837,
    "poolResult": "TRADING",
    "totalTradedEnergy": 2.7837,
    "lastTradedSupplyOffer":
"Player5Period1",
    "lastTradedDemandOffer":
"Player7Period1"
  }
}
```

```
}}
```

It should be noticed that these results are in the perspective of the market operator agent, who then sends the individual results to each participating player. Table 8 presents a snippet of a power flow output example.

Table 8 Power flow output

Power Flow Output


```
{"results": [
    {
      "period": 1,
      "error": {
        "error": "The network has already been defined !",
        "description": {
          "name": [
            "network with this name already exists."
          ],
          "network_hash": [
            "network with this network hash already exists."
          ]
        }
      }
    },
    {
      "period": 2,
      "result": {
        "results": {
          "buses": {
            "vm_pu": { "0": 1.0, ... },
            "va_degree": { "0": 0.0, ... },
            "p_mw": { "0": 242.4489795918, ... },
            "q_mvar": { "0": 242.4496921051, ... }
          },
          "lines": {
            "p_from_mw": { "6": -242.4489795918, ... },
            "q_from_mvar": { "6": -242.4496921051, ... },
            "p_to_mw": { "6": 24244.8979591837, ... },
            "q_to_mvar": { "6": 24237.7728270453, ... },
            "pl_mw": { "6": 24002.4489795918, ... },
            "ql_mvar": { "6": 23995.3231349403, ... },
            "i_from_ka": { "6": 16.4965878141, ... },
            "i_to_ka": { "6": 15.2253597533, ... },
            "i_ka": { "6": 16.4965878141, ... },
            "vm_from_pu": { "6": 1.0, ... },
            "va_from_degree": { "6": 0.0, ... },
            "vm_to_pu": { "6": 100.0, ... },
            "va_to_degree": { "6": 0.0, ... },
            "loading_percent": { "6": 149.9689801278, ... }
          },
          "loads": {
            "p_mw": { "11": 100.0, ... },
            "q_mvar": { "11": 0.0, ... }
          },
          "externalGrids": {
            "p_mw": { "0": -142.4489795918, ... },
            "q_mvar": { "0": -242.4496921051, ... }
          }, ...
        }
      }
    }, ...
  ]}
```

How to run it

MASCEM's project within the Spine Toolbox (publicly available here for download) provides the simulation of MASCEM's EMs models and power flow models. This specific project is prepared to execute the Iberian electricity market – MIBEL – and validate the power network constraints, according to the market results. The Toolbox workflow is shown in Figure 1. This project only requires the installation of Python's JSON package.

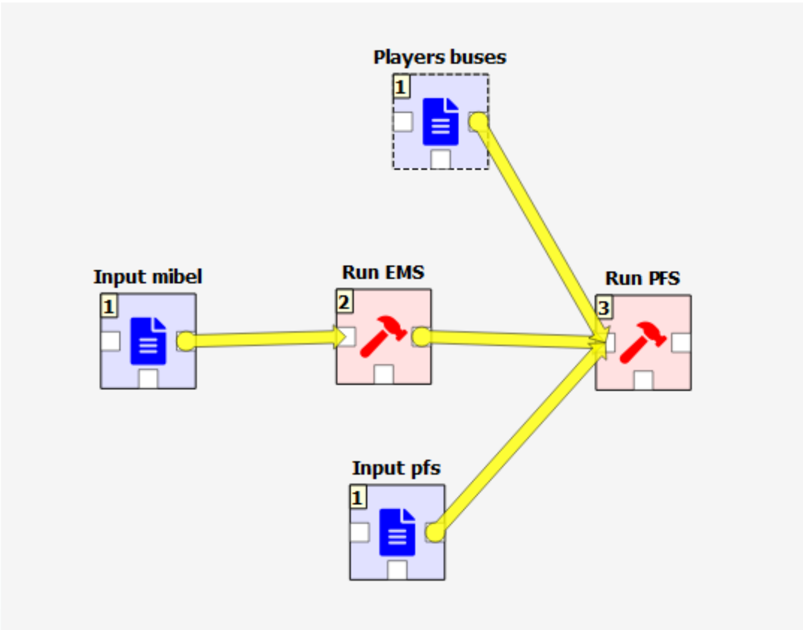

Figure 8 Workflow of MASCEM's Spine Toolbox project.

There are five components inside the MASCEM Demo Spine Toolbox project:

- **Input mibel** is the input of the **Run EMS** component and contains information regarding the players and their price/amount bids for each period. The schema that validates this input is available at ./run-ems/resources/generalSchema.json. The market input file must be placed in the run-ems folder. An example file is publicly available here.
- **Player buses** (schema) is one of the inputs of the **Run PFS** component. This file maps the players present in the MIBEL market to the buses of the network defined in the **Input pfs**.
- **Input pfs** is one of the inputs of the Run PFS component and contains information regarding the elements of the network, as well as the power flow algorithm to be used and its parameters. The schema that validates this input is available at ./run-pfs/resources/pfs_schema.json. The power flow input file

must be placed in the **run-pfs** folder. An example file is publicly available here.

- **Run EMS** executes the MIBEL day-ahead market (defined in the input of the previous component), validating it with the JSON schema created to verify the structure of the JSON input. The results of simulating the MIBEL market are then sent to the **Run PFS** tool.
- **Run PFS** executes the Power Flow service receiving the network, the power flow algorithm to be used and the loads of each bus by mapping the players' results to the buses. It also validates the **Input pfs** with the JSON schema created to verify the structure of the JSON input. The results are then saved in a JSON file.

After execution, output files are saved in dynamically generated folders named in the forms:

- **Run EMS:** run-ems/output/YYYY-MM-DDTHH.mm.ss
- **Run PFS**: run-pfs/output/YYYY-MM-DDTHH.mm.ss

Find out more

[1] Gabriel Santos, Tiago Pinto, Isabel Praça, Zita Vale, “MASCEM: Optimizing the performance of a multi-agent system“, Energy, vol. 111, pp. 513-524 (2016). DOI: 10.1016/j.energy.2016.05.127.

[2] Tiago Pinto, Zita Vale, Isabel Praça, Luis Gomes, Pedro Faria, “Multi-Agent Electricity Markets and Smart Grids Simulation with connection to real physical resources”. In “Electricity Markets with Increasing Levels of Renewable Generation: Structure, Operation, Agent-based Simulation and Emerging Designs”. F. Lopes, H. Coelho (Eds). Springer Int. Publishing (2018).

[3] Gabriel Santos, Tiago Pinto, Hugo Morais, Isabel Praça and Zita Vale, "Complex market integration in MASCEM electricity market simulator," 2011 8th International Conference on the European Energy Market (EEM), Zagreb, Croatia, 2011, pp. 256-261. DOI: 10.1109/EEM.2011.5953019.

[4] Gabriel Santos, Tiago Pinto, Zita Vale, Hugo Morais, and Isabel Praça, "Balancing market integration in MASCEM electricity market simulator," 2012 IEEE Power and Energy Society General Meeting, San Diego, CA, USA, 2012, pp. 1-8. DOI: 10.1109/PESGM.2012.6345652.

[5] Isabel Praça, Carlos Ramos, Zita Vale and Manuel Cordeiro, "MASCEM: A Multi-Agent System that Simulates Competitive Electricity Markets", IEEE Intelligent Systems, vol. 18, No.6, pp. 54-60, Special Issue on Agents and Markets, 2003.

Main contacts

Tiago Pinto tcp@isep.ipp.pt

Débora De São José drj@isep.ipp.pt

Gabriel Santos gajls@isep.ipp.pt

Zita Vale zav@isep.ipp.pt

TradeRES
New Markets Design & Models for
100% Renewable Power Systems

Annex 6 – RESTrade User Guide

RESTrade

Multi-agent Trading of Renewable Energy Sources

Overview

The RESTrade module comprises the models of the traditional power and energy reserve markets under development in the TradeRES project [1, 2]. This module supports the participation of traditional dispatchable power plants, variable renewable energy, and demand players in the system balance, i.e., automatic (aFRR) and manual (mFRR) frequency restauration reserve markets [3, 4]. Also, it uses both the marginal pricing theory (MPT) and the pay-as-bid (PAB) scheme to define prices. The aFRR capacity requirements are computed considering the balancing guide-lines of the ENTSO-E [3]. It is also possible to compute the aFRR capacity as in the Portuguese (PT) system. Furthermore, it also computes the imbalance settlement based on the PT or Spanish (SP) formulations [5, 6]. The PT formulation considers that all Balance Responsible parties (BRPs) must pay the energy used to balance the system equally. So, it computes a single penalty and dual pricing. The SP formulation computes the balance direction and only the BRPs that originate those balance needs must directly pay/receive the price of energy balance of the system.

Table 9: Iberian Imbalance Prices.

BRP position	System in up-regulation	System in down-regulation
Positive (power excess)	PT: imbalance price SP: spot price	Both: imbalance price
Negative (power deficit)	Both: imbalance price	PT: imbalance price SP: spot price

Traditional aFRR capacities are computed hourly. Very fast-responsive power plants reserve a down and up capacity to participate in this market. They can be remunerated using the MPT or the PAB. In PT their energy is remunerated considering a predefined value defined by the Regulator. However, RESTrade also provides an hourly energy aFRR market. The clearing of this market can consider both the MPT and PAB schemes. The mFRR hourly energy market is also provided, where power plants can submit bids for up and down regulation as different products. RESTrade module also enables the users to run new reserve market design models. The capacity market of aFRR is divided into up and down capacities, which allows market participants to submit independent up and down capacities. The capacity and energy reserve markets implemented in this module are also capable to deal with shorter time units starting (5, 15 or 30 minutes). This module is adapted to negotiation close-as-possible to real-time operation. These new timeframes are activated according to the data resolutions used by the users.

Inputs

The inputs to RESTrade modules are mostly from the agents' bids on reserves markets. These bids can be written to Excel files for each market. Alternatively, users can create their own input files to run the RESTrade module. Each spreadsheet of the Excel files corresponds to a period of time. All files indicated in this document are available at the root directory of the installation. All economic values are presented in monetary units (m.u.).

Table 10: Data needed to run RESTrade modules.

Agent	Variable	Unit	Description	File to edit
Supply and demand	CDsec	MW	Down capacity to offer at aFRR capacity market	Secondary.xls
	CUsec	MW	Up capacity to offer at aFRR capacity market	
	[CDsec, CUsec]	MW	Band capacity to offer at aFRR capa city market	
	PCsec	m.u ./MW	Band price to bid at aFRR capacity market	
	PDCsec	m.u /MW	Price to bid at aFRR down capacity market	
	PUCsec	m.u /MW	Price to bid at aFRR up capacity market	
	QDsec	MW	Down power to offer at aFRR energy market	SecondaryMarket.xls
	QUsec	MW	Up power to offer at aFRR energy market	
	PDEsec	m.u ./MWh	Price to bid at aFRR down energy market	
	PUEsec	€/MWh	Price to bid at aFRR up energy market	
	QDter	MW	Down power to offer at mFRR energy market	Tertiary.xls
	QUter	MW	Up power to offer at mFRR energy market	
	PDter	m.u. /MWh	Price to bid at mFRR down energy market	
	PUter	m.u ./MWh	Price to bid at mFRR up energy market	
TSO	Dmax	MW	Maximum expected demand	SecondaryNeeds.xls

(All variables are computed except Dmax and real-time energy needs)	CsecD	MW	Down capacity needs of aFRR capacity market	SecondaryOutput.xls
	CsecU	MW	Up capacity needs of aFRR capacity market	
	[CsecD, CsecU]	MW	aFRR capacity band requirements	
	EsecD, EsecU, EterD, EterU	MWh	Real-time energy requested by aFRR and mFRR reserves for up and down balances	SecondaryENeeds.xls TertiaryNeeds.xls
	EimbU, EimbD,	MWh	Total imbalanced energy	TertiaryNeeds.xls

To define the simulation behavior, a row-based binary configuration text file (*Config.txt*) is provided below. This is possible to set as follows:

Table 11: Inputs in the configuration file.

Row	Value	Description
1	0	Traditional markets
	1	Upcoming market designs
2	0	Marginal pricing
	1	Pay-as-bid
3	0	ENTSO-E aFRR procurement
	1	Portuguese aFRR procurement
4	0	Portuguese imbalance settlement
	1	Spanish imbalance settlement

This version of RESTrade module is not able to handle missing data. In case of missing data, put “0” as its respective value, and the results of that period will be “0”.

Outputs

This section presents the output of each market.

Table 12: Outputs of the RESTrade modules.

Market	Output	Unit	Description	File
Traditional aFRR capacity	[CsecDf, CsecUf]	MW	Contracted aFRR band	SecondaryOutput.xls
	PSec	m.u./MW	aFRR band strike-price	
	EsecD, EsecU	m.u./MWh	Activated aFRR up and down energy	TertiaryOutput.xls
Upcoming aFRR capacity	CsecDf	MW	Down aFRR capacity	SecondaryOutput.xls
	CsecUf	MW	Up aFRR capacity	
	PSecDf	m.u. /MW	Down aFRR capacity price	
	PSecUf	m.u. /MW	Up aFRR capacity price	
aFRR energy	CsecEf	MW	Down aFRR energy	SecondaryEOutput.xls
	CsecEf	MW	Up aFRR energy	
	PSecEf	m.u. /MW	Down aFRR energy price	
	PSecEf	m.u. /MW	Up aFRR energy price	
mFRR	EterD	MWh	Activated mFRR down energy	TertiaryOutput.xls
	EterU	MWh	Activated mFRR up energy	
	PterU	m.u./MWh	Strike-price of up mFRR reserves	
	PterD	m.u. /MWh	Strike-price of down mFRR reserves	
Imbalance settlement	Pimb	m.u./MWh	Imbalance penalty	
	PimbD	m.u./MWh	Down imbalance price	
	PimbU	m.u./MWh	Up Imbalance price	

How to run it

The RESTrade module is integrated with MASCEM in Spine Toolbox. It requires to run previously the MASCEM models as indicated in the following fluxogram:

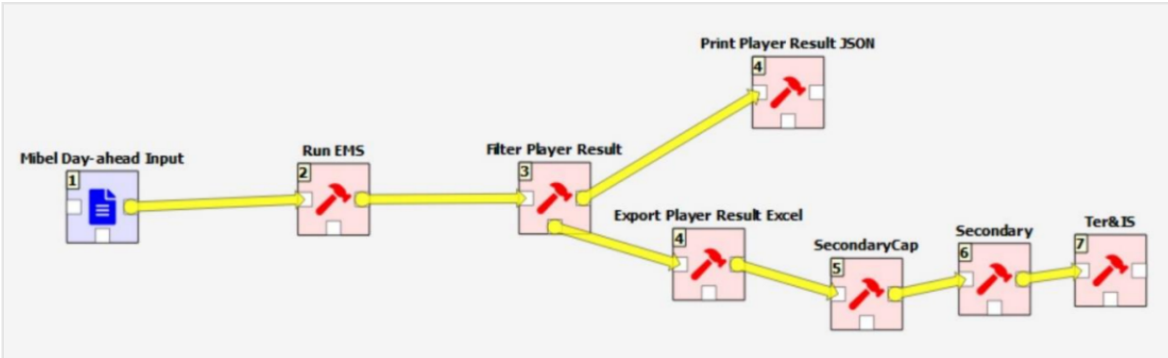

Figure 9: MASCEM and RESTrade fluxogram in Spine Toolbox.

RESTrade consists of the following blocks: “SecondaryCap”, “Secondary” and “Ter&IS”. Below is a brief explanation of these blocks and input data are provided.

SecondaryCap – This block contains both the models of the procurement of aFRR capacity and of the aFRR capacity market. To compute the aFRR procurement, it is necessary to introduce the maximum expected consumption (second column in file *SecondaryNeeds.xls*) per period under simulation (first column). The TSO submits to the market the up and down needs of the aFRR capacity and collects the agents’ bids to the market, computing the market-clearing price(s) and up and down capacities. The bids per agent and period of negotiation are inserted by the users in the *Secondary.xls* file. Each negotiation period should be introduced in the different Excel sheets. Information of each column in the *Secondary.xls* file is presented as follows:

Table 13: Description of the aFRR bids input file.

Detailed description	Supply and demand agents. Information can be inserted in ascii format	Indicate per agent the secondary capacity. This capacity has to be positive in case of traditional markets, but can be positive or negative, otherwise [MW]	Indicate per agent the price of the secondary capacity [m.u./MW]
Information needed	Indication of the agent name	CUsec/CDsec	PCsec/ PUCSec/PDCSec

Outcome: The outcomes from this market are available in the *SecondaryOutput.xls*.

Secondary – This block contains the market model for simulating the aFRR energy market. The TSO reads the secondary dispatch needs from *SecondaryENeeds.xls* file. The file needs the following information: Period, Up Needs (MW), and Down needs (MW). This agent also collects the agents' bids and computes the up and down energy prices using file *SecondaryMarket.xls.* This file has a similar format to the *Secondary.xls* file, and the same format of *Tertiary.xls,* presented as follows:

Table 14: Description of the mFRR bids input file.

Detailed description	Supply and demand agents. Information can be inserted in ascii format	Indicate per agent the secondary or tertiary power [MW]	Indicate per agent the price of the secondary or tertiary energy [m.u./MWh]
Information needed	Agent name	QDsec/ QUsec QDter/ QUter	PDEsec/ PUESec PDter/PUter

The outcomes from this market are available in the *SecondaryEOutput.xls*.

Ter&IS – This block contains both the models of the mFRR energy market and of the imbalance settlement. The TSO reads the mFRR dispatch needs from a file (*TertiaryNeeds.xls*) and collects the agents' bids (*Tertiary.xls*), and computes the up and down energy prices. This last file follows the format of the *SecondaryMarket.xls*. Tertiary energy needs and total imbalances can be edited in file *TertiaryNeeds.xls* as follows:

Table 15: Description of the TSO needs for up and down mFRR regulation.

Period number	Up needs [MWh]	Down needs [MWh]	Total Up deviations [MWh]	Total Down deviations [MWh]
1	EterU	EterD	EimbU	EimbD

In this module, the TSO also computes all energy costs with the reserve markets and the respective penalties of the imbalanced agents. The outcomes from this market, as the penalties and imbalance prices are available in the *TertiaryOutput.xls*. *SecondaryOutput.xls* and *TertiaryOutput.xls* files, contain all the capacity and energy outputs from the RESTrade modules, respectively.

TradeRES

New Markets Design & Models for 100% Renewable Power Systems

Find out more

[1] E. Rinne et al., "D4.8: Open-access tool of linked electricity market models," TradeRES project deliverable. p. 18, 2021.

[2] L. de Vries et al., "D4.5: New market designs in electricity market simulation models," TradeRES project deliverable. p. 44, 2021, [Online]. Available: https://traderes.eu/wp-content/uploads/2021/10/D4.5_TradeRES_NewMarketDesigns_H2020.pdf.

[3] ENTSO-E., "Appendix 1 - Load-Frequency Control and Performance. In ENTSO-E Operation Handbook", 2009. https://www.entsoe.eu/resources/publications/system-operations/operation-handbook/.

[4] H. Algarvio, F. Lopes, A. Couto, and A. Estanqueiro, "Participation of wind power producers in day-ahead and balancing markets: An overview and a simulation-based study," Wiley Interdiscip. Rev. Energy Environ., vol. 8, no. 5, 2019, doi: 10.1002/wene.343.

[5] Ea Energy Analyses, "Value of Forecast for a wind power plant Owner", IEA WIND TASK 36 - FORECASTING FOR WIND POWER, December 2021.https://www.ieawindforecasting.dk/-/media/sites/iea_task_36/publications/iea_task_36_value_of_forecast_100122.pdf?la=da&hash=6C75F1A29A863E39A7D75AE2D7816D0FD0B09795.

[6] P. Frade et al., "Wind balancing costs in a power system with high wind penetration–Evidence from Portugal." Energy policy 132 (2019): 702-713.

Main contacts

Hugo Algarvio hugo.algarvio@lneg.pt

Ana Estanqueiro ana.estanqueiro@lneg.pt

Fernando Lopes fernando.lopes@lneg.pt

António Couto antonio.couto@lneg.pt

Info

The TradeRES project will develop and test innovative electricity market designs that can meet society's needs of a (near) 100% renewable power system. The market design will be tested in a sophisticated simulation environment in which real-world characteristics such as actors' limited foresight into the future and risk aversion are included.

Start date
1 February 2020

End date
31 January 2024

Overall budget: € 3 988 713,75

https://traderes.eu
info@TradeRES.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 864276