



TradeRES

New Markets Design & Models for
100% Renewable Power Systems

D6.3 – Tutorial and webinar edited material (D6.2.2)

Deliverable number: D6.3

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	TU Delft	i.j.sanchezjimenez@tudelft.nl
Nelli Putkonen	VTT	nelly.putkonen@vtt.fi

Acknowledgements/Contributions		
Name	Organisation	Email
Bruno Veiga	ISEP	btsve@isep.ipp.pt
Calvin Gonçalves	ISEP	calvi@isep.ipp.pt
Débora de São José	ISEP	drj@isep.ipp.pt
Ricardo Hernandez Serna	TNO	
Filipe Sousa	ISEP	ffeso@isep.ipp.pt

Document information			
Version	Date	Dissemination Level	Description
1.0	29.04.2022	Public	Material edited as part of the tutorials and webinars (First edition).
2.0	31.03.2024	Public	Material edited as part of the tutorials and webinars (Final edition).

Review and approval		
Prepared by	Reviewed by	Approved by
Gabriel Santos (ISEP)	Nikos Chrysanthopoulos (ICL) Goran Strbac (ICL)	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 deliverable D6.3 aims to collect and provide all the edited materials from the tutorials created during task 6.2 from work package (WP) 6. In this task, the members of the consortium provided, to the stakeholders, easy access to the tools developed and improved during WP4.

However, as shown in the results presented in D6.1 [1], stakeholders have different levels of expertise regarding the topics addressed in this project. Based on that, a set of materials are being 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 final versions of the user guides are already available and can be seen in the second edition of the deliverable D6.2 [2]. The final versions of the tutorials and webinars are addressed in this report.

The tutorials developed and available online in the project's SharePoint and Website support users when executing the tools developed within the TradeRES project, offering visual assistance, with step by step guidance, to perform all necessary operations. Besides that, a set of webinars were promoted to create a dynamic interaction between specialists and stakeholders to streamline the use of the models and the feedback process, answer questions from users, increase engagement and provide a moment for exchanging information in a model closer to face-to-face.

Based on that, this report presents the edited material that resulted from the tutorials that, together with the user guides, serve as a basis for stakeholder interaction and for the webinars presentations. Table 1 presents a summary of the publicly available produced materials.

Table 1. Overview of the publicly available produced tutorials and webinars.















	Model	No. videos	Total length	Online
Tutorials	AMIRIS	4	23:46	 
	Backbone	4	18:55	 
	EMLabpy-AMIRIS	4	11:45	 
	MASCEM	4	9:29	 
	RESTrade	4	9:15	 
Webinars	AMIRIS & Backbone	1	1:02:02	 
	MASCEM & RESTrade	1	1:00:33	 

Table of Contents

Executive Summary	3
Table of Contents	4
List of Tables	5
List of Figures	6
1. Introduction	8
2. Tutorials planning and organization	9
2.1 Approach and organization.....	9
2.2 Directory and tutorial video details	10
2.2.1. AMIRIS	10
2.2.2. Backbone	13
2.2.3. EMLabpy-AMIRIS.....	15
2.2.4. COMPETES-EMLabpy.....	18
2.2.5. MASCEM.....	20
2.2.6. REStTrade.....	23
3. Webinars planning and organization	26
3.1 Webinars details	26
3.1.1. TradeRES Webinar 1 AMIRIS & Backbone.....	26
3.1.2. TradeRES Webinar 2 MASCEM & REStTrade.....	28
4. Conclusion	30
References.....	31
Annex 1 – AMIRIS narration scripts.....	32
Annex 2 – Backbone narration scripts	36
Annex 3 – EMLabpy-AMIRIS narration scripts	40
Annex 4 – COMPETES-EMLabpy narration scripts.....	43
Annex 5 – MASCEM narration scripts	46
Annex 6 – REStTrade narration scripts	49

List of Tables

Table 1. Overview of the produced tutorials and webinars.	3
Table 2. AMIRIS collection of tutorial videos with detailed information.	11
Table 3. Backbone collection of tutorial videos with detailed information.	13
Table 4. EMLabpy-AMIRIS collection of tutorial videos with detailed information.	16
Table 5. COMPETES-EMLabpy collection of tutorial videos with detailed information.	18
Table 6. MASCEM collection of tutorial videos with detailed information.	21
Table 7. RESTrade collection of tutorial videos with detailed information.	23

List of Figures

Figure 1. Example of a tutorial (MASCEM Run tutorial).	10
Figure 2. AMIRIS overview video tutorial snapshot.	11
Figure 3. AMIRIS input video tutorial snapshot.	12
Figure 4. AMIRIS output video tutorial snapshot.	12
Figure 5. AMIRIS run video tutorial snapshot.	13
Figure 6. Backbone overview video tutorial snapshot.	14
Figure 7. Backbone input video tutorial snapshot.	14
Figure 8. Backbone output video tutorial snapshot.	15
Figure 9. Backbone run video tutorial snapshot.	15
Figure 10. EMLabpy-AMIRIS overview video tutorial snapshot.	16
Figure 11. EMLabpy-AMIRIS input video tutorial snapshot.	17
Figure 12. EMLabpy-AMIRIS output video tutorial snapshot.	17
Figure 13. EMLabpy-AMIRIS run video tutorial snapshot.	18
Figure 14. COMPETES-EMLabpy overview video tutorial snapshot.	19
Figure 15. COMPETES-EMLabpy input video tutorial snapshot.	19
Figure 16. COMPETES-EMLabpy output video tutorial snapshot.	20
Figure 17. COMPETES-EMLabpy run video tutorial snapshot.	20
Figure 18. MASCEM overview video tutorial snapshot.	21
Figure 19. MASCEM input video tutorial snapshot.	22
Figure 20. MASCEM output video tutorial snapshot.	22
Figure 21. MASCEM run video tutorial snapshot.	23
Figure 22. RESTrade overview video tutorial snapshot.	24
Figure 23. RESTrade input video tutorial snapshot.	24
Figure 24. RESTrade output video tutorial snapshot.	25
Figure 25. RESTrade run video tutorial snapshot.	25
Figure 26. Snapshot of AMIRIS presentation.	27
Figure 27. Snapshot of Backbone presentation.	28
Figure 28. Snapshot of MASCEM presentation.	29
Figure 29. Snapshot of RESTrade presentation.	29

Acronyms

AMIRIS	Agent-based Market model for the Investigation of Renewable and Integrated energy Systems
COMPETES	COMPetition in Electric Transmission and Energy Simulator
EMLab	Energy Modelling Laboratory
MASCEM	Multi-Agent Simulator of Competitive Electricity Markets
RES	Renewable Energy Sources
RETrade	Multi-agent Trading of Renewable Energy Sources
WP	Work Package

1. Introduction

This report is a collection of edited materials from AMIRIS, Backbone, EMLabpy-AMIRIS, COMPETES-EMLabpy, MASCEM and REStTrade tutorials. These tools can be described as innovative electricity market designs which are being developed/improved and tested in the TradeRES project.

One of WP6 goals is to engage different stakeholders in testing the open-access tools developed in WP4 and collect feedback, information, and recommendations for ~100% RES market designs [3]. However, the relevant stakeholders identified in task T6.1 and described in the deliverable D6.1 [1] have different levels of expertise in the topics covered by the project. On this basis, a set of materials is provided as part of task T6.2 to facilitate their participation in task T6.3, the test of the open-access tools by different typologies of stakeholders, considering their characteristics and level of expertise. The materials are already available in the final version of the deliverable D6.2 [2], this mainly consists of user guides, and will be complemented in deliverable D6.3, which corresponds to the tutorial and webinar materials. All the materials are publicly available in the project's Website and YouTube channel.

All the tutorials are organized as follows: the first video introduces an overview of the respective model, followed by a set of three, describing the inputs, outputs, and how to run the model, with detailed instructions to carry out the necessary operations.

The webinars were organized in pairs and presented in 2 consecutive weeks. They were structured to include an introduction of the model, main operations, discussion of open points and provision of needed clarifications, and a moment of direct interaction with the participants for questions and answers and direct feedback.

Both tutorials and webinars were designed to facilitate stakeholders' involvement in shaping 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 [3].

This report presents an overview of tutorials created, with a small description of each type and its intention, being organized as follows. First, a small introduction is made, with focus on the scope of the deliverable, its structure, and the relationship with other deliverables. This section is followed by the tutorials planning and organization, where the tutorials' structure is presented, with the activities that lead to its execution and the links to access each one of the videos. The third section addresses the webinars planning and organization, where the webinars' structure is presented and links to access each video. Finally, the conclusions of this deliverable are presented, followed by the references and the narration scripts of the videos produced for consultation in the attachments.

2. Tutorials planning and organization

This report provides stakeholders easy access to the tools developed/improved in WP4 and the findings of WP5 through the provision of a series of video tutorials that are described in this section.

2.1 Approach and organization

For facilitating accessibility and quicker reference the tutorials have been split in four videos. One tutorial per tool consists of four components, i.e., an overall description and the three parts that refer to the distinct elements of the simulation (inputs, outputs, running). Based on that, every tool contemplates four videos divided into the following categories:

- Descriptive videos:
 - Overview – In these videos, a brief description of the tools is presented with the aim of not only presenting them to stakeholders, but also developing familiarity and supporting the greater understanding of each tool. With this, the intention is to make the stakeholders feel closer to the tool, more capable of using it, more open to the other tutorials and better prepared to understand what will be presented in the other tutorials whose content will be more technical/practical.
- Teaching videos:
 - Input – In this topic, stakeholders can see and understand how users can find and manipulate the input data required to run each model. These videos include the data to be used and in which format, and, also, any points that the user should be aware of.
 - Output – In this topic, stakeholders can see and understand how users can find the output data generated after the end of operations, in addition to the type of results they can obtain and learn how to interpret the results obtained.
 - Run – In this set of videos, stakeholders can see, step by step, how to run each model, with practical and didactic examples being displayed on screen.

During the process of planning and preparing the videos, ISEP prepared and conducted an internal seminar for presenting the recording/editing tools and for helping other partners in the process of making the videos. Thus, ensuring the alignment of the work teams and a concise design between all the videos produced. Two ISEP specialists led the meeting, presented demonstrative examples, and answered the questions presented by other partners, thus guaranteeing the fulfilment of the goal, for both task and the deliverable.

Besides that, guidance was also given regarding the requirements for creating the videos, e.g., subjects to be addressed, video duration, audience targeted, subtitles, opening image, ending image, voice over, beside other technical aspects of the videos.

All videos have the same structure, they were made by recording the computer screen itself while a specialist performed the necessary activities, step by step, in a simple and didactic way, and narrated, in English, the activities necessary to successfully complete the

process. English subtitles were introduced in all videos to make them more accessible to different audiences.

Figure 1 presents an example of a video tutorial that was created and is publicly available on the project's website.

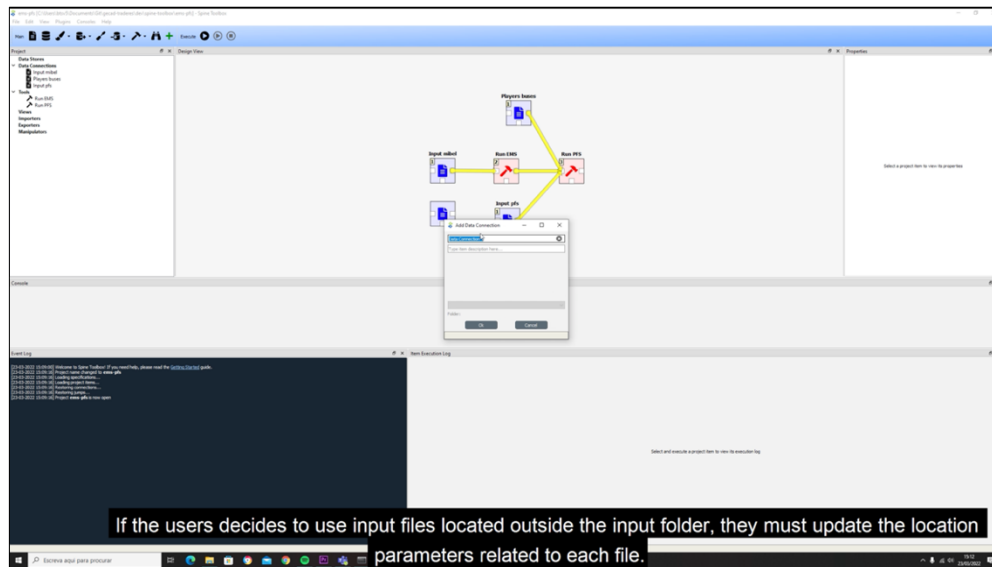


Figure 1. Example of a tutorial (MASCEM Run tutorial).

Finally, the tutorials were made with two main points in mind, in addition to the goals already mentioned: (1) the videos are independent, to facilitate access to the information one really wants, that is, a stakeholder does not need to see all the videos or access a long single video if their objective is only to know how to access the outputs obtained after using AMIRIS, for example; (2) the videos were made to be as short as possible (keeping a high quality and functionality), in order to encourage access and enhance their use.

2.2 Directory and tutorial video details

This section presents a summary of all videos that were created and made available online, with a brief description of each tool, the videos general characteristics and a link to access it.

2.2.1. AMIRIS

The open Agent-based Market model for the Investigation of Renewable and Integrated energy Systems (AMIRIS) [4] 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. AMIRIS tutorial's narration scripts are available in Annex 1 – AMIRIS narration scripts. Table 2 presents the AMIRIS tutorial videos properties.

Table 2. AMIRIS collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	2:46	1280×720	52,8MB	MPEG-4	AAC, H.264	link
Input	10:57	1920×1080	327MB	MPEG-4	AAC, H.264	link
Output	5:09	1920x1080	153MB	MPEG-4	AAC, H.264	link
Run	4:54	1920×1080	146MB	MPEG-4	AAC, H.264	link

Figure 2 to Figure 5 present a glimpse of the tutorial videos made available for the AMIRIS tool, each one of them can be seen in the links available in Table 2.

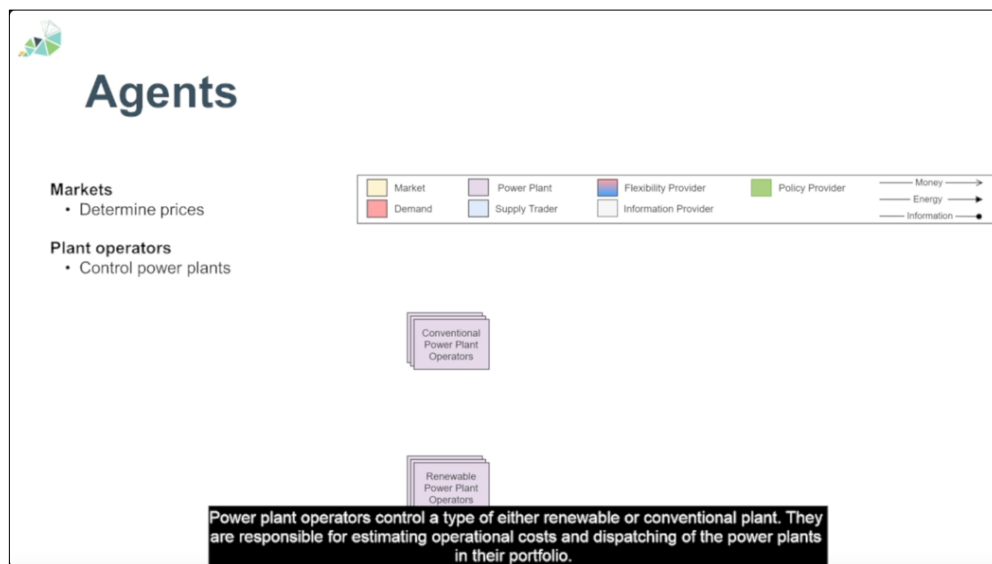


Figure 2. AMIRIS overview video tutorial snapshot.

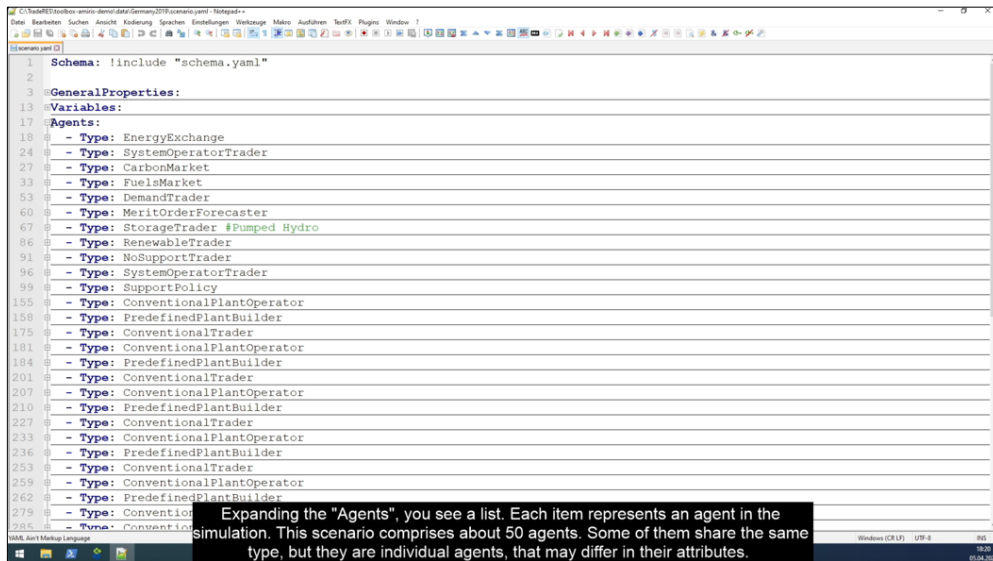


Figure 3. AMIRIS input video tutorial snapshot.

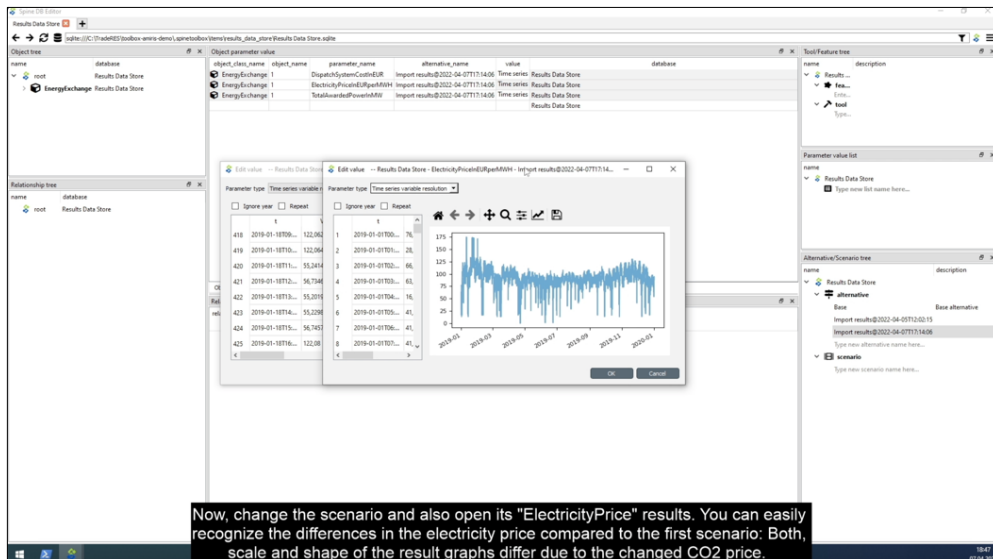


Figure 4. AMIRIS output video tutorial snapshot.

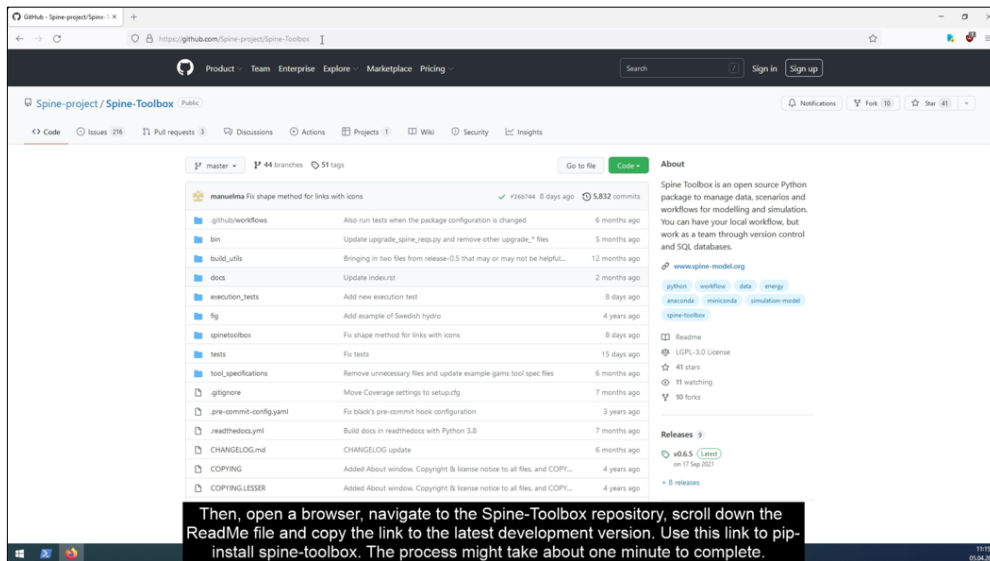


Figure 5. AMIRIS run video tutorial snapshot.

2.2.2. Backbone

Backbone [5] 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. Backbone tutorial's narration scripts are available in Annex 2 – Backbone narration scripts. Table 3 presents the Backbone tutorial videos properties.

Table 3. Backbone collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	2:22	1280×720	59MB	MPEG-4	AAC, H.264	link
Input	6:37	1280×720	339.8MB	MPEG-4	AAC, H.264	link
Output	4:28	1280×720	182.5MB	MPEG-4	AAC, H.264	link
Run	5:28	1280×720	174.1MB	MPEG-4	AAC, H.264	link

Figure 6 to Figure 9 present a catch sight of the tutorial videos made available for the Back-bone tool, each one of them can be seen in the links available in Table 3.

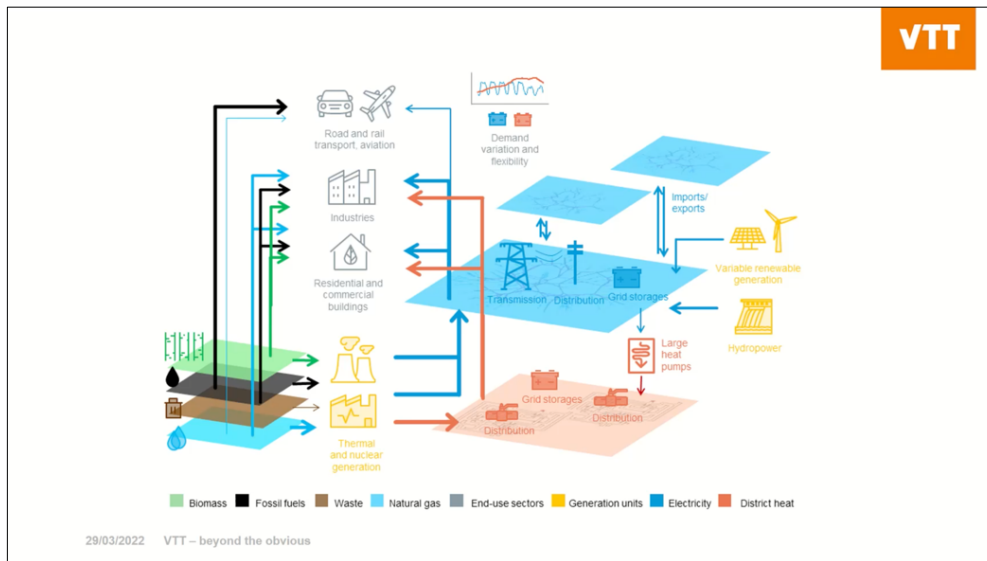


Figure 6. Backbone overview video tutorial snapshot.

The screenshot shows an Excel spreadsheet with a table of input data. The table has columns for 'unit', 'node', 'flow', 'input', 'output', 'unit size', 'invCosts', 'availabilityCapacity', 'variableCapacity', 'vmaxCosts', 'fmaxCosts', 'fmax', 'Eq', 'factor', 'check', 'margin', 'margin(B-1)', 'factor', 'check', 'margin', 'Q', 'pset1', 'pset2', 'number of stations', 'group 1', and 'group 2'. The rows list various units such as 'w_city_bioheat', 'w_city_oilheat', 'w_city_gasCHPflex', 'w_city_gasCHPfixed', 'w_city_gasCHPpgr', 'w_city_heatpumps', 'w_city_heatStorageCharge', 'w_city_heatStorageDischarge', 'w_country_wind', 'w_country_pv', 'w_country_hydro', and 'w_country_nuclear'.

Figure 7. Backbone input video tutorial snapshot.

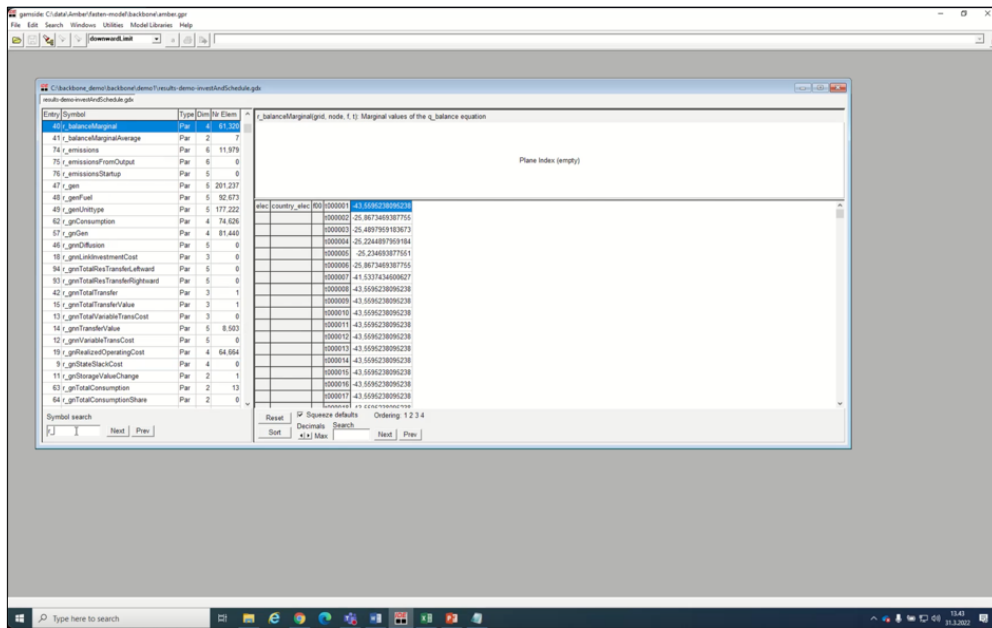


Figure 8. Backbone output video tutorial snapshot.

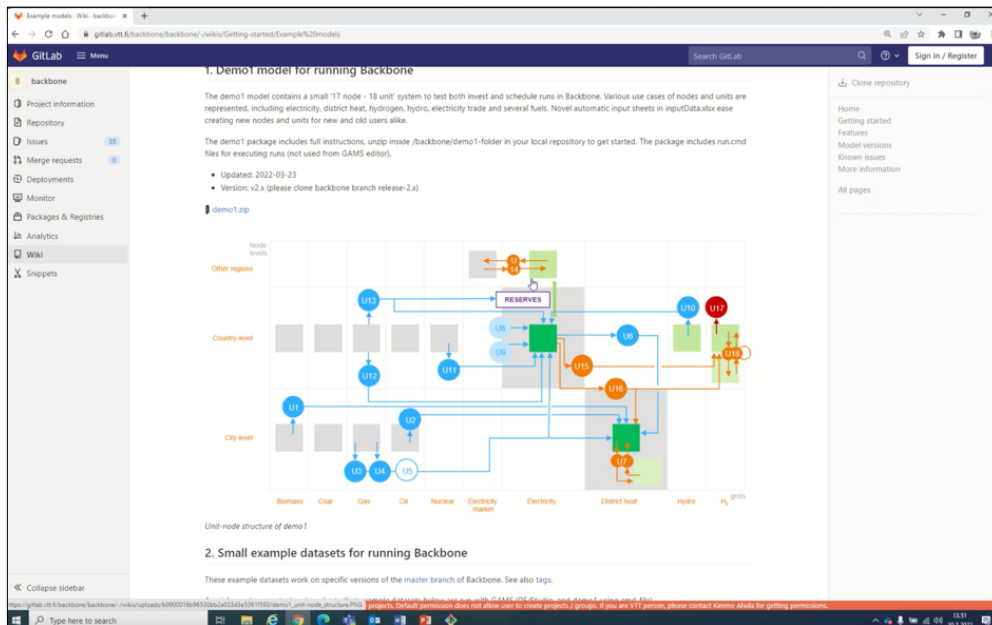


Figure 9. Backbone run video tutorial snapshot.

2.2.3. EMLaby-AMIRIS

Energy Modelling Laboratory (EMLab) [6] 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 CO2 market and capacity mechanisms. EMLaby is a Python version of the EMLab model developed in the scope of the TradeRES project for integration purposes. EMLaby-AMIRIS tutorial's narration scripts are available in Annex 3 – EMLaby-AMIRIS narration scripts. Table 4 presents the EMLaby-AMIRIS tutorial videos properties.

Table 4. EMLabpy-AMIRIS collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	4:42	1920x1080	17.8MB	MPEG-4	AAC, H.264	link
Input	1:57	1920x1080	13.1MB	MPEG-4	AAC, H.264	link
Output	1:59	1920x1080	12.2MB	MPEG-4	AAC, H.264	link
Run	3:07	1920x1080	17.6MB	MPEG-4	AAC, H.264	link

Figure 10 to Figure 13 present snippets of the tutorial videos made available for the EMLabpy-AMIRIS tool, each one of them can be seen in the links available in Table 4.

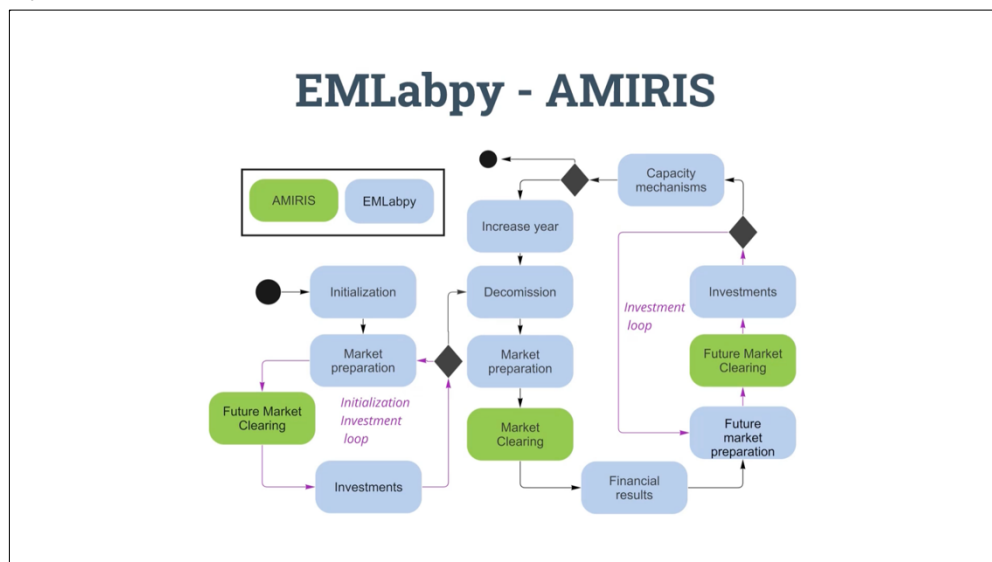


Figure 10. EMLabpy-AMIRIS overview video tutorial snapshot.

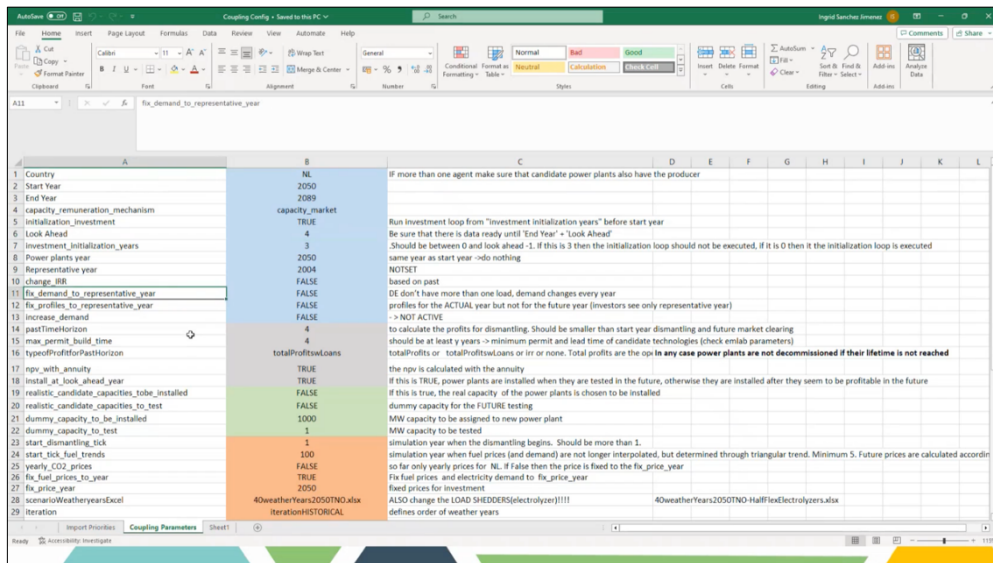


Figure 11. EMLabpy-AMIRIS input video tutorial snapshot.

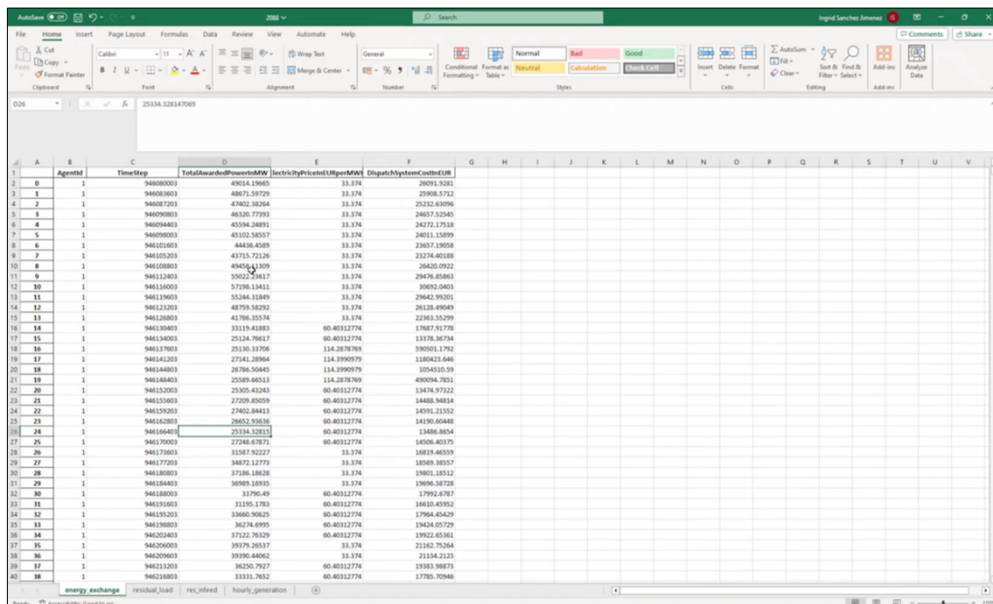


Figure 12. EMLabpy-AMIRIS output video tutorial snapshot.

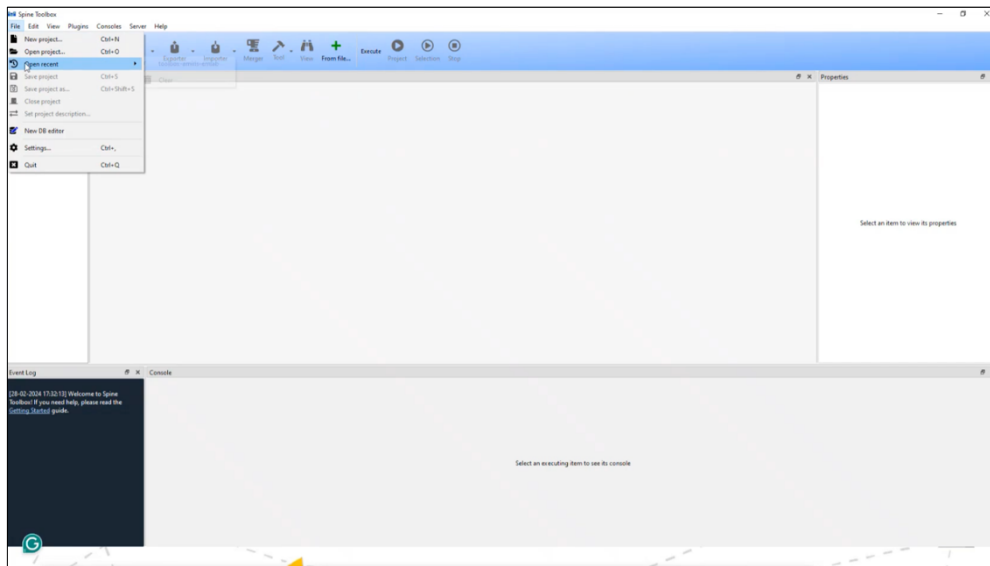


Figure 13. EMLabpy-AMIRIS run video tutorial snapshot.

2.2.4. COMPETES-EMLabpy

COMPETition in Electric Transmission and Energy Simulator [7] 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 optimize generation and transmission capacity additions and day-ahead markets, through least-cost planning and dispatch of generation and demand. Since COMPETES is not open access, its materials were not made publicly available. COMPETES-EMLabpy tutorial's narration scripts are available in Annex 4 – COMPETES-EMLabpy narration scripts. Table 5 presents the COMPETES-EMLabpy tutorial videos properties.

Table 5. COMPETES-EMLabpy collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	1:56	1280×720	10.2MB	MPEG-4	AAC, H.264	link
Input	3:37	1280×720	32.6MB	MPEG-4	AAC, H.264	link
Output	2:09	1280×720	19.4MB	MPEG-4	AAC, H.264	link
Run	3:46	1280×720	32.1MB	MPEG-4	AAC, H.264	link

Figure 14 to Figure 17 present a look at the tutorial videos made available for the COMPETES-EMLabpy tool, each one of them can be seen in the links available in Table 5.

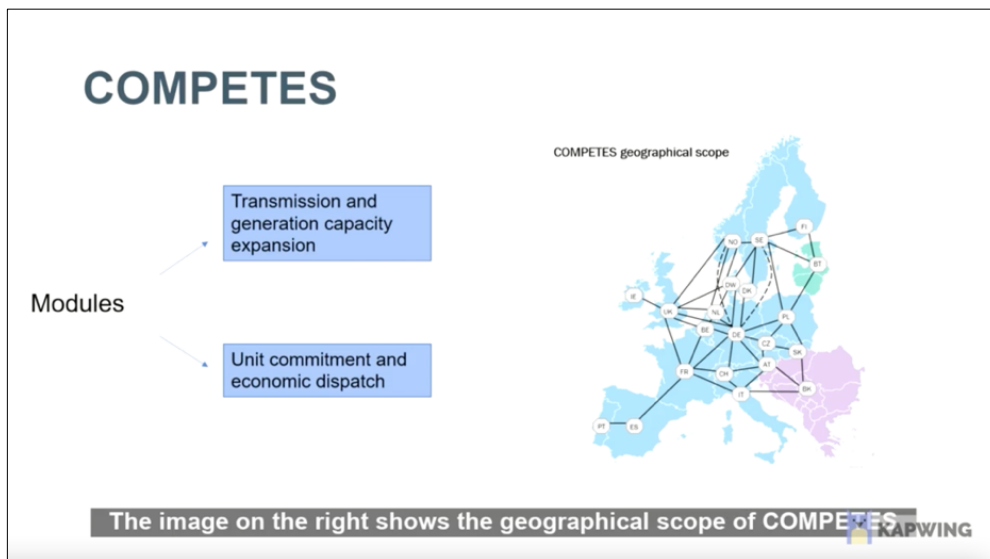


Figure 14. COMPETES-EMLaby overview video tutorial snapshot.

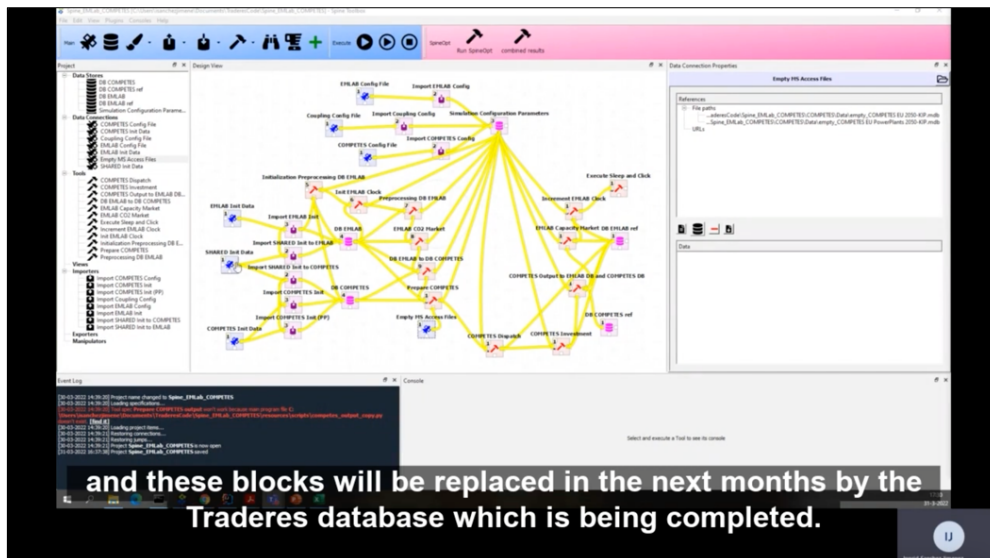
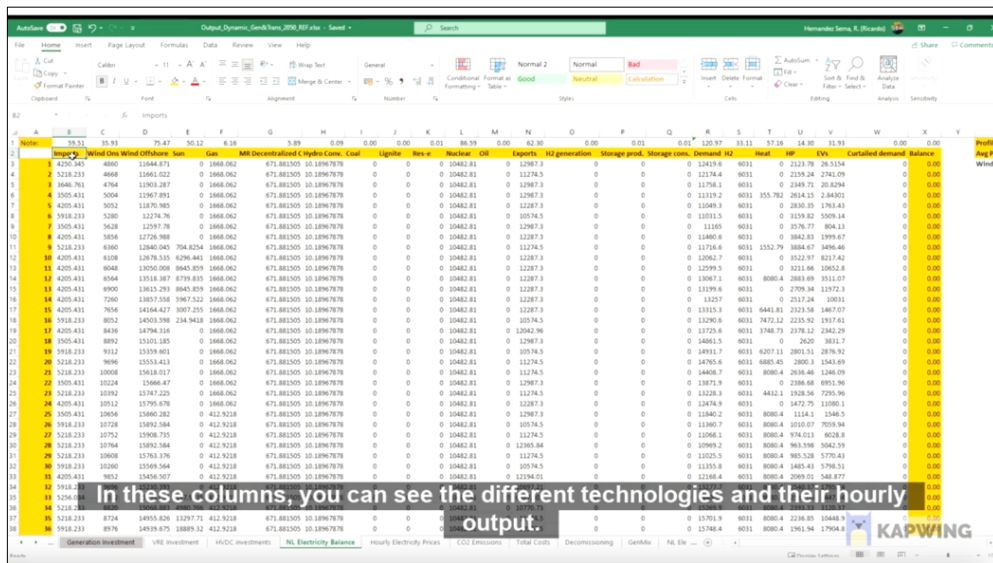


Figure 15. COMPETES-EMLaby input video tutorial snapshot.



In these columns, you can see the different technologies and their hourly output

Figure 16. COMPETES-EMLaby output video tutorial snapshot.

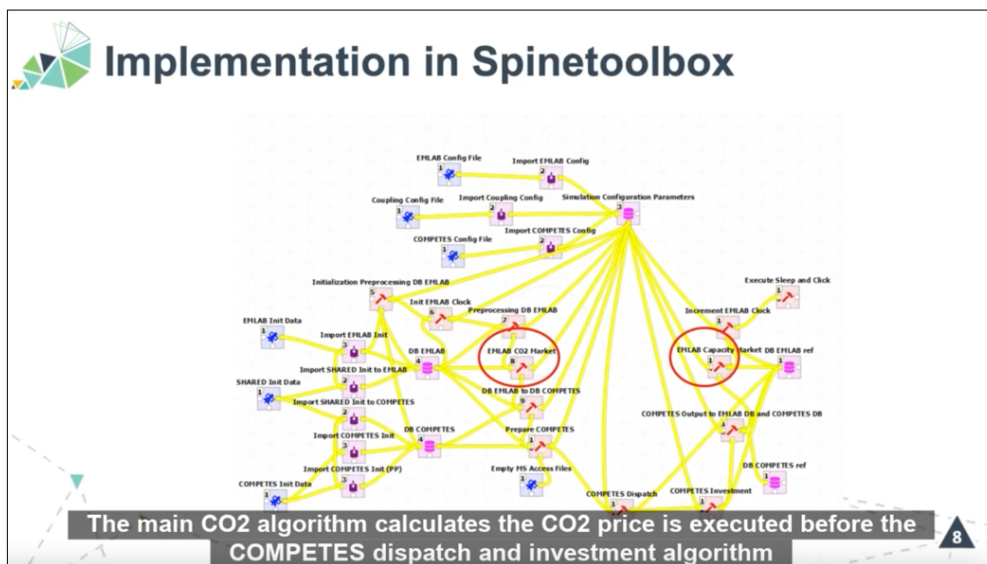


Figure 17. COMPETES-EMLaby run video tutorial snapshot.

2.2.5. MASCEM

The Multi-Agent Simulator for Competitive Electricity Markets (MASCEM) [8] is a modeling 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. MASCEM tutorial's narration scripts are available in Annex 5 – MASCEM narration scripts. Table 6 presents the MASCEM tutorial videos properties.

Table 6. MASCEM collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	2:08	2560x1440	45.6MB	MPEG-4	AAC, H.264	link
Input	3:59	2560x1440	99.9MB	MPEG-4	AAC, H.264	link
Output	1:46	2560x1440	50.1MB	MPEG-4	AAC, H.264	link
Run	1:36	2560x1440	39.9MB	MPEG-4	AAC, H.264	link

Figure 18 to Figure 21 present a preview of the tutorial videos made available for the MASCEM tool, each one of them can be seen in the links available in Table 6.

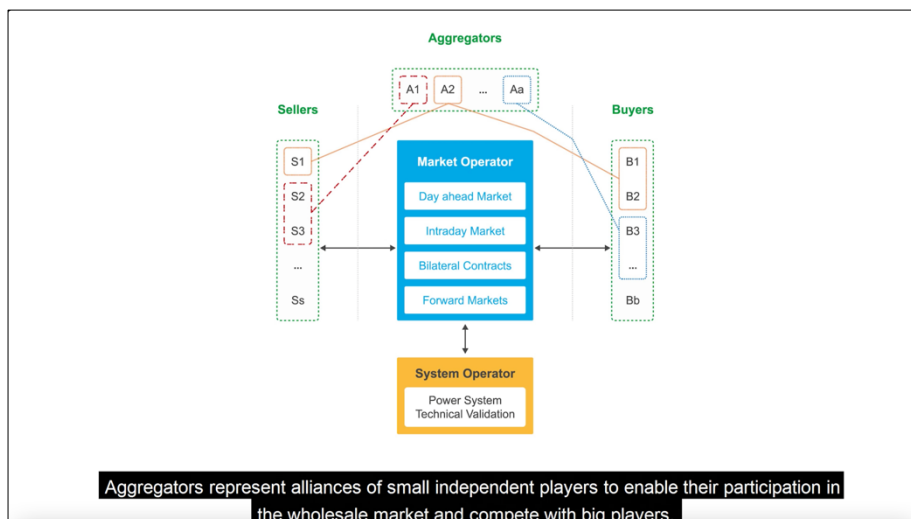
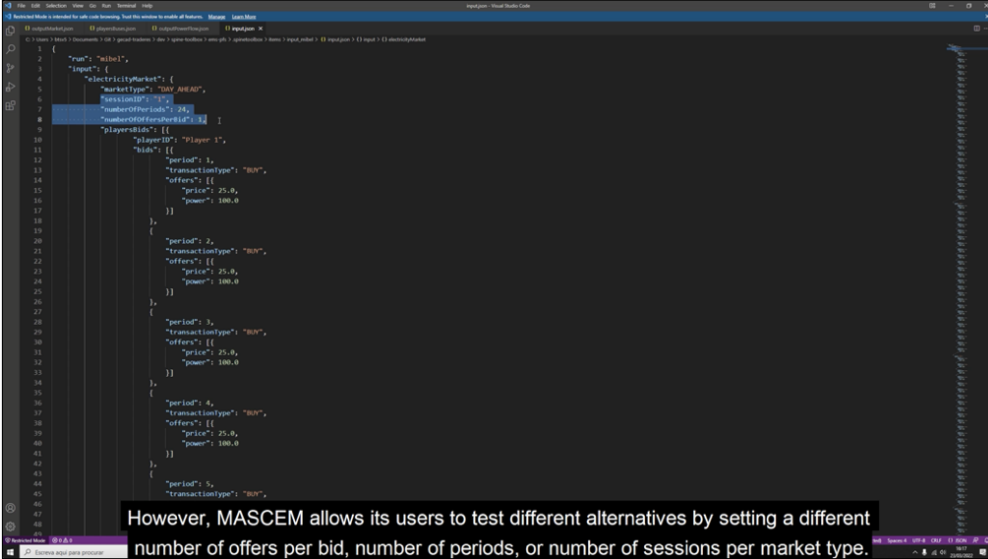


Figure 18. MASCEM overview video tutorial snapshot.



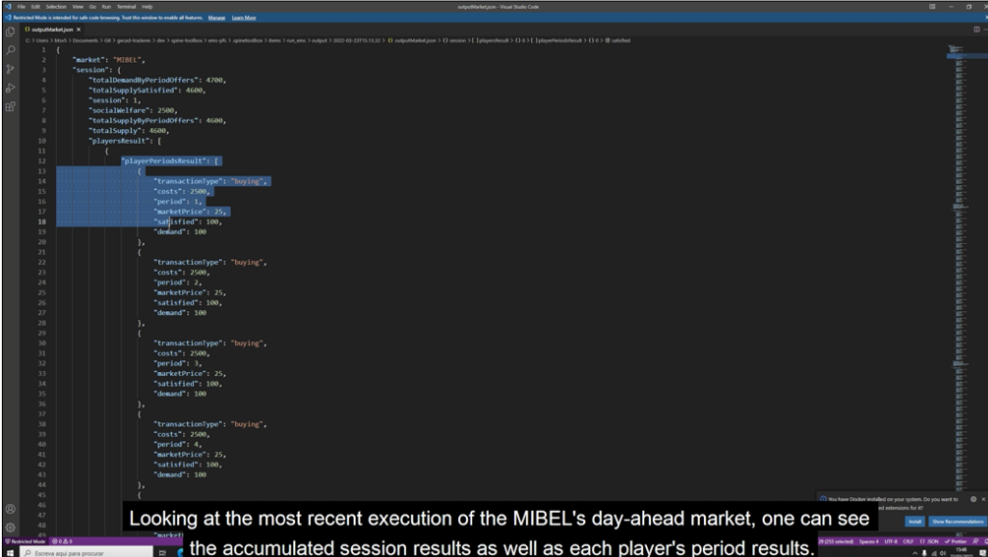
```

1 {
2   "type": "Market",
3   "input": {
4     "electricityMarket": {
5       "marketType": "MIBEL",
6       "sessionID": 1,
7       "numberOfPeriods": 5,
8       "numberOfOffersPerBid": 1
9     }
10    "players": [
11      {
12        "playerID": "Player 1",
13        "bids": [
14          {
15            "period": 1,
16            "transactionType": "Bid",
17            "offers": [
18              {
19                "price": 25.8,
20                "power": 100.0
21              }
22            ]
23          },
24          {
25            "period": 2,
26            "transactionType": "Bid",
27            "offers": [
28              {
29                "price": 25.8,
30                "power": 100.0
31              }
32            ]
33          },
34          {
35            "period": 3,
36            "transactionType": "Bid",
37            "offers": [
38              {
39                "price": 25.8,
40                "power": 100.0
41              }
42            ]
43          },
44          {
45            "period": 4,
46            "transactionType": "Bid",
47            "offers": [
48              {
49                "price": 25.8,
50                "power": 100.0
51              }
52            ]
53          },
54          {
55            "period": 5,
56            "transactionType": "Bid",
57            "offers": [
58              {
59                "price": 25.8,
60                "power": 100.0
61              }
62            ]
63          }
64        ]
65      }
66    ]
67  }
68 }

```

However, MASCEM allows its users to test different alternatives by setting a different number of offers per bid, number of periods, or number of sessions per market type.

Figure 19. MASCEM input video tutorial snapshot.



```

1 {
2   "market": "MIBEL",
3   "session": {
4     "totalDemandByPeriodOffers": 4200,
5     "totalSupplySatisfied": 4000,
6     "session": 1,
7     "totalOffers": 2500,
8     "totalSupplyByPeriodOffers": 4000,
9     "totalSupply": 4000,
10    "playerResults": [
11      {
12        "playerID": "Player 1",
13        "transactionType": "buying",
14        "costs": 2500,
15        "period": 1,
16        "marketPrice": 25,
17        "totalOffer": 100,
18        "demand": 100,
19      },
20      {
21        "transactionType": "buying",
22        "costs": 2500,
23        "period": 2,
24        "marketPrice": 25,
25        "totalOffer": 100,
26        "demand": 100,
27      },
28      {
29        "transactionType": "buying",
30        "costs": 2500,
31        "period": 3,
32        "marketPrice": 25,
33        "totalOffer": 100,
34        "demand": 100,
35      },
36      {
37        "transactionType": "buying",
38        "costs": 2500,
39        "period": 4,
40        "marketPrice": 25,
41        "totalOffer": 100,
42        "demand": 100,
43      },
44      {
45        "transactionType": "buying",
46        "costs": 2500,
47        "period": 5,
48        "marketPrice": 25,
49        "totalOffer": 100,
50        "demand": 100,
51      }
52    ]
53  }
54 }

```

Looking at the most recent execution of the MIBEL's day-ahead market, one can see the accumulated session results as well as each player's period results.

Figure 20. MASCEM output video tutorial snapshot.

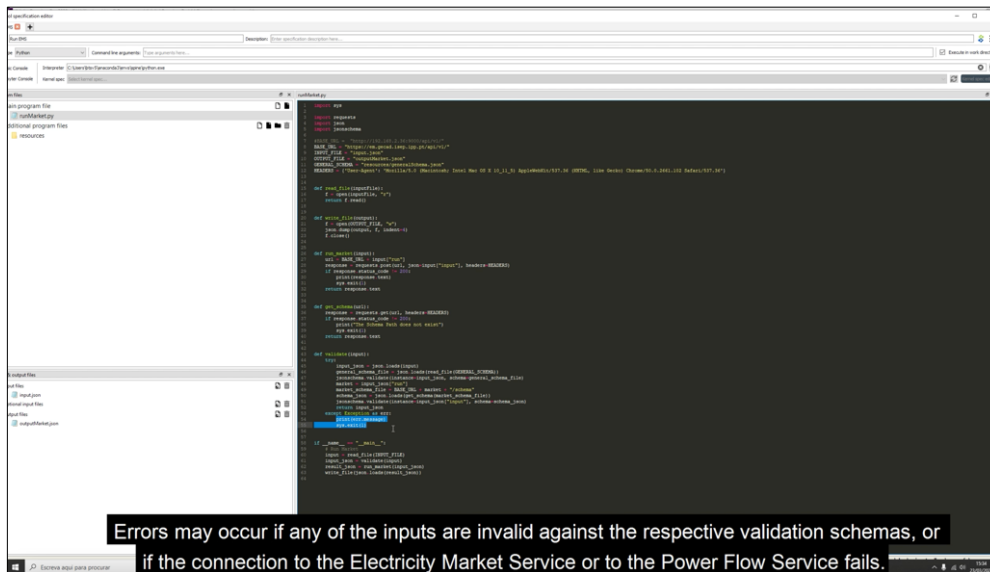


Figure 21. MASCEM run video tutorial snapshot.

2.2.6. REStTrade

The Multi-agent Trading of Renewable Energy Sources (REStTrade) 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. REStTrade tutorial's narration scripts are available in Annex 6 – REStTrade narration scripts. Table 7 presents the REStTrade tutorial videos properties.

Table 7. REStTrade collection of tutorial videos with detailed information.

Category	Length (min)	Ratio	File Size	File Type	Encoders	Online
Overview	1:46	1920x1080	8.2MB	MPEG-4	H.264, AAC	link
Input	3:07	1280x720	22.9MB	MPEG-4	H.264, AAC	link
Output	2:15	1280x720	23.5MB	MPEG-4	H.264, AAC	link
Run	2:07	1920x1080	43.4MB	MPEG-4	AAC, H.264	link

Figure 22 to Figure 25 present a glimpse of the tutorial videos made available for the REStTrade tool, each one of them can be seen in the links available in Table 7.

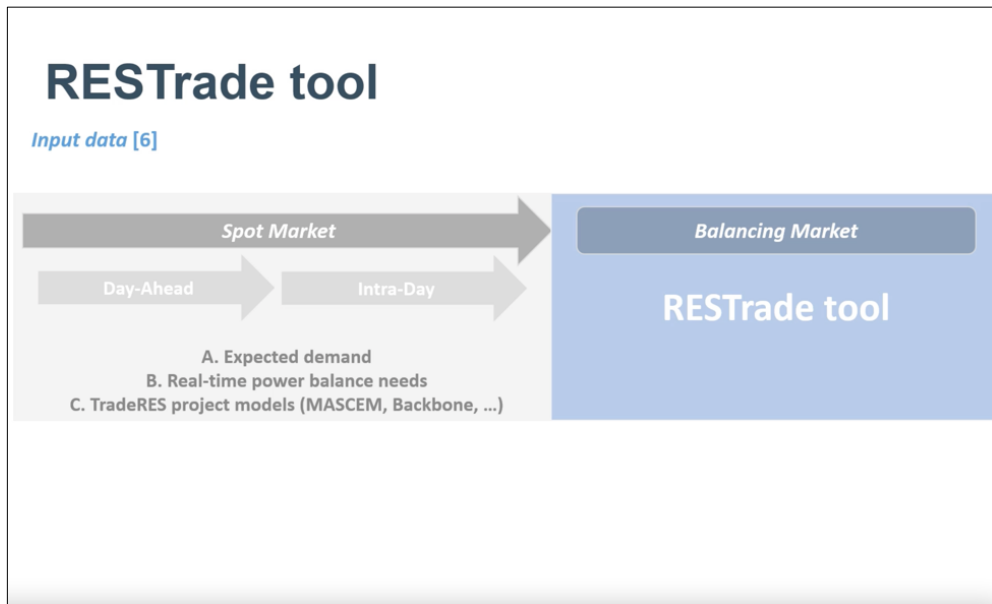


Figure 22. RETrade overview video tutorial snapshot.

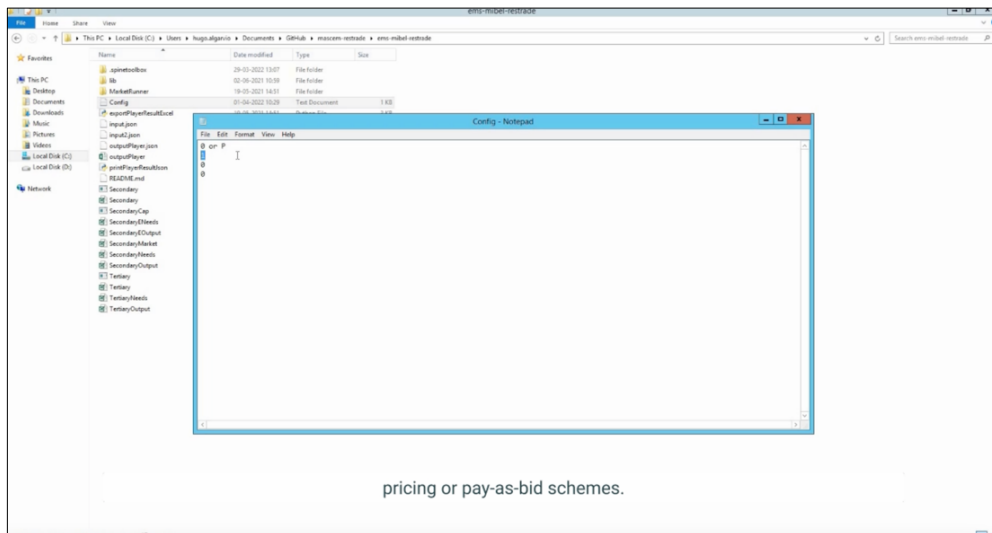
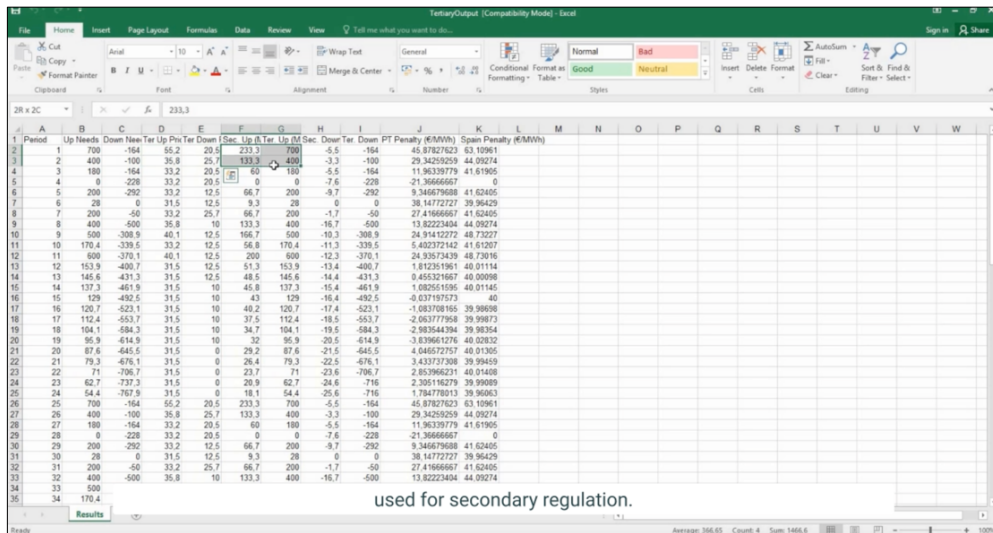


Figure 23. RETrade input video tutorial snapshot.



Period	Up Needs	Down Needs	Net Up	Net Down	Up (M)	Down (M)	PT Penalty (€/MWh)	Span Penalty (€/MWh)
1	700	-164	55.2	20.5	233.3	700	-5.5	-164
2	400	-100	35.8	25	183.3	400	-3.3	-100
3	180	-164	33.2	20.5	60	180	-5.5	-164
4	0	-228	33.2	20.5	0	0	-7.6	-228
5	200	-292	33.2	12.5	66.7	200	-9.7	-292
6	28	0	31.5	12.5	9.3	28	0	0
7	200	-50	33.2	25.7	66.7	200	-1.7	-50
8	400	-500	35.8	10	133.3	400	-16.7	-500
9	500	-308.9	40.1	12.5	166.7	500	-10.3	-308.9
10	170.4	-339.5	33.2	12.5	56.8	170.4	-11.3	-339.5
11	500	-370.1	40.1	12.5	200	500	-12.3	-370.1
12	153.9	-400.7	31.5	12.5	51.3	153.9	-13.4	-400.7
13	145.6	-431.3	31.5	12.5	48.5	145.6	-14.4	-431.3
14	127.3	-461.9	31.5	10	45.8	127.3	-15.4	-461.9
15	129	-492.5	31.5	10	43	129	-16.4	-492.5
16	120.7	-523.1	31.5	10	40.2	120.7	-17.4	-523.1
17	112.4	-553.7	31.5	10	37.5	112.4	-18.5	-553.7
18	104.1	-584.3	31.5	10	34.7	104.1	-19.5	-584.3
19	95.9	-614.9	31.5	10	32	95.9	-20.5	-614.9
20	87.6	-645.5	31.5	0	29.2	87.6	-21.5	-645.5
21	79.3	-676.1	31.5	0	26.4	79.3	-22.5	-676.1
22	71	-706.7	31.5	0	23.7	71	-23.6	-706.7
23	62.7	-737.3	31.5	0	20.9	62.7	-24.6	-737.3
24	54.4	-767.9	31.5	0	18.1	54.4	-25.6	-767.9
25	700	-164	55.2	20.5	233.3	700	-5.5	-164
26	400	-100	35.8	25.7	133.3	400	-3.3	-100
27	180	-164	33.2	20.5	60	180	-5.5	-164
28	0	-228	33.2	20.5	0	0	-7.6	-228
29	200	-292	33.2	12.5	66.7	200	-9.7	-292
30	28	0	31.5	12.5	9.3	28	0	0
31	200	-50	33.2	25.7	66.7	200	-1.7	-50
32	400	-500	35.8	10	133.3	400	-16.7	-500
33	500							
34	170.4							

used for secondary regulation.

Figure 24. REStade output video tutorial snapshot.

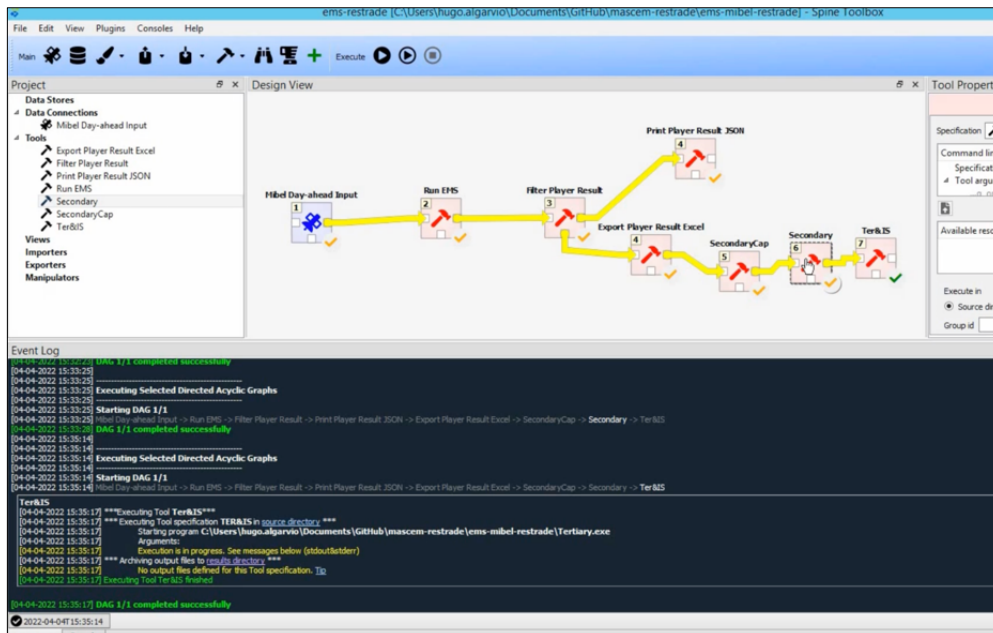


Figure 25. REStade run video tutorial snapshot.

3. Webinars planning and organization

One of WP6 goals is to engage different stakeholders in testing the open-access tools developed in WP4 and collect feedback, information, and recommendations for ~100% RES market designs [9]. To get the stakeholders attention and enhance their involvement and engagement in this process, a set of webinars were planned and run before the end of the fifth semester of the project (July 2022). Preliminary versions of the webinars were developed, tested and evaluated internally by the consortium members in an internal session on May 25th, 2022. In this way, the project's consortium was able to provide concise events, focused on attracting and engaging stakeholders and on achieving the project's goals.

The webinars were planned and organized in pairs and presented in two consecutive weeks. They included the introduction of each model, main operations, discussion of open points and provision of needed clarifications, and a moment of direct interaction with the participants for questions and answers and direct feedback. The outcomes of the July 2022 webinars have been addressed and allowed the improvement of the tools developed in the scope of the TradeRES project.

Until the end of this project, a continuous update to the user guides and tutorials, and the preparation of a new webinar is foreseen. The user guides and tutorials will be made available on the project's website. A final webinar is being prepared and it is planned for between June and July 2024.

3.1 Webinars details

This section presents a summary of the webinars performed in July 2022 and made available online on the project's website.

3.1.1. TradeRES Webinar 1 | AMIRIS & Backbone

The first TradeRES webinar, **Webinar 1 | AMIRIS & Backbone**, was held on July 12th, 2022, starting at 15:00 CET, open to the public. However, registration was required. This webinar presented AMIRIS and Backbone tools, being presented by Christoph Schimeczek (DLR) and Nelli Putkonen (VTT) respectively, and advertised as follows:

AMIRIS (DLR)

Simulations with AMIRIS 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 micro-economic decision aspects of individual actors on energy markets.

Christoph Schimeczek received his PhD in theoretic atomic physics in 2014. In the same year he joined DLR and provided significant research on the European market uptake of electric vehicles using the agent-based vehicle market model VECTOR21 within the European project ICVUE. Since 2017 his research focus shifted towards modelling future energy systems with AMIRIS. His field of expertise are market interactions of multiple flexibility options.

Backbone (VTT)

Backbone is a flexible, bottom-up, open-source energy system modelling tool, that can be used for both invest optimization and operational planning. The GAMS-based tool allows for modelling of highly variable and multi-sectoral energy systems at different scales - from local to continental. With a 6-year history of active development, Backbone is rich in both features and adaptability.

Nelli Putkonen studies the design and operation of future energy systems at VTT Finland with special focus on optimization tools. She has a MSc in Energy Engineering and a BBA in Communications. She is enthusiastic about making energy system modelling more accessible for new modellers, decision-makers, and everyone.

Figure 26 and Figure 27 present snippets of the first TradeRES webinar. The webinar video is publicly available [here](#).

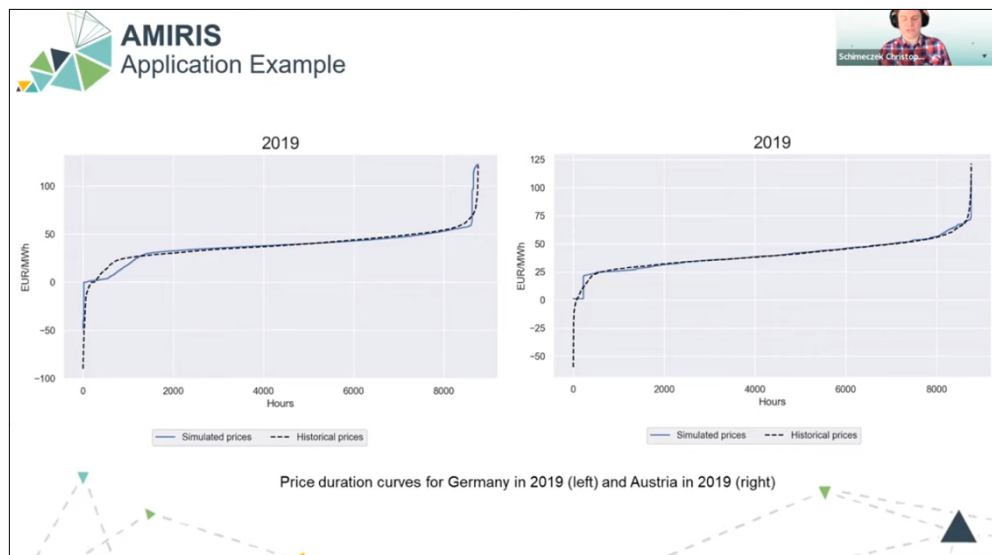


Figure 26. Snapshot of AMIRIS presentation.

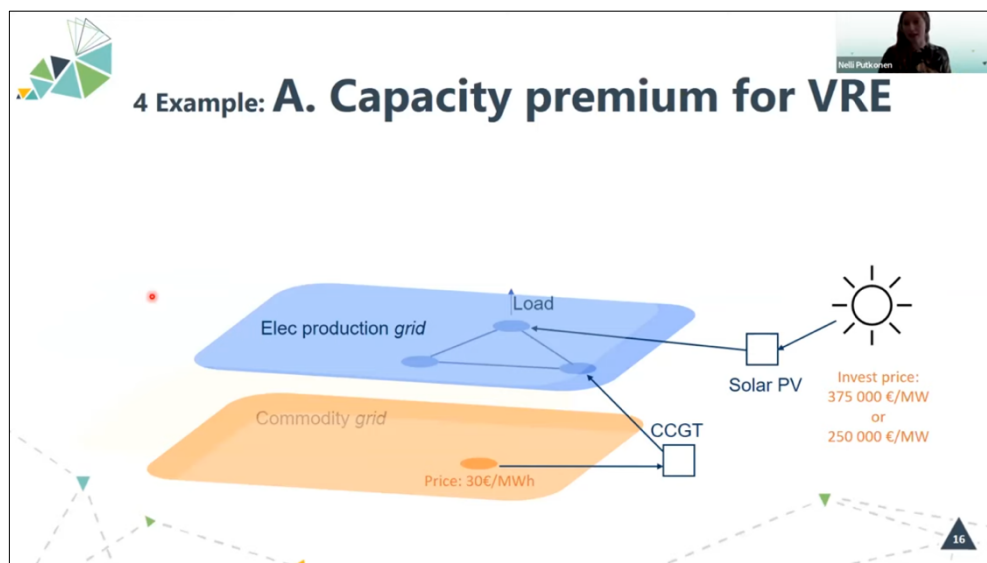


Figure 27. Snapshot of Backbone presentation.

3.1.2. TradeRES Webinar 2 | MASCEM & REStade

The second TradeRES webinar, **Webinar 2 | MASCEM & REStade**, was held on July 19th, 2022, starting at 15:00 CET, open to the public. However, registration was required. This webinar presented MASCEM and REStade tools, being presented by Gabriel Santos (ISEP) and Hugo Algarvio (LNEG) respectively, and advertised as follows:

MASCEM (GECAD)

The Multi-Agent Simulator of Competitive Electricity Markets is a modeling and simulation tool designed to study complex operations of restructured electricity markets by modeling the dynamic market agents, including their interactions, and collecting medium to long-term data and experience to support participants in decision-making based on their characteristics and objectives.

Gabriel Santos received his Bachelor's and Master's degrees in Computer Engineering from the Instituto Superior de Engenharia do Porto (ISEP) and his Ph.D. from the University of Salamanca. Currently, he is a researcher at the Research Group on Intelligent Engineering and Informatics for Innovation and Advanced Development (GECAD). His research interests include multi-agent systems, ontologies, electricity markets, decision support systems, and smart grids.

REStade (LNEG)

The Multi-Agent Trading of Renewable Energy Sources includes different models, namely traditional energy markets and energy balance, and an improved version of them. It supports traditional dispatchable power plants, variable renewable energies, and demand-side actors in participating in system balancing. Additionally, it uses both marginal pricing theory and public offer schemes to define prices in these markets.

Hugo Algarvio is a contracted researcher at LNEG, holds a Bachelor's and Master's degree in Electrical and Computer Engineering, and a Ph.D. in Sustainable Energy Systems. His main research interests focus on using multi-agent technology to model and simulate electricity markets and market agents' behavior.

Figure 28 and Figure 29 present snippets of the second TradeRES webinar. The webinar video is publicly available [here](#).

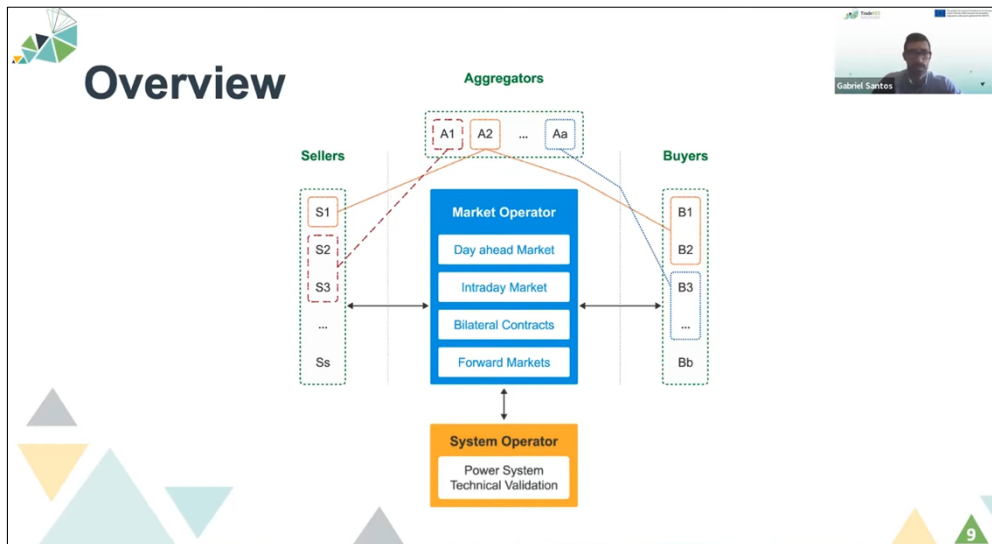


Figure 28. Snapshot of MASCEM presentation.

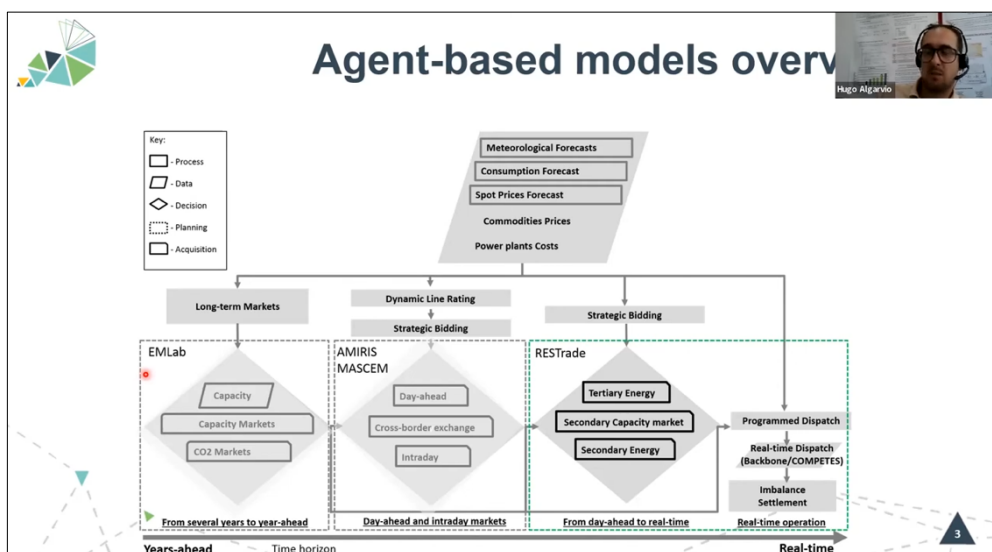


Figure 29. Snapshot of RESTRade presentation.

4. Conclusion

This deliverable reports a collection of edited material from tutorials and webinars developed and organized in the scope of the TradeRES project, that along with the user guides, have been used to engage stakeholders and ask for their comments and feedback to improve the models and tools developed within the project. The goal was to provide stakeholders easy access to the materials, to engage different stakeholders in testing the open-access tools from WP4 and collect their feedback, know-how, and recommendations to improve the project's models and tools.

The collection of materials delivered in this report aims to provide, complement, and enrich the information already available in the user guides [2]. All the materials created will continue to undergo progressive updates as the models are further developed, being publicly available online.

As a next step there's a last webinar being prepared and organized, planned for the upcoming months, where new models will be presented addressing the improvements made, enhancing the results of task T6.3, i.e., the test of the open-access tools by different typologies of stakeholders.

Finally, tutorials will undergo progressive updates as the tools are further developed, making them publicly available in the project's website.

References

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

[2] TradeRES project consortium, Deliverable 6.2 “D6.2 – User guide for TradeRES models and tools (D6.2.1) - 2nd Edition”, 2022.

[3] The Grant Agreement n° 864276 and all its annexes — TradeRES.

[4] AMIRIS - The open agent-based electricity market model. Available in <https://dlr-ve.gitlab.io/esy/amiris/home/>. Accessed on March 21st, 2024.

[5] Backbone. Available in <https://gitlab.vtt.fi/backbone/backbone>. Accessed on March 21st, 2024.

[6] EMLab – Energy Modelling Laboratory. Available: <http://emlab.tudelft.nl/>. Accessed on March 21st, 2024.

[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 on March 21st, 2024.

[8] MASCEM - Multi-Agent Simulator for Competitive Electricity Markets. Available in <http://www.mascem.gecad.isep.ipp.pt/overview.php>. Accessed on March 21st, 2024.

[9] TradeRES - New Markets Design & Models for 100% Renewable Power Systems. Available in <https://traderes.eu>. Accessed on March 21st, 2024.

Annex 1 – AMIRIS narration scripts

Part I – Overview

Welcome to this first overview of the “Agent-based Market model for the Investigation of Renewable and Integrated energy Systems” – AMIRIS. AMIRIS simulates the trading at day-ahead energy markets, the operation of power plants based on the trading results, and strategies for the bidding and operation of flexibility options. It models, from a business-oriented perspective, the behaviour of actors in energy systems - also considering their uncertainties. The temporal resolution of AMIRIS is hourly by default, but can be increased to, e.g., 15 minutes or 5 minutes in the configuration. The spatial resolution is fixed to market zones. The course of AMIRIS simulations is determined by configuring its agents and their interactions. Markets are an important agent type in AMIRIS. These do not follow own interests, but only serve to determine market prices, e.g., energy prices on the day-ahead market, and fuel / CO₂ prices on the fuels / CO₂ markets. Power plant operators control a type of either renewable or conventional plant. They are responsible for estimating operational costs and dispatching of the power plants in their portfolio. Traders are responsible for the bidding process. Thus, they calculate bids based on predetermined or dynamic bidding strategies and interact with each other at the day-ahead market. Flexibility providers, such as storage operators, are a special type of trader. They use forecasted data to develop smart bidding strategies that achieve, e.g., maximum profits. Forecasts are generated by forecaster agents which survey the other agents for their bidding behaviour. The impact of policy instruments on the simulation is controlled via dedicated agents as well, e.g., the support policy agent. Interactions of all these agents are shown here, distinguishing between monetary transactions, energy flows and information exchange. Please check out: <https://dlr-ve.gitlab.io/esy/amiris/home/>. There you find even more tutorials and helpful information about AMIRIS.

Part II – Run

Welcome to this second part of the AMIRIS tutorial. This section covers setup and execution of AMIRIS using the Spine toolbox. To run AMIRIS you require a recent python installation. In addition, you will need a Java Runtime Environment. In this tutorial, we use Open-JDK version 11 - but Java version 8 and 17 were also tested and work just fine. We start with setting up a Python environment. You can use, e.g., venv or mamba for this task, but we employ conda. Then, we proceed with installing required packages using pip. Open a conda shell and create a new environment by entering “conda create -n amirisEnv python=3.8”. Proceed and once finished, activate the environment with “conda activate amirisEnv”. Within the newly created and activated environment: enter “pip install fameio”. This will take a few moments to complete. Then, open a browser, navigate to the Spine-Toolbox repository, scroll down the ReadMe file and copy the link to the latest development version. Use this link to pip-install spine-toolbox. The process might take about one minute to complete. We now need to download or clone the AMIRIS workflow for Spine-Toolbox. In any case, open the TradeRES repository on GitHub using your browser. Navigate to the “toolbox-amiris-demo” project. Click on “Clone” an either select the “download” or “clone option. We will use cloning here. Copy the path to the clipboard. In the shell, use “git clone”

and that path to obtain the workflow files. Now, start-up Spine-Toolbox. Once open, navigate to File -> Settings -> Tools. Make sure that the path to the Python interpreter matches the environment you created for running AMIRIS. Adjust if necessary and close the dialogue. Now, open the downloaded project “toolbox-amiris-demo”. You may also use the “open-recent” entry, if you opened the project recently. Once the project is loaded, click the “Play” button to execute the workflow. Now that you know how to run AMIRIS in the Spine-Toolbox, learn how to modify input files and how to inspect simulation results.

Part III – Input

Welcome to the third tutorial on AMIRIS. This video covers input parameters and their modification. Open your amiris python environment. In this case via “conda activate amirisEnv”. Start Spine-Toolbox and open the “toolbox-amiris-demo” project. Click on the “Scenario” item and double-click the “scenario.yaml” to open the associated file. The scenario.yaml is >the< config file to define an AMIRIS simulation. It consists of at least 4 sections. “Schema” defines the simulation’s capabilities. “General properties” control overall simulation parameters. “Agents” is a list of all entities in your simulation and their parameters. And “Contracts” define how and when the agents interact with each other. In “Variables” optional parametrisation templates can be stored. Let’s first have a look at the “General Properties” section. The most important part there is “Simulation”. Its first parameter “StartTime” defines the first time within your simulation. Accordingly, “StopStime” tells the last simulated time. “RandomSeed” controls the overall seed for pseudo-random number generation in the simulation. Ignore the other parameters “RunId” and “Output” for now. Expanding the “Agents”, you see a list. Each item represents an agent in the simulation. This scenario comprises about 50 agents. Some of them share the same type, but they are individual agents, that may differ in their attributes. Each agent has at least a “Type” and “Id” specification. Its “Type” relates to the Java class name that defines the agent’s capabilities. “Id” is a 4-byte positive integer unique to each agent. No two agents may have the same ID within a simulation. All message exchange is coordinated using this ID. Keep this in mind for the “Contract” definitions later on. Configuration parameters specific to each agent can be defined below the keyword “Attributes”. The inner structure of these Attributes depends on the Type of agent. It can be a plain map of keys to values or a more complex hierarchy. In this case, the “FuelPrices” item resembles a list of items, each specifying a FuelType, associated Price and ConversionFactor. However, Attributes can also take a simpler structure as in the case of the “EnergyExchange”. Here, only two “key: value” pairs are required. Some agents, like the “SystemOperatorTrader”, do not require any Attributes at all. To keep the “Scenario.yaml” file tidied-up, the “Contracts” definitions are not directly part of it. Instead, we use the “!include” command to load content from other YAML files into the scenario. To see how this works, open the “toolbox-amiris-demo” folder and navigate to “data”. Open the “Germany2019” scenario folder. There you see the scenario file we are editing right now. You can find the subfolder’s name “contracts” also in the string of the scenario’s “!include” command. This string tells which files to include relative to the location of the scenario file. Opening such a contract definition file, you can see the “Contracts” section. Only this part of the file is imported into the scenario - as specified by the second string of the “!include” command. The “Contracts” section contains a list of contracts. Each contract comprises five elements. “SenderId” refers to the unique Id of the agent that will

send the message. Likewise, “ReceiverId” identifies the agent receiving the message. “ProductName” defines the content of the message. The first time a message is sent in the simulation is given by “FirstDeliveryTime”. Finally, “DeliveryIntervallInSteps” tells how many seconds pass until another message is sent. Similar to the “Contracts”, the “Schema” is also not a direct part of the scenario file. It is, again, imported using the “!include” command - but this time referring to a specific file “schema.yaml”. It is located in the same folder as the scenario file. Opening the schema file, you find the definitions for each type of agent: the “Attributes” they require (also defining their structure), and the “Products” they can emit in their messages. You will find the “products” names from the schema to match the “Product-Name” in the contract definitions. The sending agent defines the available products. Now, you might have noticed earlier, that in the definition of the FuelsMarket’s “Attributes”, the “Price” value for some items took floating point numbers, and for other items was represented by a file-path string. Navigate to your scenario folder and enter the “timeseries” sub-folder. You will find the file mentioned in the string there. Open it, to see for its structure. Its two columns are separated by a semicolon. The first column is a timestamp that defines at which point in time the floating-point value in the second column shall be valid. Notice the underscore between date and time-of-day parts of the timestamp. Interpretation of timeseries varies in AMIRIS. Most agents, however, interpolate values at missing timestamps. This grants high flexibility when defining timeseries, since one may use any (even irregular) time stamp entries. If a timeseries shall take a constant value, you need not define a new timeseries file that has but one entry. Simply replace the file-path string with the desired constant value, as is the case for the “NUCLEAR” and “LIGNITE” fuel prices here. You can find further and detailed information on AMIRIS input parameters in our wiki. Browse to our GitLab repository and click on the “wiki” badge. The “Model Classes” section highlights each type of agent and their hierarchy. Check out the “RenewablePlantOperator” - a child of “PlantOperator”. Besides other descriptions, there is the “Input from file” section. It describes input parameters used in the scenario.yaml for that type of agent, “Set” and “EnergyCarrier”. Looking at the scenario definition, the agent with ID 53 has these input parameters. It does not define a “Set” parameter, as it is optional. But it features the additional “YieldProfile” which is parameter for the child class “VariableRenewableOperator”. With all that background, let us go back to the Spine-Toolbox. Open the scenario file, navigate to the “CarbonMarket” agent, replace the timeseries for the “Co2Prices” with any value you like (e.g., 100 €/t), save the file, and run the workflow. See our next video to learn about the AMIRIS outputs and how to inspect the results from workflow executions.

Part IV – Output

Welcome to the fourth tutorial on AMIRIS. This video covers simulation outputs. In case you haven’t already done so: Open the AMIRIS Python environment and start the Spine-toolbox application. Once loaded, open the “amiris-spine-toolbox-demo” project. Search for the “Results Data Store” item in the workflow and double-click it. The results of previous runs are shown in the DB editor’s “Scenario tree” - identified by timestamp. This name matches the “alternative_name” column in the top view. Result columns from all scenarios are shown in this top view. Focus on results from one scenario, by selecting it in the scenario tree to the right. Open the results for “ElectricityPrice” by double-clicking the “Time series” entry in the “value” column. In the new window, you can see time-value pairs to the left and

a graph depicting the results for the whole simulation period on the right. Now, change the scenario and also open its “ElectricityPrice” results. You can easily recognize the differences in the electricity price compared to the first scenario: Both, scale and shape of the result graphs differ due to the changed CO2 price. So far, the results only comprise outputs from the EnergyExchange agent. However, previously many more agents were shown to be part of the simulation configuration. What about their outputs? To add output from further agents, go back to the “Design View” and double-click on the “Combine AMIRIS results” script. The option “Config.AGENT_LIST” defines which agent types to consider. Open the scenario.yaml and copy the name of that agent type you want to see outputs for. Add that agent type to the AGENT_LIST and save the script. Press and hold the “CTRL” key and click on “Combine AMIRIS results” and the two subsequent items in the workflow. Hit the smaller play-button to re-execute the selected items. Reopen the Database editor and narrow down the scenario selection to the newest results. We can confirm, that the “SystemOperatorTrader” outputs are now included. To inspect the raw outputs of any item in the workflow: Right-click it and select “Open results directory”. Results of each workflow execution are placed in a separate folder. You can, e.g., open the script results and see them in .csv format. By the way: ignore the two last items in the workflow for now - those only transcribe results to a different representation. That’s it for the AMIRIS tutorials. To gain even more insights, visit the online resources on AMIRIS, e.g., our web site, repositories, Javadoc page or wiki.

Annex 2 – Backbone narration scripts

Part I – Overview

This is an overview of Backbone energy system modelling tool.

Backbone is a multifunctional energy system modelling framework that can be used for creating different kinds of energy system models.

It is completely open-source and under active development. You can clone Backbone to your own computer using any git software and cloning from gitlab.vtt.fi/backbone. Backbone is written in GAMS coding language, so all you need is a GAMS licence to start creating models with Backbone.

Backbone is a useful tool for both energy system planning and operational modelling. It is ideal for creating regional or national-level energy system models with multiple energy sectors, like electricity, heat, natural gas, and hydrogen. But because of general and flexible composition, it can be used for modelling larger or smaller systems with different energy vectors.

The core of Backbone is cost optimization. It's solving the unit commitment problem at each hour of the year and optimally committing the generation of units and transfers in the system while delivering energy demand. Alternatively, it can optimize investments in new units based on representative periods. Or you can use them together to first optimize investments, and then model the operation of the system with the new investments. Also, the temporal settings are flexible, so you can use something else than hourly resolution and one-year modelling scale.

The benefits of Backbone include flexible formulation, allowing for modelling different types and scales of systems. Also, you can use a wide range of parameters and constraints that are important to consider in future energy system. These include for example forecasts and both short- and long-term variability (stochastics), as well as electricity grid balancing constraints like operating reserves and capacity adequacy constraints.

To find out how to work with a simple demo model created for getting started with Backbone, please see our other videos and visit the GitLab Wiki.

Part II – Input

This video will describe the input data of the demo1 model. To get to this point, please see the “how to install backbone and run demo model” video. I have opened the input data excel file to study and edit the input data. Comprehensive instructions of the contents are provided on instructions and maps sheets. On INDEX sheet, there is a list of all sets and parameters that are translated to Backbone in GAMS in order to define the grids, nodes and units of our system.

The sheets in white and grey are input sheets, so you define the objects and fill the parameters there. Most important ones are node data - where you define grids and nodes - and unit data - where you define units.

In node data, you will notice that one row equals one node. Nodes are the unique inputs here, and the grid they are associated with is specified next to it. One row equals one node. The first parameter for our node is the selection of node balance. Node balance TRUE means the energy balance for this node is on - it is used for nodes that have to have the

same amount of demand and generation (or input and output) at each time step. Use price TRUE on the other hand means that the commodity comes from outside the model, and its price is defined.

Next, we define storages. In Backbone, only nodes can store energy, not units. To activate storage, energy stored per unit of state is flagged as 1. From now on, we will not go through individual parameters. To understand what they are used for, there are two places: First, on instructions sheet, we have a description of each node. Second, under backbone and inc, there is 1a_definitions.gms -file where you can find descriptions of each parameter.

Next on node data, we can specify emissions related to using that node - groups for summing up results, giving additional constraints, or for example, linking reserves to nodes. Finally, nodes can have transfer lines between other nodes on the same grid. See Map of nodes on “Maps” sheet for a visual representation of grids and nodes in the demo model.

Let's move on to unit data and consider the first unit, linking biomass node and district heat node. Units are defined in “unit data” sheet. You will first notice that the same unit is defined twice: for biomass node input and for district heat output. Not going through all the possible parameters but listing the mandatory ones: you will give your unit capacity, a conversion coefficient, a unittype, efficiency calculation setting between OFF / linear / MIP for all time levels, availability, efficiencies and corresponding operating points. The efficiency and operating point selections are linked to the calculation setting, so be careful. Additionally, you can assign a multitude of unit parameters, like operating costs, possible operational constraints, groups, reserve properties, maintenance breaks, calculation aggregations and investment constraints. Notice, that different types of units (like flow units or units with multiple outputs) may have different mandatory parameters associated with them.

Finally, some units can be invested into, and some can't. You can allow investments into any unit by filling out the investment parameters: unitsize, investment cost, unit count, max unit count and become available at time step one. Additionally, maximum and minimum investments or other constraints can be set.

To add a new node or a new unit, simply input the name of your node or unit on any row below, assign its connections to grids and nodes, and give it parameters. The checks in grey columns will help you spot errors in inputs. The other necessary sheets to define these sets and parameters to GAMS, like shown on INDEX, are automatically updated. To understand the output of the demo model, please see the next video.

Part III – Output

This video will describe interpreting the output data of the Backbone demo model. To run the model and generate the results, please see previous video on how to install backbone and run demo model.

After running the full year invest and schedule run, the model has first optimized the capacity investments on new units and new transfer lines. The calculation is based on representative sample weeks, that are selected in “investlnit.gms”. The invest run has generated the output file “invest_results.inc”. When we open the file, first we have the number of new subunits that were invested into. Looking at inputdata, we see that some subunits are only 1 MW in size while others are up to 500 MW. The number of subunits is multiplied with the unitsize and the added capacities are listed next. Some investments to new storage

capacity and transfer connections are also made. Some of the investments are forced, and others can be bound to one another, as specified in input data excel. After this, the invest parameters are turned off.

When running the combined invest and schedule run, these new investments are added to the model, and the full year schedule run is optimized with the new capacities. The least overall cost unit commitment problem is solved for each hour, resulting in dispatching the units in a cost optimal way, while honouring the demand and constraints. Let's open the results.gdx in GAMS editor to look at the results.

There are many tables, some represent time series and some aggregated results. For example, r_gen shows the energy generation for each unit at each time step. The table can be modified by clicking and dragging. Here we see that for example, the wind unit generated 543 MWs at timestep one. Notice, that if a certain unit doesn't generate at a certain timestep, then that timestep is not listed in the timeseries.

Then, at r_gnuTotalGen, we see that during the whole year, the wind unit generated this many MWhs. The name of this result table translates to "results in grids, nodes, units - total generation".

Similar tables exist for transfer lines...

states in storages...

and for example, emissions...

and costs.

Many of the results are not in a very useful form, and for further result analysis, the results can be converted to excel using GDXXRW conversion tool. This bit is however not included in the demo1 package, and it is up for the user to process the results.

This video concludes the short introduction to Backbone energy modelling framework, and demo1 model created for it. Notice, that we have not gone through all the details, and to learn the usage of Backbone will require spending many hours studying. If you are interested in starting to use Backbone in your own research project and want further introduction, please contact this address: neli.putkonen@vtt.fi.

Part IV – Run demo model

This video is about learning how to get Backbone and the demo model running on your own computer. Before we start, you need to have done three steps: First, you have installed GAMS. Second, you have installed any git software. And third, you have added GAMS as a path variable in Windows. This will later allow you to run GAMS and Backbone using a command file.

To install Backbone, first we have to clone the Backbone repository. I will do it using a command line git and an empty folder. Let's first create the empty folder. Then let's open command line git. And cd...to our created folder. I will then type "git clone", select the address of the remote repository and specify I want to download branch "release-2.x".

We have now cloned Backbone to our local repository.

To get the demo model, we go to Backbone GitLab, then to Wiki and example models. We call this the demo1 model. The demo1 model is created for demonstrating the different features of Backbone in a small demo system.

The structure of the demo1 model is pictured here. We have three different types of elements here: grids on the bottom for different energy carriers, nodes - the squares - located on those grids, and units - the circles - converting energy between grids. Some nodes represent the whole country, while some represent an individual city in that country. For example, unit 11 here is converting nuclear fuel to electricity grid. Unit 8 and 9 are not using a fuel, but a flow, like wind or solar. Some units don't generate, they just convert or transfer energy between nodes - like unit 15, an electrolyser unit, or unit 7 a charge and discharge unit for storage.

To get the demo1 model, we save the zip file inside backbone-folder in the local repository and unzip to a folder called demo1. All these files define the demo model input data and run instructions. The most important files are the inputdata demo1 excel, where you define the grids, nodes and units of your system. The second most important ones are scheduleinit and investinit where you define the temporal settings of the model. To find out more about editing the input data, please see the next video or get to know the Demo 1 instructions.

To run the demo1 model, we use any of these five command files starting with "run". There are five options: reduced for working with GAMS demo licence, 1 week test for quick test runs, schedule for full year operational schedule run, install for optimizing investments, and invest and schedule for first invest-optimizing and then running full year. Let's double click the test.

The demo model is now finished running and three.gdx.files were created. The important one is the results.gdx that we can open with GAMS editor. To find out more about understanding the output, please see the video on that.

Annex 3 – EMLabpy-AMIRIS narration scripts

Part I – Overview

In this video, we will give an overview of the EMLabpy AMIRIS co-simulation.

EMLabpy is an agent-based model that can simulate myopic investment and decommissioning decisions with limited information and no assumption of equilibrium. It allows testing the long-term effect of energy policies.

EMLabpy can simulate capacity remuneration mechanisms but have a limited representation of flexibilities. On the other side, AMIRIS is an agent-based model that simulates dispatch decisions with an hourly resolution and can represent multiple types of flexibilities. Through a co-simulation, we use the strength of both models. We use the Spine Toolbox to co-simulate EMLabpy and AMIRIS.

Spine Toolbox is a workflow management tool that allows to exchange data between models, keeping the data in a sqlite database and modifying specific entries. It also enables scenario management and workflow loops. Here we represent the conceptual workflow of the co-simulation.

EMLabpy simulates generation expansion from an initial set of power plants. Nevertheless, if the initial capacities are insufficient, severe shortages may occur in the first simulation years. For this reason, the workflow starts with an initialization investment loop. First, an envelope by module exports the data to AMIRIS. Then AMIRIS clears a future market, and the results are read by EMLabpy to make the investment decisions.

In the EMLabpy investment module, each candidate technology is checked for its technical limits. If these are not reached, then the net person value is calculated, and if it is positive, then this technology is set as investible. This is done for all technologies. Finally, the technology with the highest net person value is being invested. This loop is repeated until all candidate technologies are no longer profitable.

Each simulation year commences with decommissioning of old and non-profitable power plants. Then the data is prepared to be read by AMIRIS, which then clears the market on an hourly basis. After this, each power plants loans are registered, and their financial performance are assessed in the financial results module. Next, the investment loop begins as explained before, the market data for a future year, for example, four years ahead is exported to AMIRIS. AMIRIS then clears the market, and then EMLabpy makes the investment decisions. Finally, a capacity remuneration mechanism module calculates capacity market price or strategic reserve payments.

This is how the workflow is seen in the Spine Toolbox. The blue symbols represent the input data, which will be explained in the next video. The purple symbols are the importers that map the data into the EMLab database, which will be accessed by the modules linked with the yellow arrows. The database on the right is a copy of the EMLab database. All the red symbols refer to execution files. This initialization modules prepare the data as required by EMLabpy. This is the initialization investment loop. And this is the yearly simulation loop, which commences with a decommissioning step. Then the data needed from AMIRIS is exported. AMIRIS is then cleared, and then the results are saved in a second database. The AMIRIS results are used to calculate the financial performance of the power plants,

and after this comes a yearly investment loop. Finally, the capacity remuneration mechanisms are calculated.

Thank you for your attention. For more information, please visit our trade risk website where you can find our publications, including the publication: “*Can an energy only market enable resource adequacy in a decarbonized power system?*” which presents a case study for this workflow.

Part II – Run

First, we need Anaconda, not mini Conda, because we had previously issues with it. Then it's optional to install GitHub, but it is useful because you need more repositories and it's easier to update them later on. Then you need EMLabpy AMIRIS code.

You can go to the link indicated. In the link, you can find in the Readme file the instruction to prepare that Python environments with an Anaconda. There is also a link to go to the Spine Toolbox repository. You also need to download this code and install the anaconda environment as indicated in the readme file of Spine Toolbox.

It's worth mentioning that AMIRIS executable is downloaded with the EMLabpy-AMIRIS code, but if you need want to see more about the code, you can also go to the GitLab AMIRIS repository.

Finally, you need to install Java 11 to trigger the co-simulation, you can follow the instructions as indicated in the Readme file of that toolbox-emlabpy-amiris. Or an easier way. Once you have downloaded the code, you can just double click in this file update emlabpy spineToolbox. This will prepare the environments and will trigger automatically the toolbox.

You can open the project toolbox-amiris-emlabpy add in the settings tools, make sure the Jupyter Console is, selecting the Conda emlabpy environment. The folder where the Conda executable is should be saved in this place. Make sure that all the modules have selected the kernel specification with the emLabpy conda environment. All modules should have emlabpy kernel and the only ones different are AMIRIS modules. These are indicated with that emlabenv python executable. You only have to do that specification the first time.

Then you can just run the project running the play button and wait for some hours.

Part III – Input

Here we will learn about the input data. As we mentioned in the previous video. Here you can see the most important input files. The first one is the configuration file. Here the user can specify the most important parameters of the scenario. For example, the country, the start year, the end year, which capacity remuneration mechanism the user wants to simulate, if there would be an initialization investment the look ahead years for the simulation, and so on and so forth.

As you can see, we try to give some explanation in the column C. Then we have the import EMLab parameters. Here are the files related to the EMLab investment decisions. For example, the capacity market parameters, strategic reserve operator parameters, capacity subscription, electricity spot market parameters candidate power plants, more specifications about the technology and so on. Then there are the power plants excel. Power plants can be grouped by age and technology.

The final Excel file is a TradeRES common database. Here is a TradeRES data, for example, the investment cost, fixed cost, fuel prices, technology specifications, and so on.

Thank you, and if you have any questions, please contact us.

Part IV – Output

Welcome to the last tutorial of EMLabpy-AMIRIS. Here we'll explain about the output data.

Once your workflow completed, you can find the results in the main folder, you go to toolbox AMIRIS EMLab, and there you find the temporal results with the simulation results. If you click in one of the Excels, you find the AMIRIS results per year. For example, you can find the electricity price in euro per megawatt, the total awarded power in megawatt, and the aggregated hourly generation per generation type.

In this folder, you can also find, the EMLab AMIRIS sqlite databases that we explained in the previous video. And these databases can be read by a Python script that shows results already in graphs.

You go to EMLabpy/plots/scenarios you will not have as many scenarios. The one that which the last word temporal. There you will find the results, and here you can find the, the most important plots, for example, annual generation per year the install capacity, the investments and the commissions per year awarded, capacity, capacity market clearing price, volume, revenues per capacity type, and so on.

Feel free to explore and modify the plots modifying the plots python file in the project.

Thank you, and if you have any questions, don't hesitate to contact us.

Annex 4 – COMPETES-EMLabpy narration scripts

Part I – Overview

Welcome to this tutorial in which we will be giving an overview of the EMLab–COMPETES tool.

This tool has been developed in the TradeRES project, and it links the two different models: COMPETES and EMLab.

First, we start describing COMPETES. It is a power system optimisation and optimal dispatch model that seeks to minimize the total power system costs of the European power market.

There are two modules:

- A transmission and generation capacity expansion module in order to determine and analyse least-cost capacity expansion with perfect competition;
- A unit commitment and economic dispatch module to determine and analyse least-cost unit commitment (UC) and economic dispatch with perfect competition.

It accounts for the technical constraints of the generation units and transmission constraints between the countries.

The image on the right shows the geographical scope of COMPETES.

EMLab is an agent-based tool that was developed with the purpose of investigating the long-term effects of climate and energy policies.

In this agent-based model, the agents are power companies with limited information about the future system and thus make imperfect investment decisions.

EMLab can simulate a CO₂ market and capacity mechanisms (CM) such as strategic reserve, capacity market, and capacity subscriptions.

The investment decisions are based on the profitability of possible power plants in a future need.

We use COMPETES dispatch and investment modules to decide on operational decisions and how will investments look like over a long-time horizon

In EMLabpy we capture the CO₂ market which 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 which adds an extra payment to generators, so they recover their investments.

Part II – Run

In this video we will explain the Emlabpy competes implementation in Spinetoolbox and how to run it.

Besides python and the libraries from spine toolbox. Also, Aimms, Competes and Microsoft Access have to be installed. The code for the soft linking of Competes Emlab can be downloaded from the traderes github website. Competes is not open source, but the python version of Emlab, called Emlabpy, is available.

Here we see the implementation of the workflow that was explained in the first video. In spinetoolbox the blue symbols refer to the input data which was explained in the second video. The purple symbols are the importers which map the data into the databases and the red symbols refer to the execution files. On the top we have the configuration parameters. 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 must be imported into the databases. This information is fed into the configuration database, and it will be accessed by all the modules linked with the yellow arrow.

In this project we have two databases, the Emlab and the Competes database which store the data to run the respective models. These databases are mirrored on the right side. The Initialization Preprocessing block prepares the data so that it is equivalent between both models. It also sets the generator status to operational or dismantled considering the simulation year. This module should only be run one time. Next, the Init EMLAB clock sets the system clock tick parameter to 0. The main CO2 algorithm calculates the CO2 price is executed before the COMPETES dispatch and investment algorithm and the capacity market module is executed after the Competes modules.

The translation script “DB EMLab to DB Competes” extracts the CO2 and capacity market results from Emlab and transfers them into COMPETES. Because COMPETES don't have an interpretation for capacity market revenues, these are considered as fixed operating and capex cost reductions for some power plants. Similarly, the COMPETES output to EMLAB DB transfers the results from Competes into Emlab. The ‘Prepare COMPETES’ script feeds the Microsoft Access databases which will be read by AIMMS to run COMPETES. On the bottom the COMPETES dispatch and investment blocks trigger these algorithms in AIMMS through an HTTP request.

This project was developed before the looping functionality of Spine Toolbox was available. Therefore, the clock module increments the system tick by one and the “sleep and click” module triggers the next year iteration.

Part III – Input

In this video we will explain the input files needed to run the soft linking of Emlabpy and competes. First, let's have a look at the config files. These are being stored in the resources\configurations folder of the Spine Emlab Competes project. Let's go to that folder. Here you see the three files. And let's see, for example, the Emlab config file which contains the order in which the variables should be imported into the Emlabpy repository. Then, the Competes config file, classifies the data as it should be imported into the Competes database. Finally, the coupling config file, contains the start year, end year, time step and look ahead year for the investment algorithm.

Let's proceed to see the input files. These are these blocks on the left and these blocks will be replaced in the next months by the Traderes database which is being completed. First, let's have a look at the Emlab init data. Here is the data necessary to run specifically the Emlabpy scripts. For example, we have the capacity market data. Then we have the shared data which stores information for both models. For example, the VRE technology costs. And finally, we have the Competes specific data which is necessary to run the dispatch and the investment algorithm. For example, here are the renewables profiles. All these three files can be found in the folder resources\data. Finally, we have the Microsoft

Access files which are necessary to run Competes, through AIMMS. Initially we have the Competes empty files and these files are being filled through the script. Into these files, which will be finally fed into Competes and these are being fed once this “prepare COMPETES” block is being run. The complete workflow will be explained in the fourth video.

Part IV – Output

Welcome to this tutorial in which we will be showing the main outputs of the EMLab–COMPETES tool.

As mentioned in previous videos, EMLab’s outputs are the CO2 market clearing price and the capacity market-clearing price

The COMPETES model calculates the shown main outputs for the EU as a whole and for the individual EU countries and aggregated regions.

This model outputs are written in an excel file that can be found in the results folder of COMPETES.

Here we show one example output file, which consists of different sheets where you can find various sorts of information.

These sheets show the investments in generation capacities, conventional and renewable. These are an output of the capacity expansion module. For this particular example, there were some required investments in different countries.

The following sheet shows the investments in cross-border transmission, where you can see the required capacity between different countries.

This sheet shows the hourly allocation or dispatch of installed power generation and interconnection capacities. Here you can see the different technologies and their hourly output.

Hourly nodal prices are calculated per country or region, as shown in this sheet.

The tool provides many other outputs, such as CO2 emissions per sector decommissioning and generation mix. These can be found in the following sheets.

We encourage the users to go through the output files where they can find more results.

Annex 5 – MASCEM narration scripts

Part I – Overview

Welcome to our first tutorial video about the Multi-Agent Simulator of Competitive Electricity Markets – MASCEM – a simulation and modeling tool developed to study electricity markets operation.

It models the main market entities as software agents and their interactions, such as market and system operators, buyers, sellers, and aggregators.

The Market Operator regulates pool negotiations by validating and analyzing the players' bids depending on the negotiation type, determines the market price, the accepted and refused bids, and market clearing.

The System Operator examines the technical feasibility from the power system point of view and solves congestion problems that may arise, being responsible for the system's security and ensuring that all conditions are met.

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 and are seen from the market's point of view as buyers or sellers.

MASCEM includes day-ahead and intraday markets (symmetric or asymmetric, with or without complex conditions), and bilateral contracts.

By selecting a combination of these market models, it is also possible to perform hybrid simulations.

MASCEM accommodates the simulation of three of the main European electricity markets: MIBEL, EPEX, and Nord Pool.

Part II – Input

Welcome to part II of our tutorial videos.

This video will introduce and explain the input models required to run the MASCEM spine toolbox project in the scope of TradeRES.

To run this scenario, the user must consider three types of input data, i.e., the input data for the MIBEL day-ahead market; the input data for the validation of the power flow of the electric grid; and the mapping between MIBEL players and their busses in the power network.

The "MIBEL input" corresponds to the request body that will be sent to the Electricity Markets Service to simulate a MIBEL Day-ahead market session.

Looking at its content, it is a JSON file containing all the information necessary to run a MIBEL day-ahead or intraday session.

It starts by identifying the name of the market, its type per period, and the list with the players' bids per period.

Each bid specifies the period number, the type of transaction (i.e., BUY vs SELL), and a list of offers.

Each offer details its energy amount and price per unit.

According to MIBEL, the maximum allowed number of offers per bid is 25.

However, MASCEM allows its users to test different alternatives by setting a different number of offers per bid, number of periods, or number of sessions per market type.

MIBEL also considers the use of Complex Conditions.

Their use depends on the type of market selected.

The day-ahead market only allows sellers to use Complex Conditions, while intraday markets allow their use by both buyers and sellers.

More details about “MIBEL inputs” are publicly available at <https://em.gecad.isep.ipp.pt>

After the market clearing, the market operator must validate the power flow feasibility with the system operator to ensure the quality and security of power transmission and distribution.

To this end, it requires the electricity market session results, the representation of the power grid considered for simulation, and the mappings of each player with its bus.

The market session results are automatically mapped to the power flow service in the spine toolbox project.

Users will need to provide the power grid representation, considering the existing elements, such as loads, buses, lines, generators (if available), batteries (if available), to name a few;

This example demonstrates the definition of the grid’s busses, lines, and external grid, and the type of algorithm to use for the power flow validation.

The user may also define the output format (i.e., JSON vs Excel file).

Further details about the Power Flow input data are publicly available at <https://pf.gecad.isep.ipp.pt>

The Players Buses input maps the players participating in the MIBEL day-ahead session with the respective buses defined in the power network.

These mappings ease the correct application of the players’ market outcomes in their busses for the power flow validation.

Thank you!

Part III – Output

Welcome to our third tutorial video.

This video will briefly explain the outputs of running the MASCEM spine toolbox project in the scope of TradeRES.

The outputs of this scenario correspond to the results that both the Electricity Market Service and the Power Flow Service produce and are available by pressing the “Results...” button, where all the results for each execution will be present.

Looking at the most recent execution of the MIBEL’s day-ahead market, one can see the accumulated session results as well as each player’s period results.

For each period, it is detailed the player's demand/supply, the satisfied or traded energy amount, and the period's market price.

Each player's amount of traded energy is used to validate the electrical grid's power flow for each period.

To this end, mappings between each player and its respective bus are provided as input, as explained in the second part of our tutorial videos.

The power flow analysis results, in turn, consist of the calculation of several metrics for each network element, and their interpretation, considering the electricity market outcomes.

In this particular case, the power flow analysis returns invalid because the voltage values of some buses and loading percentages of some lines, are outside the allowed limits.

Thank you!

Part IV – Run demo model

Welcome to our fourth and last tutorial video.

This video demonstrates a running example of the MIBEL day-ahead market, considering an example power network for the power flow validation.

To run MIBEL demonstrative scenario, users must provide the necessary inputs, i.e., the input data for the MIBEL day-ahead market, the input data for the validation of the power flow of the electric grid, and the mapping between MIBEL players and their busses in the power network.

Further details about these inputs are provided in our second tutorial video.

If the users decide to use input files located outside the input folder, they must update the location parameters related to each file.

Errors may occur if any of the inputs are invalid against the respective validation schemas, or if the connection to the Electricity Market Service or to the Power Flow Service fails.

The system will indicate to the user the type of error that occurred.

The user must also be aware of the necessary python dependencies to run the MASCEM spine toolbox example.

These are the packages: "requests", "json", and "jsonschema".

Finally, the example is run by pushing the "Execute..." button and, at the end of the simulation, the results are made available to the user.

Thank you!

Annex 6 – REStTrade narration scripts

Part I – Overview

Welcome to our first tutorial video regarding REStTrade – the Multi-agent Trading of Renewable Energy Sources modelling tool. This tool was developed to simulate traditional and new designs of the balancing markets and the imbalance settlement.

Balancing markets are important to ensure and maintain the system frequency at predefined limits and active power exchanges within scheduled values.

The current version of REStTrade modules can simulate the capacity procurement of the secondary reserve based on the ENTSO-E proposals, or on the Portuguese formulation. It is also possible to simulate the traditional energy procurement of the tertiary reserve. The clearing of these markets considers marginal pricing, but users can also choose pay-as-bid, as used in some European countries. The imbalance settlement can be computed based on the Portuguese or Spanish rules.

Currently, this tool can also simulate new market designs for the secondary reserve energy market and ancillary systems. These designs consider the participation of new players as variable renewable sources technologies and demand. Other features such as rolling gate closures closer to real-time operation, shorter market time units, and shorter products were conceived and considered for all balancing markets.

Furthermore, also the separate procurement of upward and downward capacity has been considered. Details of the designs implemented can be seen in deliverable 4.5 publicly available on TradeRES project website.

To run these modules users should collect input data, such as the programmed and real-time power dispatch of market players, used to compute the real-time power balance needs, required by REStTrade modules. Users should check the user guide and the input data tutorial about the required data and may use other tools provided by the TradeRES project to obtain such data.

Please see the following references to better understand how to deal with the models of this tool.

Thank you for listening.

Part II – Input

Welcome to the REStTrade Tutorial part II. This part focuses on the required input data to run the REStTrade tool.

Please check the user guide to see a detailed description of the required data.

The input data can be obtained using other tools of the TradeRES project. Alternatively, users can manually edit the files needed.

Supply and demand bids to the secondary capacity market must be collected and written in the file Secondary.

Each sheet of this file corresponds to a time period. The first row of each sheet contains the agent's identification, while the proposed capacity is indicated in the second row, and the price per unit of capacity in the third row.

Traditional markets do not consider a separate procurement between upward and downward capacity. Therefore, users may only put the upwards capacity and the program will automatically compute the capacity band that they bid, being the price computed for the entire band.

The new market designs developed in TradeRES project consider that transmission system operators have a separate procurement of upward and downward capacity as it is presented in this illustrative file. Under these circumstances, users may write the upward and downward bidding capacities they want to bid, such as their respective prices in the Secondary file.

Some traditional markets do not consider a secondary energy market being the price defined by the Regulator. In the REStTrade tool, the secondary and tertiary energy markets can simulate the separate procurement of upward and down energy. In this case, users have to fill the Excel files *SecondaryMarket.xlsx* and *Tertiary.xlsx* with the agents' upward and downward bids of price and energy in the secondary and tertiary energy markets, respectively.

The transmission system operator agents compute the procurement of secondary capacity considering the ENTSO-E proposals or the Portuguese formulation. This traditional procurement of secondary capacity considers the maximum expected demand per time period, which have to be written in the Excel file *SecondaryNeeds.xlsx*.

Users have to use other tools to compute the balancing markets' needs and the agents' deviations. The secondary and tertiary up and down energy needs have to be written in Excel files *SecondaryENeeds.xlsx* and *TertiaryNeeds.xlsx*, respectively. Furthermore, the total up and down imbalances also have to be written in the file *TertiaryNeeds.xlsx*, to compute the imbalance settlement.

Using the file *Config.txt*, users can set up some configurations of the simulation, it is a row-based file. In the first row, users may indicate if they want to simulate new or traditional market designs. Traditional market designs do not include the secondary energy market, so they have to write the regulated price of energy to be used in the imbalance settlement. In the second row, the user may select if the markets clearing considers marginal pricing or pay-as-bid schemes. In the third-row users may select between using the procurement of secondary capacity based on the ENTSO-E proposal or the Portuguese formulation. Finally, in the fourth-row, users may indicate if they want to use the imbalance settlement according to the double-price Portuguese or the single-price Spanish formulation.

Thank you for your attention.

Part III – Output

Welcome to the REStTrade tutorial part III. This tutorial focuses on the output data.

You should check the user guide for a detailed description of the output data. The output data depend on the settings of the configuration file - *Config.txt*.

To obtain these data you have to clone and run the MASCEM-REStTrade project in the spine toolbox, available on Github. Check the next tutorial for a detailed description of how can be obtained the data.

All output files are written in the root folder of the MASCEM-RETrade project. To avoid overwriting previous results, after each run, you need to copy the output files to a different folder between simulations.

The output of the secondary capacity market is present in the file `Secondaryoutput.xlsx`. If users set the simulation of traditional markets, this file presents, per period, the up and down capacity needs, the up and down contracted capacity, and the price. Otherwise, in the case of using a new market design, it presents the up and down capacity prices instead of a single price, as illustrated in the present file.

With this tool, you will be able to simulate new market designs of the secondary energy market. The output of the secondary energy market is present in the excel file `Secondaryoutput.xlsx`. This file presents the up and down needs for secondary regulation and their prices. Otherwise, if you simulate the traditional markets, only a regulated price is considered for the entire period under simulation and the file is not created.

The output of the tertiary energy market and the imbalance settlement are presented in the file `Tertiaryoutput.xlsx`. This file presents the up and down needs for tertiary regulation and their prices. It also presents the energy used for secondary regulation. All costs with energy regulation are used to compute the imbalance prices obtained in the imbalance settlement considering the Portuguese or the Spanish formulations, as illustrated in the present file.

Please check deliverable 4.5 on the TradeRES website for a better understanding of the market algorithms used to obtain the presented output results. Check the user guide for a detailed description of the output variables and files. In the next tutorial, tutorial number four, a comprehensive explanation of how to run the MASCEM-RETrade project is presented.

Thank you for your attention.

Part IV – Run demo model

Welcome to the RETrade Tutorial part IV. This part focuses on how to run the tool, for a simplified initiation of RETrade usage.

To use this coupled version, we recommend you to check MASCEM and RETrade tutorials and user guides on the TradeRES website. They explain to you all the models and how to collect input data to run this coupled tool.

First, you should get familiar with GitHub. We recommend you to see its documentation and learn how to use it. To run RETrade tool you need more than one GitHub project, so, we recommend you install and see the documentation of GitHub Desktop. It is a software where you can easily clone and manage all your GitHub projects.

With this software, you get informed of real-time changes in your projects and automatically upgrade the projects in your personal folders.

Furthermore, to run the RETrade tool you also have to get familiar with Spine Toolbox, a public tool used to integrate different models. Spine Toolbox was developed in Spine Horizon 2020 European project. Using GitHub is possible to clone Spine Toolbox. By reading its user-guides, users can understand how to install and run it manually or on GitHub desktop.

After you also need to clone MASCEM-RESTRade project that is being under development in the TradeRES project. This project is private, so you need to ask for access.

The first four blocks of the projects belong to MASCEM.

The “SecondaryCap” block simulates the capacity secondary market. The “Secondary” block simulates the secondary energy market.

“Ter&IS” block simulates the tertiary energy market and the imbalance settlement.

All RESTRade input and output files are in the root folder of the coupled project. So, users may copy their input and output files to save results, otherwise, they will be overwritten.

To run the coupled model, you can click on the first play symbol of Spine Toolbox, and MASCEM results will overwrite some of the input data. Running MASCEM is not enough to obtain all required input data, so, please check other tools as Backbone also available through TradeRES project. If you want to use your own data, select the RESTRade module that you want to simulate and click on the second play button.

Thank you for your interest in the RESTRade tool.