Input-Output analysis and modelling with MARIO

Hands-on 9 – Adding new sectors

Please, be aware that all the supporting materials required for this hands-on session is available on Zenodo at the following link: https://doi.org/10.5281/zenodo.8308515

Please use the following citation for:

- **MARIO Software**

Mohammad Amin Tahavori, Lorenzo Rinaldi, & Nicolò Golinucci. (2022). SESAM-Polimi/MARIO: MARIO v0.1.0 (v0.1.0). Zenodo. https://doi.org/10.5281/zenodo.5879382

Learning outcomes

By the end of this exercise, you will learn how to:

1) Add a commodity into a SUT
2) Add an activity into a SUT

Important requirement

Please make sure you have Microsoft Excel (or an equivalent alternative) installed on your PC.

Step 0: the Zenodo repository

All the supporting files for this and other Hands-ons and Lectures are available in the Zenodo repository associated to this course.

You find the repository at the following link: XXX

Adding activities into SUTs

In this Hands-on, we are going to add a new activity and a new commodity into a SUT Database, to complement the IOT case. The activity and commodity we are going add will be, in a similar fashion to Lecture 9, **the 'Manufacturing of Li-ion batteries' activity and the 'Li-ion batteries' commodity.**

First of all, we need to start by parsing a database, in particular we will parse the database that we exported at the end of Hands-on 8.

To parse the table we are using, again the parse_from_txt method, specifying the path, the type of table and the mode. **Notice that the path is targeting a 'flows' folder: this is because the export ('to_txt') method, creates automatically a 'flows' and/or 'coefficients' folders within the path you provide to it, since the table is split into multiple .txt files.**

```
#%% parse .txt table used in Hands-on9
"""
N.B: You can decide where to save your database
For the sake of keeping the material organized properly in the Zenodo repository,
we decided to save all the databases in the 'Databases' folder.
To do so, a small workaround is needed to get to a non-nested path and
contemporarily make it work indipendently of the user

N.B. #2: This will work if you downloaded the whole .zip folder contained
in the Zenodo repository and you did not changed the location of the files within it

N.B. #3: Set your current working directory to the script path first

"""
import os
path = "\\".join(os.getcwd().split("\\")[:-1])+"\\Databases\\Hands-on8\\flows"

world = mario.parse_from_txt(path=path, table='SUT', mode='flows') # use function to
```

N.B. For the sake of keeping the material organized properly in the Zenodo repository and to make the procedure work independently on the user (provided that the files locations within the folder have not been modified), we used a small workaround to direct to the 'Databases\Hands-on8' folder. However, **you are free to change the path of your files as you prefer.** Actually, to get acquainted with working with files in Python, we suggest you to practice in this sense as much as possible.

As we showed in Lecture 9, MARIO provides the user the possibility to fill an **Excel template file** to add sectors. However, this table we are using is a **SUT**, therefore the procedure will be slightly different.

Let's begin by getting the MARIO template for adding the activity, by using the get_add_sectors_excel method. The only difference with respect to the IOT case, when we added a **sector**, is that you need to provide not only the *new_sectors*, the *regions* where to add the activity and the *path,* but also the *item*, specified as 'Activity'

```
world.get_add_sectors_excel(
    path='Hands-on9 - Add activities.xlsx',
    new_sectors=['Manufacturing of Li-ion batteries'],
    regions=['EU27'],
    item='Activity', # you need to specify it for a SUT!
    )
```

This will save an empty Excel file in the current working directory named as "Hands-on9 - Add activities.xlsx". In the Zenodo repository, in the same folder, you find also the "Filled" counterpart file, which is already filled. Open this latter.

You will notice that the only sheet filled are:

- **input_from.** As explained in Lecture9, here you can specify which are the commodities *from which* the new activity gets its *inputs*. You may notice the inputs are exactly the same as for the case of the IOT in Lecture9.
- **Factor of production**, which complements the **input_from** characterization.
- **Satellite account**, to finalize the activity's interaction with environment (CO2 emissions in this case).
- **units**. Here we specify the output of the new activity is measured in EUR, like the rest of the activities' outputs (please refer to Lecture8 and Hands-on8 to check how to navigate the units of a table in MARIO)

We don't specify any final demand since final demand is to be specified for commodities, and we don't specify any commodity output in the **output_from** sheet. In fact the output of this activity would be the "Li-ion batteries" commodity **which has not yet been added!**

Finally, once the template has been filled properly, it is enough to call the 'add_sectors' method, providing again the path[1] to the template, the new_sectors, the item and the regions.

```
#%% Add activity to the SUT
world.add_sectors(
    'Hands-on9 - Add activities - Filled.xlsx',
    new_sectors=['Manufacturing of Li-ion batteries'],
    regions=['EU27'],
    item='Activity', # you need to specify it for a SUT!
    )
```

Adding commodities into SUTs

The next step is to add the commodity. The process is very similar to the addition of the activity: (i) call the get_add_sectors_excel method, (ii) fill the Excel template and (iii) read the Excel filled template back through the add_sectors method.

(i) **Call the add_sectors_excel_method**: you need to pass the same arguments as before, by changing them according to the commodity instead of the activity

```
# Continue by getting the template for commodities

world.get_add_sectors_excel(
    path='Hands-on9 - Add commodities.xlsx',
    new_sectors=['Li-ion batteries'],
    regions=['EU27'],
    item='Commodity', # you need to specify it for a SUT!
    )
```

(ii) **Fill the Excel template**: check into the already fille Excel file you find in the Zenodo repository ('Hands-on9 - Add commodities - Filled.xlsx'). You may notice that

- Again we specified EUR as the **units** of the commodity flow.

[1] The argument of the function is not called *path* but *io*, since the same information provided into the Excel template can be passed through a Pandas Dataframe. The same is for the *aggregate* method if you notice.

- In the **output_from** sheet we specified that the "Li-ion batteries" commodity comes as an *output from* the "Manufacturing of Li-ion batteries" previously added. We put 1, to indicate that the 100% of the commodity is produced by such activity. However, in case of different activities producing the same commodity, it would be possible to define a **market share.** Below here, an example of **import share**, where European "Li-ion batteries" are 70% imported from China and 30% domestically produced.
-

A	B	C	D
			EU27
			Commodity
Region	Category	Activity	Li-ion batteries
EU27	Activity	Manufacturing of Li-ion batteries	0.3
China	Activity	Manufacturing of Li-ion batteries	0.7

- We added a 150 EUR of **final demand** of Li-ion batteries in EU27 (approximately the cost of 1 kWh). The value to put here is totally arbitrary and depends on the type of analysis you want to perform.

In any case, it is **of utmost importance to define at least a certain consumption** of the new commodity by final consumers (**final demand** sheet) or other activities (**input_to** sheet), otherwise, once applying the Leontief Production Model, having characterized just the supply side **technical coefficients** without defining any demand side, would lead to a **null production** of the new implemented activities/commodities.

In practical terms, it would be like the new activities and commodities are empty rows and columns.

(iii) **Read the Excel filled template back through the add_sectors method**

```
world.add_sectors(
    'Hands-on9 - Add commodities - Filled.xlsx',
    new_sectors=['Li-ion batteries'],
    regions=['EU27'],
    item='Commodity', # you need to specify it for a SUT!
    )
```

We can check the total production of Li-ion batteries, and we expect it to be equal to the total consumption of Li-ion batteries, **which in our case is just final demand (150 EUR)**

```
In [45]: world.X.loc[(slice(None),'Commodity','Li-ion batteries'),:]
Out[45]:
                                                    production
Region               Level     Item
Africa               Commodity Li-ion batteries         0.0
Australia            Commodity Li-ion batteries         0.0
China                Commodity Li-ion batteries         0.0
EU27                 Commodity Li-ion batteries       150.0
India                Commodity Li-ion batteries         0.0
Middle East          Commodity Li-ion batteries         0.0
Rest of America      Commodity Li-ion batteries         0.0
Rest of Asia&Pacific Commodity Li-ion batteries         0.0
Rest of Europe       Commodity Li-ion batteries         0.0
Russia               Commodity Li-ion batteries         0.0
USA                  Commodity Li-ion batteries         0.0
```

Before closing the hands-on, just export again the database, similar to how we did in Hands-on 8, this time let's use the 'to_excel' parser **(in case it takes more than 10-20 minutes, you can definitely come back at using to 'to_txt' parser!)**.

```
#%% Export database
import os
export_path = "\\".join(os.getcwd().split("\\")[:-1])+"\\Databases\\Hands-on9.xlsx"

world.to_excel(export_path)
```