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
Gabriel Santos	ISEP	gjs@isep.ipp.pt
Hugo Algarvio	LNEG	hugo.algarvio@lneg.pt
Ingrid Sanchez Jimenez	TUDelft	i.j.sanchezjimenez@tudelft.nl
Niina Helistö	VTT	niina.helisto@vtt.fi

Acknowledgements/Contributions		
Name	**Organisation**	**Email**
Filipe Sousa	ISEP	ffeso@isep.ipp.pt
Ricardo Hernandez Serna	TNO	ricardo.hernandezserna@tno.nl

Document information			
Edition	**Date**	**Dissemination Level**	**Description**
1.0	28.02.2021	Public	User guide and information to support users when executing the tools developed by the project (First edition).
2.0	**31.01.2024**	**Public**	**User guide and information to support users when executing the tools developed by the project. (Final edition)**

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 second edition of deliverable D6.2, as part of task T6.2, aims to provide a guide for users to readily access the tools that are being developed and used in the scope of the TradeRES project. As stakeholders' engagement and feedback are crucial to enable suitable testing, validation and refinement of the developed models, it is essential to create the necessary means to support this engagement.

Task T6.1 identified relevant stakeholders, as described in the deliverable D6.1, with distinct levels of expertise regarding the various topics addressed by the project. Therefore, a set of materials are provided to promote the engagement with different types of players, having in mind their characteristics and levels of expertise; such as video tutorials, webinars, meetings (e.g., conferences, seminars, workshops) and user guides.

The user guides presented in this deliverable support users when executing the tools developed and proposed by the project, being part of a report that includes a brief overview of each model, for which a user guide has been produced.

Table of Contents

List of Figures

List of Tables

Acronyms

ABM	Agent-based model
aFRR	automatic Frequency Restoration Reserve
AIMMS	Advanced Interactive Multidimensional Modeling System
AMIRIS	Agent-based Market model for the Investigation of Renewable and Integrated energy Systems
BRP	Balance Responsible Party
CAPEX	Capital Expenditures
COMPETES	COMPetition in Electric Transmission and Energy Simulator
CHP	Combined Heat and Power
CSV	Comma-separated Values
EM	Electricity Market
EMLab	Energy Modelling Laboratory
ENTSO-E	European Network of Transmission System Operators for Electricity
EPEX	European Power Exchange
ES	Spanish
EU	European Union
FAME	open Framework for distributed Agent-based Modelling of Energy systems
GAMS	General Algebraic Modeling System
HTML	HyperText Transfer Protocol
JSON	JavaScript Object Notation
MASCEM	Multi-Agent Simulator of Competitive Electricity Markets
mFRR	manual Frequency Restoration Reserve
MIBEL	Iberian Electricity Market
MPT	Marginal Pricing Theory
NPV	Net Present Value
NTC	Net Transfer Capacities
O&M	Operation and Management
OPEX	Operational Expenditures
PAB	Pay-as-bid
PT	Portuguese
RESTrade	Multi-agent Trading of Renewable Energy Sources
SIDC	Psn-European Single Intraday Coupling
SQL	Structured Query Language
TSO	Transmission System Operator
VRE	Variable Renewable Energy
vRES	Variable Renewable Energy Sources
WP	Work Package
YAML	YAML Ain’t Markup Language

1. Introduction

This document reports a collection of user guides for a set of innovative electricity market (EM) designs developed and tested in the scope of TradeRES project including the models: Agent-based Market model for the Investigation of Renewable and Integrated energy Systems (AMIRIS), Backbone, Energy Modelling Laboratory (EMLab) & AMIRIS integration, COMPetition in Electric Transmission and Energy Simulator (COMPETES) & EMLab soft coupling, Multi-Agent Simulator of Competitive Electricity Markets (MASCEM), and Multi-agent Trading of Renewable Energy Sources (RESTrade).

After engaging with different stakeholders to test the open-access tools developed in work package (WP) WP4, gathering their feedback, knowledge, and recommendations for a near 100% renewable-based market design, this second edition of deliverable D6.2 reflects the final version of the developed models and tools [1]. As the relevant stakeholders identified in task T6.2 and described in deliverable D6.1 [2] have different characteristics and levels of expertise in the topics covered by the project, task T6.2 provides a set of materials to promote their participation. These materials include video tutorials, webinars, meetings (such as conferences, seminars, and workshops) and user guides, being supplied in deliverables D6.2 and D6.3 - *Tutorial and webinar edited material*. Additionally, all these materials are publicly available in TradeRES website [3].

The user guides were designed to simplify stakeholders' involvement in shaping new market designs, while testing and validating the developed open-access market tools. The stakeholders' feedback and suggestions were taken into consideration to improve the developed models and tools, while defining recommendations for a near 100% renewable-based market design [2]. This document addresses the final and refined editions of the publicly available user guides.

The user guides are organized as follows: they start by 1) presenting an overview of the respective model/tool; after, 2) the required input data is detailed, followed by 3) the description of the relevant output data that is provided by the tool; next 4) is illustrated how to run it; and, finally, 5) the user guides conclude with references, online links, and the main contacts for the reader to learn more about the respective to the respective models/tools.

The present deliverable includes each model's user guide in Annexes A through F, as these documents are independent and may require continuous updates until the end of the project. An overview of each model is presented in the following section.

2. TradeRES Models

This report provides stakeholders and interested parties effortless access to the models and tools (developed in WP4 that are being used in the project to obtain the results in WP5) through the provision of user guides for each developed model/tool. This section is meant to provide a brief overview of each of the developed models.

2.1 AMIRIS

The open Agent-based Market model for the Investigation of Renewable and Integrated energy Systems (AMIRIS) is a model that computes electricity prices endogenously based on the simulation of strategic bidding behaviour of prototyped market actors. Simulations with AMIRIS thus enable the investigation of the influence of political framework conditions on the behaviour 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]. The AMIRIS user guide is available in Annex A. AMIRIS User guide.

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]. The Backbone user guide is available in Annex B. Backbone User guide.

2.3 EMLab-AMIRIS

Energy Modelling Laboratory (EMLab) 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]. The EMLab user guide is available in Annex C. EMLabpy-AMIRIS User guide.

2.4 COMPETES-EMLab

COMPetition in Electric Transmission and Energy Simulator (COMPETES) & EMLab 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 perform simulations for two least-cost capacity expansion to optimize 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 Annex D. COMPETES-EMLab User guide.

2.5 MASCEM

The Multi-Agent Simulator of Competitive Electricity Markets (MASCEM) is a modelling and simulation tool designed to study complex restructured electricity market operations by modelling 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 Annex E. MASCEM User guide.

2.6 RESTrade

The Multi-agent Trading of Renewable Energy Sources (RESTrade) includes different models' development in the TradeRES project, i.e., the traditional, power and energy reserve markets. The model 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 Annex F. RESTrade User guide.

3. Final Remarks

The present deliverable introduced a brief description of the different models and tools providing user guides for each one of them (available in the Annexes A through F). The objective is to provide stakeholders and interested parties straightforward access in testing, experimenting, and using the developed open-access tools and models from WP4.

The user guides are part of task T6.2 from WP6, being followed by deliverable *D6.3: Tutorial and webinar edited material* (in Month 50 of the project), which comprises a set of online video tutorials and webinars to complement and enrich the descriptions provided by the user guides. User guides may be subject to further updates according to the continuous development of the various models/tools. The user guides and support materials for the TradeRES open-access tools are also available online on the project's website.

References

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

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

[3]. TradeRES – New Markets Design & Models for 100% Renewable Power Systems. Available: https://traderes.eu. Accessed on January 15th, 2024.

[4]. AMIRIS – The open agent-based electricity market model. Available: https://dlr-ve.gitlab.io/esy/amiris/home/. Accessed on January 15th, 2024.

[5]. Backbone. Available: https://gitlab.vtt.fi/backbone/backbone. Accessed on January 15th, 2024.

[6]. EMLab – Energy Modelling Laboratory. Available: http://emlab.tudelft.nl/. Accessed on January 15th, 2024.

[7]. COMPETES – COMPetition in Electric Transmission and Energy Simulator. Available: https://repository.tudelft.nl/islandora/object/uuid:63691862-9a26-4df3-b6dd-57bed6c9d8a5?collection=education. Accessed on January 15th, 2024.

[8]. MASCEM – Multi-Agent Simulator of Competitive Electricity Markets. Available: http://www.mascem.gecad.isep.ipp.pt/overview.php. Accessed on January 15th, 2024.

Annex A. 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 behaviour of prototyped market actors. This enables AMIRIS to not only consider marginal prices but also support instruments and uncertainties. Figure A - 1 shows agents and their associated flows of information, energy, and money modelled in AMIRIS.

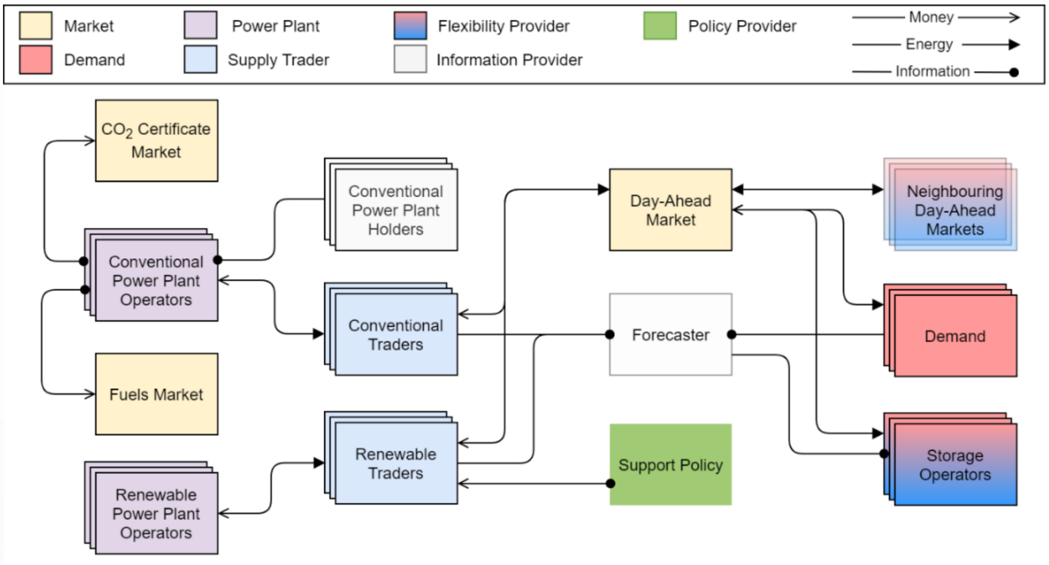


Figure A - 1. Overview of agents and their interactions in AMIRIS (open version).

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 (recursive acronym: YAML Ain't Markup Language) 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 attributes 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 the open Framework for distributed Agent-based Modelling of Energy systems (FAME) package for input file creation for FAME models and digestion of FAME outputs, i.e., 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.

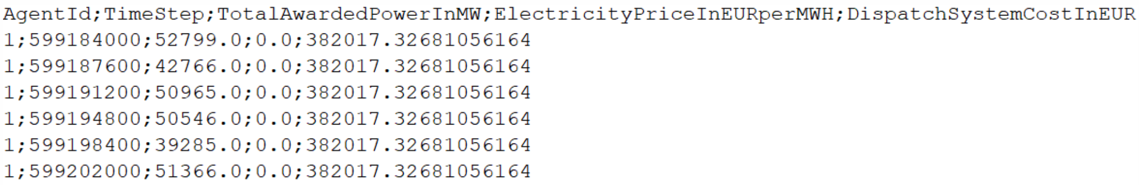
The “**Contracts**”-section configures the interactions between the agents. Each contract comprises a sender and receiver agent (identified by ID), a product type, as well as an initial execution time and execution interval (in seconds). Usually, changes to contracts are only required if agents are added 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 https://gitlab.com/dlr-ve/esy/amiris/amiris/-/wikis/GetStarted/Getting-started 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 https://gitlab.com/dlr-ve/esy/amiris/examples 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 https://gitlab.com/fame-framework/fame-io 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 A - 2 provides an example output in that format.

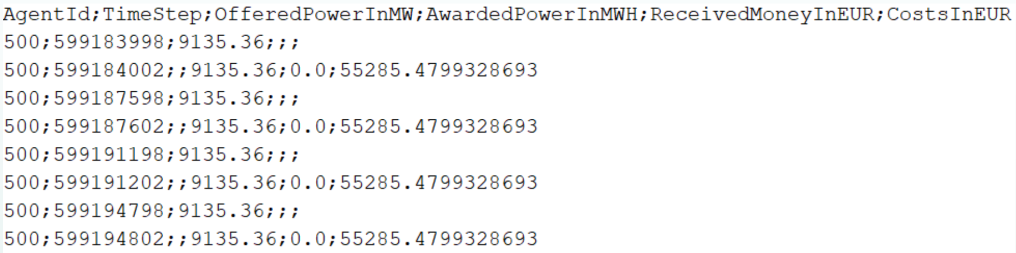


```
AgentId;TimeStep;TotalAwardedPowerInMW;ElectricityPriceInEURperMWH;DispatchSystemCostInEUR
1;599184000;52799.0;0.0;382017.32681056164
1;599187600;42766.0;0.0;382017.32681056164
1;599191200;50965.0;0.0;382017.32681056164
1;599194800;50546.0;0.0;382017.32681056164
1;599198400;39285.0;0.0;382017.32681056164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

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

Each created output file features minimally the columns “AgentId” and “TimeStep”. 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 necessarily all output columns are used in each time step. Figure A - 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 A - 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 https://gitlab.com/dlr-ve/esy/amiris/amiris/-/wikis/GetStarted/Getting-started 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 preparation, running and output conversion are joined into a single workflow. An experimental workflow to run AMIRIS within the Spine Toolbox is available 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 A - 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.

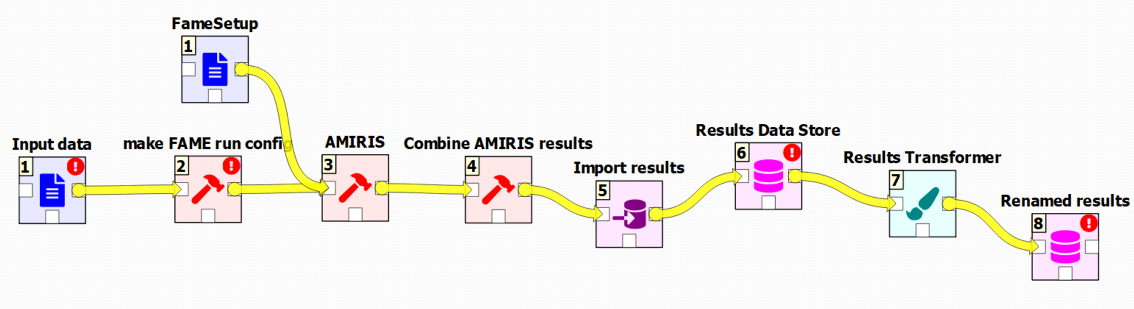


Figure A - 4. Workflow executing AMIRIS in the Spine Toolbox.

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 A - 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 other database section (8).

Table A - 1. FAME setup input parameters.

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”

TradeRES

New Markets Design & Models for
100% Renewable Power Systems

Find out more

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/

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.

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
30 November 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

Annex B. 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 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 requirements 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 (https://gitlab.vtt.fi/backbone/backbone/-/tree/master/defModels).

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 (https://gitlab.vtt.fi/backbone/backbone).

The main input data types are tabulated below.

Table B - 1. The main input data types of Backbone.

Input group	Main data types
General system data	Capacity margins, resource limits, emission taxes and limits
Transmission data	Installed capacities, technical and cost parameters including coefficients for uncontrolled flows
Generation capacity and other technology data	Installed production and consumption capacities, storage capacities, technical and cost parameters
Time-series data	Energy demand, imports, hydro inflow, wind and solar generation, reserve requirements, fuel prices, node state boundaries
Fuel data	Emission rates

Outputs

Backbone writes results in the results.gdx file. Result parameters are described in 1d_results.gms (https://gitlab.vtt.fi/backbone/backbone/-/blob/master/inc/1d_results.gms). The main output data types a tabulated below.

Table B - 2. The main output data types of Backbone.

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 each unit
Transmission between nodes	Use of transmission links
Emissions	Emissions from fuel consumption
Costs	Operational Expenditures (OPEX): fuel costs, other operation and management (O&M) costs, emission costs, penalty costs from inflexibilities
	Capital Expenditures (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, variable renewable energy (VRE) curtailment

How to run it

The Backbone tool https://gitlab.vtt.fi/backbone/backbone/-/tree/master comes with a Spine Toolbox project that is stored in the .spinetoolbox folder. The Spine Toolbox project includes a database for storing the input data in a Backbone-compatible format, as well as tools to export the input data in the inputData.gdx format to the Backbone tool.

Another demo of Backbone is shown at https://github.com/TradeRES/TradeRES-Backbone-demo. This Spine Toolbox project includes links to the common TradeRES database. In addition, the project includes tools to first convert the data in the common database to the Backbone-compatible database format and then further to the input-Data.gdx format.

TradeRES

New Markets Design & Models for 100% Renewable Power Systems

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

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
30 November 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

Annex C. EMLabpy-AMIRIS User Guide

EMLabpy-AMIRIS

Energy Modelling Laboratory & Agent-based Market model for the Investigation of Renewable and Integrated energy Systems

Overview - EMLabpy

EMLabpy is an agent-based model (ABM) developed with the purpose of investigating the long-term effects of energy policies. In EMLabpy, the agents are power companies with limited information about the future power system. The investment decisions are based on the expected profitability of candidate technologies. The expected profitability is simulated by another short-term ABM, AMIRIS. This ABM can represent flexible resources, which is essential to integrate a high share of variable renewable energy sources (vRES). Furthermore, it allows to investigate the impact of renewable energy subsidies. The Spine Toolbox is utilized to execute this co-simulation, during which models exchange data and results are saved to structured query language (SQL) databases.

EMLabpy is based on the model EMLab: http://emlab.tudelft.nl/.

Workflow

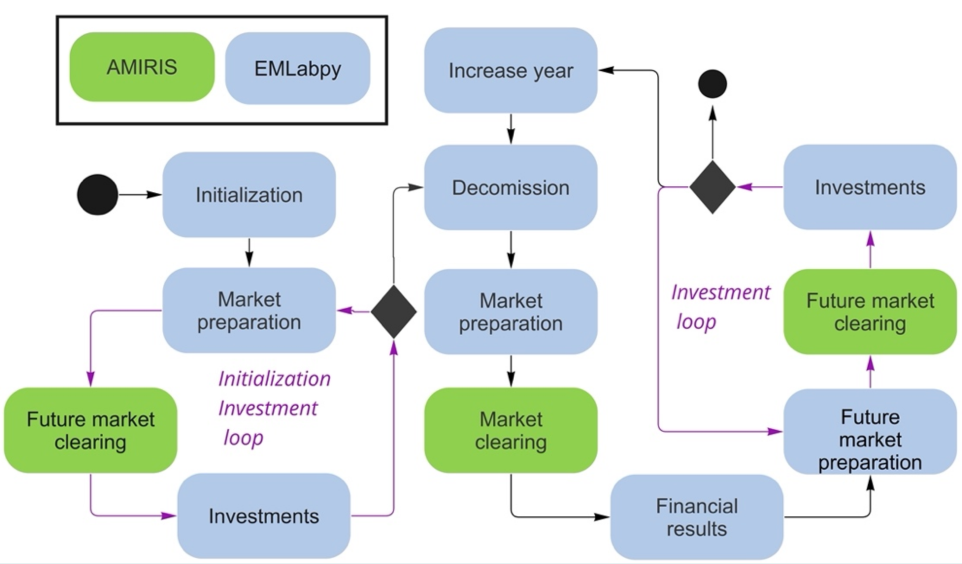


Figure C - 1. Conceptual workflow of EMLabpy-AMIRIS co-simulation.

Figure C - 1 presents a conceptual workflow of the co-simulation where the blocks in blue are executed by EMLabpy and the blocks in green are executed by AMIRIS.

Previous to the yearly loop, an initialization investment loop accounts for the power plants invested previous to the simulation start. Each simulation year (N) the age of the power plants is increased by one year, and their status is updated. Then old and unprofitable plants are decommissioned. In the market preparation step, the operational power plants are filtered and passed into AMIRIS. AMIRIS then clears the market and the results are saved in the Spine database. Along the simulation, the cash flow of the power plants is registered. In the investment loop, a future market is simulated by selecting the power plants that should be operational in a future year and evaluating potential technologies. The EMLabpy long-term investment module selects the technology with the highest Net Present Value (NPV). This investment loop is repeated until the financial and physical constraints limit any additional investment.

Inputs

The data needed to run AMIRIS is prepared by the EMLabpy modules.

The configuration data can be specified in Coupling config.xlsx. The rest of the data is found in EMLABparameters.xlsx, Power_plants.xlsx, and an Excel with the vRES profiles which can be specified in the configuration Excel (available at https://github.com/TradeRES/toolbox-amiris-emlab/tree/dev/data). Finally, in the exported_Traderes.xlsx, the technology data (investment costs, fixed and variable costs, efficiency) can be updated.

AMIRIS

- Hourly demand (MWh)
- Fuel prices (EUR/MWh)
- Variable costs (EUR/MWh)
- Power plants capacity (MW)
- Power plants efficiency (%)
- Demand (MWh)
- Renewable profiles

EMLabpy

- Yearly revenues per power plant (EUR) from AMIRIS
- CAPEX per technology per year (EUR/MW)
- Fuel prices (EUR/MWh)
- Fixed O&M (EUR/MW)
- CO_2 prices (EUR/ton CO_2)
- Renewable potentials per country
- Emission intensity per technology (tCO_2/MWh)
- Installed power plants
- Capacity mechanisms parameters
- Flexible demand share and willingness to be curtailed (EUR/MWh)

Outputs

AMIRIS

- Yearly revenues per power plant (EUR)
- Hourly production per power plant (MW)
- Hourly zonal price (EUR/MW)

EMLabpy

- New power plants per year (MW)
- Capacity Market Clearing price (EUR/MWh)
- Capacity Mechanisms revenues per power plant (EUR)

How to run it

The Spine Toolbox can be installed following the instructions available at https://github.com/spine-tools/Spine-Toolbox.

The latest code of EMLabpy-AMIRIS can be found in this repository: https://github.com/TradeRES/toolbox-amiris-emlab. Two virtual environments should be created. One to run AMIRIS and the other one to run EMLabpy. Follow the instructions in the readme.md to install all required packages and add the required files and executables as indicated. Verify and adjust the data as needed. Once all the software and data are installed and initialized, the workflow can easily be triggered clicking the play button marked in red (see Figure C - 2).

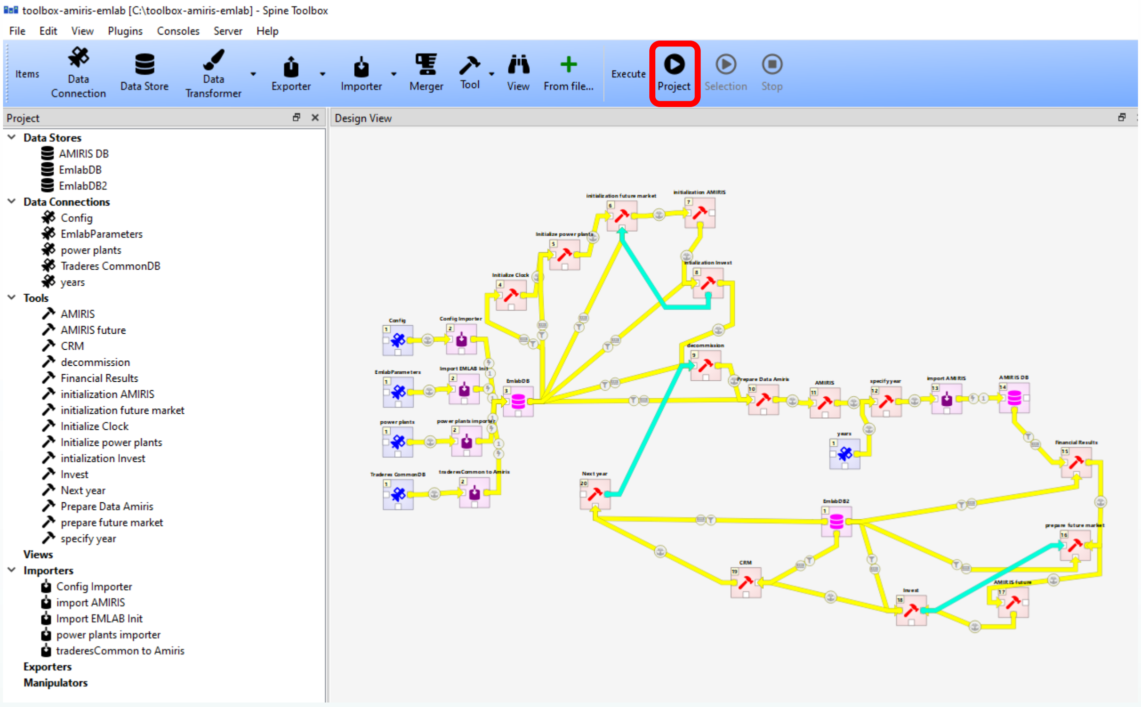


Figure C - 2. EMLabpy-AMIRIS co-simulation in the Spine Toolbox.

TradeRES

New Markets Design & Models for
100% Renewable Power Systems

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 source code: https://github.com/ejlchappin/emlab-generation2.

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

Main contacts

Ingrid Sanchez Jimenez i.j.sanchezjimenez@tudelft.nl

Milos Cvetkovic m.cvetkovic@tudelft.nl

Laurens de Vries l.j.devries@tudelft.nl

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
30 November 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

Annex D. COMPETES-EMLab User Guide

COMPETES-EMLabpy
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 cross border 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)’.

EMLabpy – COMPETES soft linking

COMPETES doesn’t consider Capacity mechanisms and doesn’t model CO_2 endogenously, hence in a first attempt, we soft-linked the COMPETES optimization model with the capacity market and CO_2 market modules of EMLabpy. Figure D - 1 shows the workflow of one simulation year. The CO_2 market uses the costs and the short-term market results. The CO_2 price is used to simulate the Market Clearing and generation expansion in COMPETES. Finally, the capacity market module is done in EMLabpy.

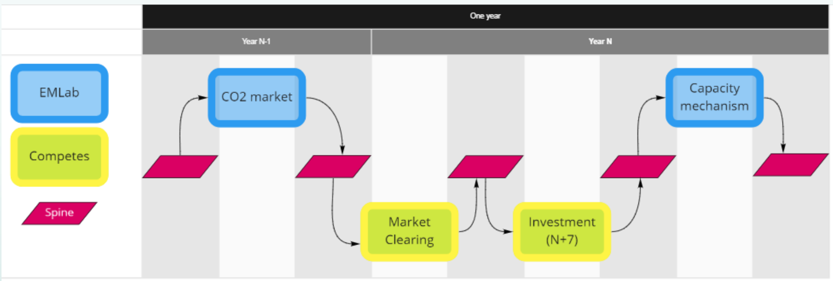


Figure D - 1. EMLabpy-COMPETES soft coupling workflow.

Inputs

This list contains the data that is exchanged between COMPETES and EMLabpy.

EMLabpy

- Average unit production costs (EUR/MWh).
- Fuel prices (EUR/MW).
- Fixed O&M costs (EUR/MW).

Data from COMPETES

- Yearly revenues per power plants (EUR).
- 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 (EUR/MW).
- Transmission capital expenditures (CAPEX; EUR/MW).
- Fixed O&M costs (EUR/MW).
- Hourly demand (MWh).

Data from EMLabpy

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

EMLabpy

- CO_2 Market Clearing Price (EUR/MWh).
- Capacity Market Clearing Price (EUR/MWh).

How to run it

Requirements and Installation Instructions

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

- Python.
- Spine Toolbox.
- Advanced Interactive Multidimensional Modeling System (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 the Spine Toolbox.

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 the Spine Toolbox 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 the Spine Toolbox shows no red exclamation marks, it 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 'Set-up_RESTAPI' has to be run. This procedure enables the HTTP request module from the AIMMS DataExchange library. This enables the Spine Toolbox to communicate with AIMMS and is necessary to let Spine Toolbox run COMPETES. The soft-linking is now ready to be executed.

The entire loop in Spine Toolbox entails one year of execution time. The Spine Toolbox workflow shown in Figure D - 2 represents one simulation year. This implementation was developed before the looping functionality of Spine Toolbox 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.

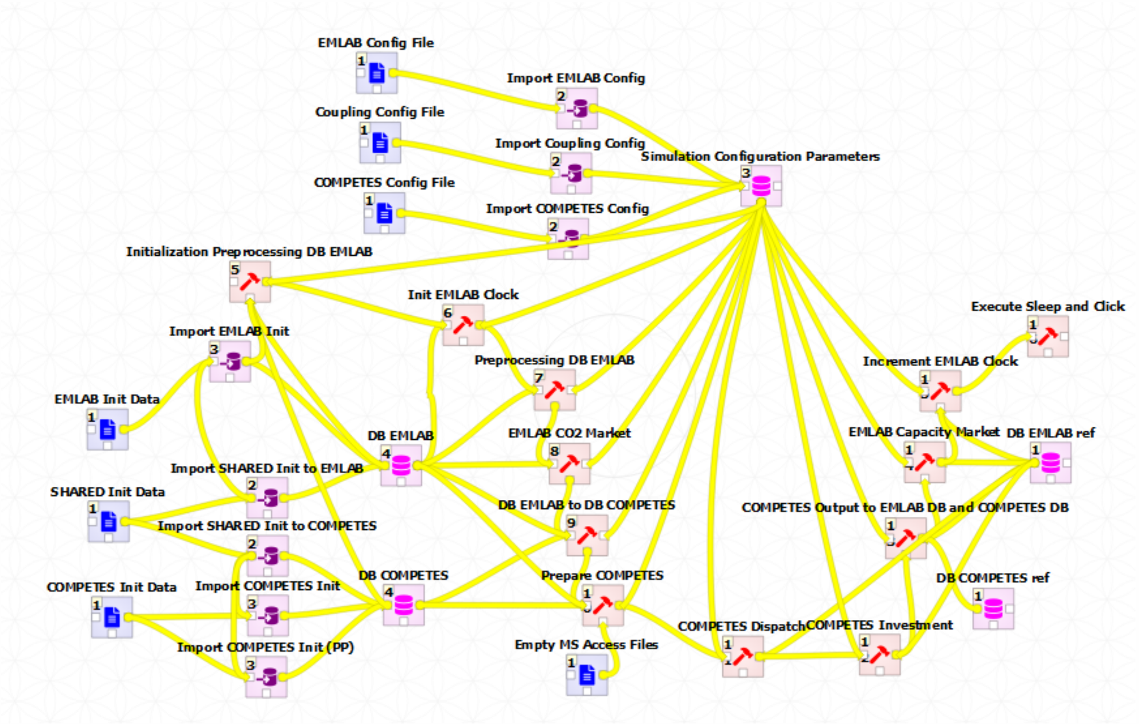


Figure D - 2. COMPETES implementation in the Spine Toolbox.

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 EMLabpy.

After preprocessing the data to be imported to EMLabpy, 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 EMLabpy considering the current year dispatch results.

TradeRES

New Markets Design & Models for
100% Renewable Power Systems

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-EMLabpy 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

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

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
30 November 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

Annex E. MASCEM User Guide

MASCEM
Multi-Agent Simulator of Competitive Electricity Markets

Overview

The Multi-Agent Simulator of Competitive Electricity Markets (MASCEM) [1], [2] is a simulation and modelling tool developed to study and simulate electricity market (EM) operations. 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 analysing 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's 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 EMs 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 in order to enable their participation in the wholesale market and compete with bigger 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 presented in the EM, that are included in MASCEM are day-ahead and intraday markets (symmetric or asymmetric, with or without complex conditions), 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: the Iberian EM (MIBEL), European Power Exchange (EPEX), and Nord Pool. Recently, in the scope of TradeRES project, the Pan-European Single Intraday Coupling (SIDC) has also been included in MASCEM.

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 SIDC. It also simulates symmetric and asymmetric pools for a specific period. Inputs are passed using a JavaScript Object Notation (JSON) file for each specific type of simulation. Table E - 1 presents snippets with examples of the inputs required for MIBEL, EPEX, Nord Pool and SIDC. For each input a link to an online *schema* is provided.

Table E - 1. Snippets of input examples for simulating MIBEL, EPEX, Nord Pool, and SIDC.

MIBEL (schema)

```
{
  “electricityMarket”: {
    “marketType”: “DAY_AHEAD”,
    “sessionID”: “1”,
    “numberOfPeriods”: 24,
    “numberOfOffersPerBid”: 1,
    “playersBids”: [
      {
        “playerID”: “Player 1”,
        “bids”: [
          {
            “period”: 1,
            “transactionType”: “SELL”,
            “offers”: [
              {
                “price”: 25,
                “power”: 100
              },
              ...
            ]
          },
          ...
        ],
        “complexConditions”: {
          “indivisibility”: true,
          “chargeGradient”: 10,
          “minimumIncome”: {
            “totalIncome”: 100,
            “valuePerPowerUnit”: 5,
            “scheduledStop”: true
          }
        }
      },
      ...
    ]
  }
}
```

EPEX (schema)

```
{
  “electricityMarket”: {
    “sessionID”: “1”,
    “numberOfPeriods”: 24,
    “numberOfOffersPerBid”: 1,
    “playersBids”: [
      {
        “playerID”: “Player 1”,
        “bids”: [
          {
            “period”: 1,
            “transactionType”: “BUY”,
            “offers”: [
              {
                “price”: 25,
                “power”: 100
              },
              ...
            ]
          },
          ...
        ],
        “blockOffers”: [
          {
            “price”: 25,
            “power”: 100
            “startPeriod”: 1,
            “endPeriod”: 3,
            “transactionType”: “SELL”
          },
          ...
        ]
      },
      ...
    ]
  }
}
```

Nord Pool (schema)

```
{
  “electricityMarket”: {
    “sessionID”: “1”,
    “numberOfPeriods”: 24,
    “numberOfOffersPerBid”: 2,
    “playersBids”: [
      {
        “playerID”: “Player 1”,
        “bids”: [
          {
            “period”: 1,
            “transactionType”: “BUY”,
            “offers”: [
              {
                “price”: 20,
                “power”: 100
              },
              ...
            ]
          },
          ...
        ],
        “blockOrders”: [
          {
            “price”: 25,
            “power”: 100
            “startPeriod”: 1,
            “endPeriod”: 3,
            “transactionType”: “BUY”
          },
          ...
        ],
        “flexibleOrders”: [
          {
            “price”: 25,
            “power”: 25,
            “transactionType”: “SELL”
          },
          ...
        ]
      },
      ...
    ]
  }
}
```

SIDC (schema)

```
{
  “sob”: {
    “day”: “2023-01-01”,
    “opening”: “14:00:00”,
    “closure”: “15:00:00”,
    “round”: 17,
    “tradingPeriod”: {
      “startTimeStamp”: “2022-12-31 16:00:00”,
      “endTimeStamp”: “2023-01-01 00:00:00”
    },
    “config”: {
      “size”: {
        “value”: 0.1,
        “unit”: “MW”
      },
      “priceTick”: {
        “value”: 0.01,
        “unit”: “EUR/MWh”
      },
      “volumeRange”: {
        “minimumValue”: 0.1,
        “maximumValue”: 999,
        “unit”: “EUR/MWh”
      },
      “priceRange”: {
        “minimumValue”: -9999,
        “maximumValue”: 9999,
        “unit”: “EUR/MWh”
      },
      “products”: {
        “id”: “hourly”,
        “duration”: “PT1H”
      }
    }
  },
  “orders”: [
    {
      “contract”: {
        “startTimeStamp”: “2023-01-01 16:00:00”,
        “endTimeStamp”: “2023-01-01 17:00:00”
      },
      “price”: {
        “value”: -0.01,
        “unit”: “EUR/MWh”
      },
      “energy”: {
        “value”: 20,
        “unit”: “MWh”
      },
      “energyPeakSize”: {
        “value”: 10,
        “unit”: “MWh”
      },
      “pricePeakDelta”: {
        “value”: 0,
        “unit”: “EUR/MWh”
      },
      “timeStamp”: “2023-01-01 14:00:00”,
      “round”: 17,
      “productID”: “hourly”,
      “zone”: “10YES-REE------0”,
      “agent”: “SOREX”,
      “unit”: “SORXC01”,
      “transactionType”: “sell”,
      “executionCondition”: “ICE”,
      “validityCondition”: “GTD”
    },
    ...
  ]
}
```

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. In the context of SIDC simulations, it is characterized by establishing limits on energy and price transactions using “energyPeakSize” and “pricePeakDelta”, defining “executionCondition” to pre-set trading conditions (“NON”, “ICE/ICE_PI”, “IOC”, “FOK”, and “AON”), and applying “validityCondition” to scrutinize bids: “GTD” or “GFS”. Table E - 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 E - 2. Snippet of input examples for running asymmetrical and symmetrical pool algorithms.

Asymmetrical Pool (schema)

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

Symmetrical Pool (schema)

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

After the market’s execution, the power flow validation of the electrical grid’s technical feasibility is simulated. Table E - 3 presents a snippet of the power flow validation input.

Table E - 3. 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 performed as demonstrated below.

Table E - 4. Snippet of Player(s)-Bus(es) mappings.

Player-Bus Mapping

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

Outputs

MASCEM’s outputs, in turn, depending on what we are simulating, may assume three forms, as demonstrated below.

Table E - 5. Snippets of output examples for each market type.

<table>
<tr><td><b>MIBEL, EPEX, & Nord Pool (Session)</b></td><td><pre>
{
  “status”: “ok”,
  “message”: {
    “market”: “MIBEL”,
    “marketType”: “SPOT”,
    “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
        },
        ...
      ]
    }
  }
}
</pre></td></tr>
<tr><td><b>SIDC (Session)</b></td><td><pre>
{
  “status”: “ok”,
  “message”: {
    “matched”: [
</pre></td></tr>
</table>

```
  {
    “buy”: {
      “uuid”: “a01249b0-5be5-4300-a74f-e23bcea617de”,
      “contract”: {
        “startTimeStamp”: “2023-01-01 16:00:00”,
        “endTimeStamp”: “2023-01-01 17:00:00”
      },
      “price”: {
        “value”: 30.92,
        “unit”: “EUR/MWh”
      },
      “energy”: {
        “value”: 5.4,
        “unit”: “MWh”
      },
      “energyPeakSize”: {
        “value”: 0,
        “unit”: “MWh”
      },
      “pricePeakDelta”: {
        “value”: 0,
        “unit”: “EUR/MWh”
      },
      “timeStamp”: “2023-01-01 14:02:38”,
      “round”: 17,
      “productID”: “hourly”,
      “zone”: “10YES-REE------0”,
      “agent”: “EGED”,
      “unit”: “EGEDRE6”,
      “transactionType”: “buy”,
      “executionCondition”: “”,
      “validityCondition”: “GTD”
    },
    “sell”: {
      “uuid”: “2a8312cf-409e-499f-bd78-3d38beb91415”,
      “contract”: {
        “startTimeStamp”: “2023-01-01 16:00:00”,
        “endTimeStamp”: “2023-01-01 17:00:00”
      },
      “price”: {
        “value”: -0.01,
        “unit”: “EUR/MWh”
      },
      “energy”: {
        “value”: 5.4,
        “unit”: “MWh”
      },
      “energyPeakSize”: {
        “value”: 10,
        “unit”: “MWh”
      },
      “pricePeakDelta”: {
        “value”: 0,
        “unit”: “EUR/MWh”
      },
      “timeStamp”: “2023-01-01 14:00:00”,
      “round”: 17,
      “productID”: “hourly”,
      “zone”: “10YES-REE------0”,
      “agent”: “SOREX”,
      “unit”: “SORXC01”,
      “transactionType”: “sell”,
      “executionCondition”: “ICE”,
      “validityCondition”: “GTD”
    },
    “matchedPrice”: {
      “value”: 0,
      “unit”: “EUR/MWh”
    },
    “matchedEnergy”: {
      “value”: 5.4,
      “unit”: “MWh”
    }
  },
  ...
],
“unmatched”: {
  “buy”: [
    {
```

	“uuid”: “00d75a74-6d5c-4f02-8347-f7837cb8c63d”,
	“contract”: {
	  “startTimeStamp”: “2023-01-01 16:00:00”,
	  “endTimeStamp”: “2023-01-01 17:00:00”
	},
	“price”: {
	  “value”: -4,
	  “unit”: “EUR/MWh”
	},
	“energy”: {
	  “value”: 4,
	  “unit”: “MWh”
	},
	“energyPeakSize”: {
	  “value”: 0,
	  “unit”: “MWh”
	},
	“pricePeakDelta”: {
	  “value”: 0,
	  “unit”: “EUR/MWh”
	},
	“timeStamp”: “2023-01-01 14:08:41”,
	“round”: 17,
	“productID”: “hourly”,
	“zone”: “10YES-REE------0”,
	“agent”: “SHEEN”,
	“unit”: “SHEVD03”,
	“transactionType”: “buy”,
	“executionCondition”: “”,
	“validityCondition”: “GTD”
	},
	...
	],
	“sell”: [
	{
	“uuid”: “2fd6ead6-3175-4e17-8ea7-48c097a3276e”,
	“contract”: {
	  “startTimeStamp”: “2023-01-01 16:00:00”,
	  “endTimeStamp”: “2023-01-01 17:00:00”
	},
	“price”: {
	  “value”: 1.09,
	  “unit”: “EUR/MWh”
	},
	“energy”: {
	  “value”: 10.5,
	  “unit”: “MWh”
	},
	“energyPeakSize”: {
	  “value”: 6,
	  “unit”: “MWh”
	},
	“pricePeakDelta”: {
	  “value”: 0,
	  “unit”: “EUR/MWh”
	},
	“timeStamp”: “2023-01-01 14:02:35”,
	“round”: 17,
	“productID”: “hourly”,
	“zone”: “10YES-REE------0”,
	“agent”: “EGED”,
	“unit”: “EGEDRE3”,
	“transactionType”: “sell”,
	“executionCondition”: “ICE”,
	“validityCondition”: “GTD”
	},
	...
	]
	}
	}
	}

Asymmetrical & Symmetrical Pools (Period)	{
	  “status”: “ok”,
	  “message”: {
	    “marketPrice”: 0.1598,
	    “hasTrading”: true,
	    “tradingResults”: [
	      {
	        “offerUUID”: “Player10Period57BidOffer”,

```
        “wasTraded”: true,
        “transactionType”: “buy”,
        “price”: 0.1827,
        “energy”: 0.0621,
        “tradedEnergy”: 0.0621
      },
      ...
    ],
    “totalDemand”: 20.4421,
    “totalSupply”: 2.7837000000000005,
    “poolResult”: “TRADING”,
    “totalTradedEnergy”: 2.6027000000000005,
    “lastTradedSupplyOffer”: “Player12Period57BidOffer”,
    “lastTradedDemandOffer”: “Player5Period57BidOffer”
  }
}
```

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 E - 6 presents a snippet of a power flow output example.

Table E - 6. Snippet of Power Flow output.

Power Flow

```
{
  “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”: {“0”: -242.4489795918, ...},
      “q_from_mvar”: {“0”: -242.4496921051, ...},
      “p_to_mw”: {“0”: 24244.8979591837, ...},
      “q_to_mvar”: {“0”: 24237.7728270453, ...},
      “pl_mw”: {“0”: 24002.4489795918, ...},
      “ql_mvar”: {“0”: 23995.3231349403, ...},
      “i_from_ka”: {“0”: 16.4965878141, ...},
      “i_to_ka”: {“0”: 15.2253597533, ...},
      “i_ka”: {“0”: 16.4965878141, ...},
      “vm_from_pu”: {“0”: 1.0, ...},
      “va_from_degree”: {“0”: 0.0, ...},
      “vm_to_pu”: {“0”: 100.0, ...},
      “va_to_degree”: {“0”: 0.0, ...},
      “loading_percent”: {“0”: 149.9689801278, ...}
    },
    “loads”: {
      “p_mw”: {“1000”: 100.0, ...},
      “q_mvar”: {“1000”: 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 E - 1. This project only requires the installation of Python's JSON package.

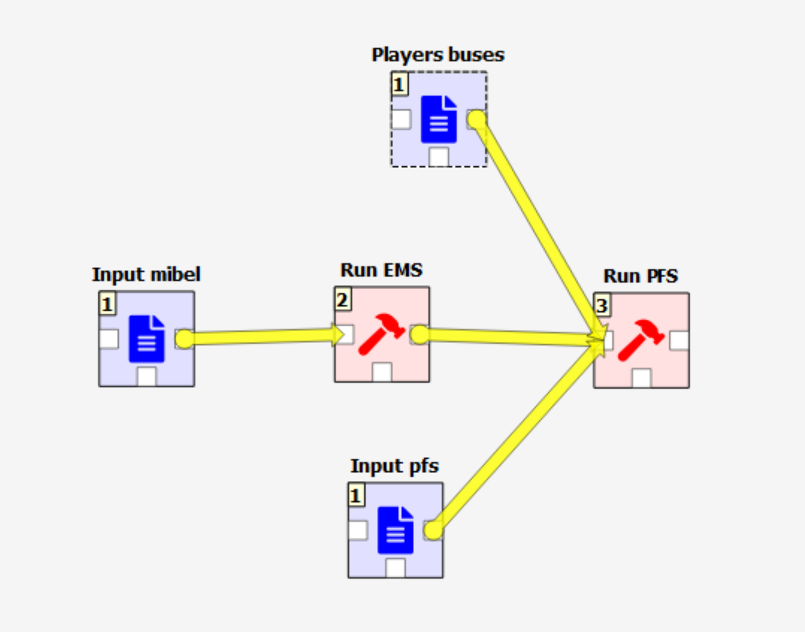


Figure E - 1. MASCEM's Spine Toolbox project workflow.

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

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

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

Gabriel Santos gjs@isep.ipp.pt

Zita Vale zav@isep.ipp.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
30 November 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

Annex F. 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 [1-3]. 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 [4, 5]. 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 guidelines of the European Network of Transmission System Operators for Electricity (ENTSO-E) [3]. It is also possible to compute the aFRR capacity as in the Portuguese (PT) system. Furthermore, it also possible to add dynamic aFRR capacity needs. Further, it also computes the imbalance settlement based on the PT or Spanish (SP) formulations [6, 7]. 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 F - 1. 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 [8]. 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.). On the scope of project TradeRES, the system was adapted to handle the separated balancing markets of PT and ES. So users may use the typical files to solve single or coupled markets or the alternative ES and PT single markets by adding "ES" or "PT" to the end of each filename.

Table F - 2. 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 capacity 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	

Agent	Variable	Unit	Description	File to edit
TSO (All variables are computed except Dmax and real-time energy needs)	Dmax	MW	Maximum expected demand	SecondaryNeeds.xls
	CsecD	MW	Dynamic aFRR down capacity needs	
	CsecU	MW	Dynamic aFRR up capacity 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 behaviour, a row-based configuration text file (Config.txt) is provided below. This is possible to set as follows:

Table F - 3. Inputs in the configuration file.

Row	Value	Description
1	0	Upcoming market designs
	(PSecEf)	Traditional markets (Regulated aFRR Energy Price)
2	0	Marginal pricing
	1	Pay-as-bid
3	0	ENTSO-E aFRR procurement
	1	Portuguese aFRR procurement
	2	Dynamic aFRR procurement
4	0	Portuguese imbalance settlement
	1	Spanish imbalance settlement
5	0	Separated markets (ES and PT)
	1	Single or coupled markets

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 [9].

Table F - 4. 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	
	PSecEDf	m.u./MWh	Down aFRR energy price	
	PSecEUf	m.u./MWh	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 (available in [10]). It requires to run previously the MASCEM models as indicated in the following flowchart [10,11]:

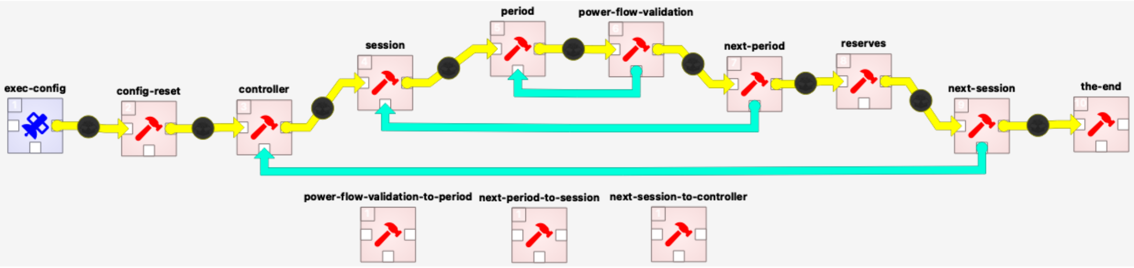


Figure F - 1. MASCEM and RESTrade flowchart in Spine Toolbox.

RESTrade consists of the “reserves” block containing the following modules: “BidsSec”, “SecondaryCap”, “BidsEnergy”, “Secondary” and “Ter&IS”. Below is a brief explanation of these blocks and input data are provided. Users may check the MASCEM user guide to understand the other blocks.

BidsSec – After the clearing of MASCEM markets, this module receives that information and the information of each aFRR participant, strategically computing the bids to the capacity aFRR market. The bids per agent and period of negotiation are computed or 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 F - 5. 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 must 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’s name	CUsec/CDsec	PCsec/ PUCSec/PDCSec

SecondaryCap – This module 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*) or the dynamic aFRR Up and Down capacities (second and fourth columns, respectivelly) per period under simulation (first column). The Transmission System Operator (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. This agent also collects the agents' bids and computes the up and down capacity prices using file *Secondary.xls*.

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

BidsEnergy – After the clearing of MASCEM and the aFRR capacity markets, this module receives the programmed dispatch of each player on each market, their prices and the information of each mFRR participant, strategically computing the bids to the energy aFRR and mFRR markets. The aFRR and mFRR energy bids per agent and period of negotiation are computed or inserted by the users in the *SecondaryMarket.xls* and *Tertiary.xls* files, respectively.

Secondary – This module 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 F - 6. 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 module contains both the algorithms 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*), computing 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 F - 7. 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..

[3] "RESTrade Tutorial Part I: Overview", https://youtu.be/bZnCma26X9c

[4] 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/.

[5] 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

[6] H. Algarvio, "Strategic Participation of Active Citizen Energy Communities in Spot Electricity Markets Using Hybrid Forecast Methodologies." Eng 4.1 (2022): 1-14.

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

[8] "RESTrade Tutorial Part II: Input Data". Available: https://youtu.be/I38UcGRSSJ0

[9] "RESTrade Tutorial Part III: Output Data". Available: https://youtu.be/xSV7AFdwC0U

[10] "MASCEM and RESTrade systems". Available: https://github.com/TradeRES/hands-on-mascem-restrade

[11] "RESTrade Tutorial Part IV: How to run it". Available: https://youtu.be/9VuI8XbrXmw

Main contacts

Ana Estanqueiro ana.estanqueiro@lneg.pt

Hugo Algarvio hugo.algarvio@lneg.pt

António Couto antonio.couto@lneg.pt

Fernando Lopes fernando.lopes@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
30 November 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