Journal of open research software

MARIO: A Versatile and User-Friendly Software for Building Input-Output Models

SOFTWARE METAPAPER

MOHAMMAD AMIN TAHAVORI
NICOLÒ GOLINUCCI
LORENZO RINALDI
MATTEO VINCENZO ROCCO
EMANUELA COLOMBO

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

MARIO (Multi-Regional Analysis of Regions through Input-Output) is a Python-based framework for building input-output models. It automates the parsing of well-known databases (e.g. EXIOBASE, EORA, Eurostat) and of customized tables. With respect to similar tools, like pymrio, it broadens the scope of application to supply-use tables and handles both monetary and physical units. Employing an intuitive Excel-based API, it facilitates advanced table manipulations and allows for modelling additional supply chains through a hybrid LCA approach. It provides built-in functions for footprinting and scenario analyses as well as for visualizations of model outcomes. Results are exportable into various formats, possibly supplemented by a metadata file tracking the full history of applied changes. MARIO comes with extensive documentation and is available on Zenodo, GitHub, or installable via PyPI.

CORRESPONDING AUTHOR:
Mohammad Amin Tahavori
MARIO lead developer and co-designer; eNextGen, Via Principe Eugenio 9, 20155, Milan, Italy;
VITO NV – Hoofdkantoor, Boeretang 200, 2400, Mol, Belgium
amin.tahavori@enextgen.it

KEYWORDS:
Scenario analysis; Footprints; MARIO; Input-output analysis; Supply and use; hybrid-LCA; Python

TO CITE THIS ARTICLE:
Tahavori MA, Golinucci N, Rinaldi L, Rocco MV, Colombo E 2023 MARIO: A Versatile and User-Friendly Software for Building Input-Output Models. *Journal of Open Research Software*, 11: 14. DOI: https://doi.org/10.5334/jors.473

(1) OVERVIEW

INTRODUCTION

Input-Output (IO) analysis plays a crucial role in unravelling the intricate interconnections between diverse economic sectors [1]. By offering a numerical representation of the flows of goods and services in an economy, these models enable analysts to evaluate the effects of alterations in demand or supply on a range of economic, social, and environmental factors [2]. The growing emphasis on sustainability in today's world makes these models even more indispensable, as policymakers and stakeholders in every sector grapple with the challenges of the ongoing sustainability transition. In this context, IO models offer a valuable tool for understanding the impacts of economic activities on sustainability and identifying opportunities for enhancing it. However, despite the benefits of IO modelling, practical impediments exist.

Despite the importance of IO models in assessing the impacts of changes in demand or supply on economic, social, and environmental aspects, the use of these models can be challenging due to their complexity and the mathematical operations involved. Moreover, the available software tools for IO modelling are often limited in flexibility and scalability, making building and modifying IO models time-consuming and difficult. To achieve the most accurate life-cycle assessment (LCA), researchers can adopt process-based databases, IO databases, or a hybrid approach combining the two. However, managing the increasingly large databases required for these approaches is a practical issue that must be addressed. As database detail continues to expand, the need for efficient and scalable software tools that can handle the complexity of IO models becomes more pressing.

Open Science and Open-Source Software in Industrial Ecology

Open Science is a movement toward more accessible, transparent, and collaborative research practices that can benefit the scientific community and society as a whole [3]. By sharing knowledge and data openly and promoting collaboration among researchers and other stakeholders, Open Science can accelerate the pace of scientific discovery, improve the quality of research, and promote more inclusive and equitable access to scientific information [4]. Transparency and accessibility of models and their associated data are crucial for producing higher-quality science, increasing productivity, and improving the science-policy boundary in the energy research community [5].

Widespread use of open-access software can, at the same time, increase the quality, transparency, and reproducibility of Industrial Ecology (IE) research [6]. The growing data and computation intensity in IE research has led to a greater need for the development of scientific software, but this has also created challenges related to transparency, reproducibility, reusability, and collaboration. The authors of [6] propose a fourfold response to this problem. Firstly, the authors propose the implementation of existing general principles for the development of good scientific software in IE and related fields. Secondly, they argue that collaborating on open-source software could make IE research more productive and increase its quality and provide guidelines for the development and distribution of such software. Thirdly, the authors call for stricter requirements regarding general access to the source code used to produce research results and scientific claims published in the IE literature. Finally, a set of open-source IE modelling tasks, which they hope will turn their recommendations into practice.

As a result of these efforts, a Python toolbox for industrial ecology has been introduced, which includes a range of frameworks and modules such as *Brightway2* [7], *ecospold2matrix* [8], *pySUT* [9], *pymrio* [10], and others. Recently, other Python packages such as *pycirk* [11, 12], and *pyLCAIO* [13] have also been developed to enable modeling of Circular Economy scenarios and streamline the hybridization of process-based Life Cycle Assessment and Environmentally Extended IO databases. Recently, *unfold* was released, a repository that improves the packaging and sharing of data, making it easier to reproduce life-cycle databases that are based on proprietary data sources [14].

This paper introduces MARIO, a Python-based IO modelling framework that builds upon and improves many of the features of the previously cited pieces of software. MARIO provides a scale- and database-agnostic approach to IO modelling, allowing the user to adopt any kind of IO table, ranging from a single- to multi-region, monetary- or physical-units, and symmetric (or IOTs) or supply-use (SUTs). MARIO's key features include automatic parsing of various well-known available databases, an intuitive Application Program Interface (API) for applying advanced modifications to an existing database ranging from aggregation, creation of scenarios for policies and processes economic and environmental impact assessment, and extensions to new supply chains by adopting a hybrid-LCA approach. Finally, by standardizing the mathematical complexities of IO models and being designed both for Python newcomers and expert users, MARIO represents to modelers an efficient tool to avoid recursive and time-consuming operations while focusing on their analyses.

IMPLEMENTATION AND ARCHITECTURE

Underlying mathematics and nomenclature

A comprehensive reference text to deepen IO analysis theory and application scopes is [1]. However, for the sake of clarity, it is necessary to synthetically provide an overview of the main concepts as well as to define the nomenclature adopted in MARIO and along the paper itself.

As mentioned, IO tables represent an insightful instrument to inspect the economic transactions that occurred during one year. Economic accounts are stored in various matrices, specifically:

- Matrix **Z** includes the transactions of goods and services occurring from one industrial activity to another. In the case of a supply-use table (SUT), activities are distinct from the commodities they produce, therefore two sub-matrices can be identified within **Z**, namely a supply matrix **S**, representing the production of each commodity by the supplying activities, and a use matrix **U**, characterizing the inputs, in terms of commodities, required by each activity.
- Matrix **Y** shows the consumption of goods and services by those generally denominated as consumption categories, such as households and government. Also, information about changes in valuables and investments is reported in **Y**.
- Matrix **V** complements **Z** by reporting the information on the value added provided by each industrial activity to its output in addition to the costs sustained due to its input consumption (i.e. capital, labor, taxes, and subsidies).

Multiregional tables introduce information on trades by specifying the geographic origin of each economic account. Moreover, IO tables are also often extended with environmental transactions, tracking the interactions of industrial activities (matrix **E**) and consumption categories (matrix **EY**) with different environmental dimensions such as emissions, primary energy, water, and land use.

Given any IO table, it is possible to calculate the total production vector **X**, which reports the total production of each industrial activity (and of each commodity, in the case of a SUT), as described by Eq. 1 (**i**, in this case, is a vector with dimension equal to the rows number of **Z**). It is possible to express any of the above-mentioned matrices (except for **Y**) per unit of total production, calculating the so-called technical coefficients matrices by dividing the desired matrix by the inverse of **X**. An example is provided by calculating the intersectoral transactions coefficients matrix **z** in Eq. 2. Please note that the "hat" symbol indicates a diagonalized vector. The same equation, applied to matrix **E**, leads to the derivation of matrix **e** which can be identified as direct environmental impact coefficients matrix, or production-based accounting impact matrix.

Following Eq. 2, Eq. 1 can be re-arranged to derive the Leontief production model (Eq. 3), expressing the total production of each activity (and/or commodity) represented in the IO table that is required to fulfil given final demand yields; **I**, in this case, represents an identity matrix and **w** is conventionally known as Leontief Inverse matrix.

$$\mathbf{X} = \mathbf{Z} \cdot \mathbf{i} + \mathbf{Y} \qquad \textit{Eq. 1}$$

$$\mathbf{z} = \mathbf{Z} \cdot \hat{\mathbf{X}}^{-1} \qquad \textit{Eq. 2}$$

$$\mathbf{X} = \mathbf{z} \cdot \mathbf{X} + \mathbf{Y} = (\mathbf{I} - \mathbf{z})^{-1} \cdot \mathbf{Y} = \mathbf{w} \cdot \mathbf{Y} \qquad \textit{Eq. 3}$$

IO tables can also be used for consumption-based impact accounting (or footprinting analysis), by adopting the so-called Leontief impact model (Eq. 4). Unlike **e**, matrix **f**, called specific footprint matrix, allocates to each industrial activity not only its direct but also its indirect environmental impact.

$$\mathbf{f} = \mathbf{e} \cdot \mathbf{w} \qquad \textit{Eq. 4}$$

Furthermore, IO tables are often adopted to perform scenario analyses, generally implemented by modelling a perturbation (or shock) of the economic system and by evaluating the difference between the new and the original system. Eq. 5 describes how the matrices of a new shocked economic system $t + 1$ can be seen as the sum of the corresponding matrix of the original system t and a matrix represented only the shock to be implemented, indicated as $\Delta(\mathbf{z},\mathbf{v},\mathbf{e},\mathbf{Y})$.

$$(\mathbf{z},\mathbf{v},\mathbf{e},\mathbf{Y})_{t+1} = (\mathbf{z},\mathbf{v},\mathbf{e},\mathbf{Y})_{t} + \Delta(\mathbf{z},\mathbf{v},\mathbf{e},\mathbf{Y}) \qquad \textit{Eq. 5}$$

Code Structure

Before deepening the code structure, a preliminary glossary of the main terms and items is provided in Table 1 and adopted henceforth in the paper.

ITEM	DEFINITION
Table	Any input-output or supply-use table that can be parsed in MARIO.
Matrix	Components of a table. In MARIO, each table is structured in a set of matrices, represented as *Pandas MultiIndexed DataFrames* [15]
Sets	Basic information characterizing each matrix. The combination of the appropriate sets defines the *MultiIndex* of the *DataFrame* representing the matrix. For instance, the final demand matrix of an IOT is characterized by *regions* and *sectors* on the rows, and by *regions* and *consumption categories* on the columns.
Scenario	Any shocked version of the same table. In MARIO, by default, the first scenario is always called *baseline*.
Instance	A MARIO Database object, characterized by all the matrices of all the scenarios.

Table 1 Glossary.

The core functionalities of MARIO are implemented through an Object-Oriented programming paradigm, specifically by means of two classes. These classes are named *CoreIO* and *Database*.

The CoreIO class encompasses general methods and functionalities for any IO model, including tasks like calculating desired matrices and checking whether a table is balanced. On the other hand, the Database class contains additional methods and functionalities specifically tailored for scenario analyses, such as shock analysis.

In MARIO, an IO table is structured as a set of matrices, represented by *MultiIndex Pandas DataFrames* [15]. This allows for a unified representation of various IO tables by utilizing different levels of indexing. Regarding MARIO scenario analysis, all the matrices for a particular scenario are gathered into a nested Python dictionary after they have been calculated. In practical terms, through this dictionary, it is possible to compare multiple shocked versions of the same table with the *baseline* one.

Each instance is additionally associated with another object called *MARIOMetaData*. This object serves as metadata and tracks all the changes made to the instance, as well as the analyses performed by the user. The metadata keeps a record of the instance's history, which can be printed out as a JSON or TXT file, upon user request. An essential representation of MARIO's structure is sketched in Figure 1.

Tables parsing

One of the main challenges associated with using IO tables is the lack of a standardized approach for storing and organizing the data. Each specific table follows its structure, both in terms of data storage and labelling conventions for elements within the database. For example, Eora [16] stores all the matrices of a single-region table in a unique TXT file, the monetary version of EXIOBASE 3 [17] stores each matrix in a distinct TXT file, while the hybrid version [18] adopts CSV files and a significantly different manner to store environmental transactions.

This challenge becomes even more complex when working with SUTs or hybrid tables that incorporate additional dimensions of information into the IO framework. Additionally, non-structured tables, such as those developed by specific groups or for particular projects, add further inconsistencies. The lack of standardized structure among different existing tables hinders the reproducibility of IO modelling exercises, as modellers constantly need to adapt to different paradigms for reading and handling various types of tables.

To overcome this issue, MARIO adopts its specific structure. Regardless of the type or origin of the data, any table read by MARIO is transformed into a standardized format according to which each matrix is characterized with specific indexing, as reported in Table 2.

MARIO incorporates a set of parsing functions tailored to differently structured tables:

- Eora, namely Eora26, single-region IOTs and single-region SUTs;
- EXIOBASE 3, including both IOTs and SUTs as well as monetary and hybrid versions;
- Eurostat SUTs.

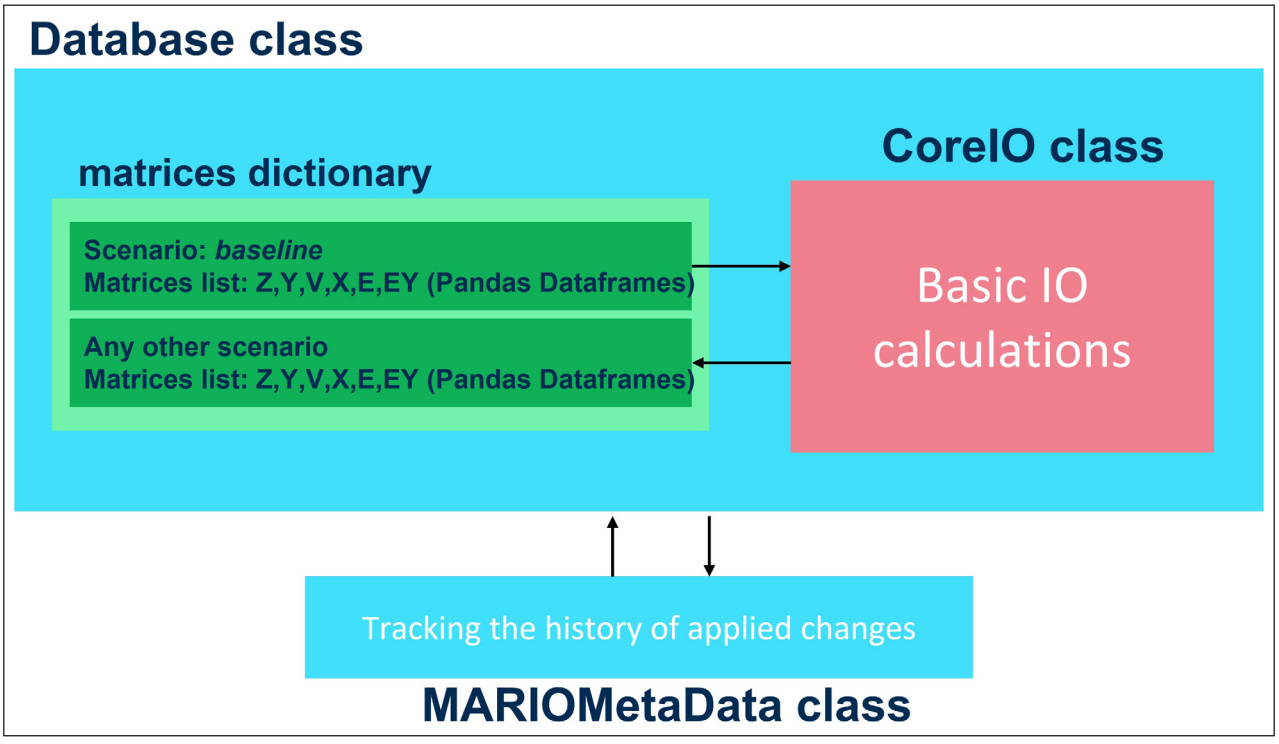


Figure 1 MARIO core classes and data properties.

MATRICES	ROW INDEX LEVELS	COLUMN INDEX LEVELS	DESCRIPTION
Z (**z**)	Region, Item(s), Label	Region, Item(s), Label	The intermediate transactions matrices always represent supplying and consuming regions and items. Items can be alternatively "Sectors" in case of symmetric input-output tables (IOTs) or "Commodities" and "Activities" in case of supply-use tables (SUTs).
U (**u**)	Region, Commodity, Label	Region, Activity, Label	The use transactions matrices always represent the supplied commodities and consuming activities by region of production and consumption. Use matrices are calculated just in the case of SUTs.
S (**S**)	Region, Activity, Label	Region, Commodity, Label	The supply transactions matrices always represent supplied commodities by supplying activities by region of production and consumption. Supply matrices are calculated just in the case of SUTs.
Y	Region, Item(s), Label	Region, Consumption category, Label	The final transactions matrix always represents the consumption of items in different regions and by different consumption categories. Items can be alternatively "Sectors" in case of symmetric input-output tables (IOTs) or "Commodities" and "Activities" in case of supply-use tables (SUTs).
V (**v**)	Factors of production labels	Region, Item(s), Label	The value added matrices always represent the consumption of factors of production in different regions and by different items. Items can be alternatively "Sectors" in case of symmetric input-output tables (IOTs) or "Commodities" and "Activities" in case of supply-use tables (SUTs).
E (**e**)	Satellite accounts labels	Region, Item(s), Label	The intermediate environmental transaction matrices always represent the consumption of satellite accounts in different regions and by different items. Items can be alternatively "Sectors" in case of symmetric input-output tables (IOTs) or "Commodities" and "Activities" in case of supply-use tables (SUTs).
EY	Satellite accounts labels	Region, Consumption category, Label	The final environmental transaction matrix always represents the consumption of satellite accounts in different regions and by different consumption categories.

Table 2 Matrices indexing logic. Such logic is applied to any parsed table independently of the original formatting.

When dealing with non-structured tables, MARIO is capable of reading datasets from Excel (XLSX), CSV, or TXT files provided that they have been firstly structured in a pre-defined shape. Furthermore, with the release of v.0.2.0, it is also possible to import data from a pymrio object [10] (IOSystem class) and transform it into a MARIO Database class. In version 0.2.2, MARIO can also exploit the pymrio table downloading functions. More about the tables available for download at pymrio documentation: https://pymrio.readthedocs.io/en/latest/notebooks/autodownload.html.

Calculating matrices

The CoreIO class includes a function called *calc_all*, which enables users to calculate a specified list of matrices for a given scenario within the model. If the calculation of a particular matrix relies on another matrix that is not yet present in the scenario, MARIO will determine the appropriate order of calculations for different matrices to generate the requested matrix.

In addition to the basic matrix calculations, MARIO offers a query function that allows users to perform more advanced automatic calculations across different scenarios. This includes calculations such as determining the absolute or relative changes of a matrix among different scenarios.

Furthermore, all the mathematical calculations related to IO analysis are programmed as separate functions. This enables users to utilize these functions outside of the main structure of a MARIO Database or CoreIO class, providing greater flexibility in their calculations.

Modifying an instance

The Database class in MARIO offers several capabilities to alter an existing instance, including the following functionalities:

- Aggregating (or just renaming) the instance into different dimensions of data by reducing the number of regions, sectors, and so on.
- Adding new sectors in case of an IOT, or new commodities and activities in case of a SUT.
- Adding or dropping satellite accounts to the existing instance.

Excel-based interaction

To enhance the user experience for MARIO users, several functionalities have been designed to be accessible through automatically generated Excel files which serve as an interface. These interfaces allow modellers to provide the minimum required inputs to make their changes. Furthermore, the users are supported by list validation, to minimise input errors and typos.

For instance, to aggregate an instance, users can utilize the *get_aggregation_excel* function, which generates an Excel file with pre-filled indices related

to the instance sets such as the list of regions, sectors, satellite accounts... This Excel file enables easily defining the desired aggregation and then it is enough to read back the file to execute the aggregation process. Similarly, tasks such as adding a new sector, or satellite account, or implementing a shock can be accomplished using automatically generated Excel files within MARIO.

Surely, however, every task which is simplified through Excel interfaces can be also performed directly in Python, in case of more advanced necessities.

Example application

In this section, we show a simple yet comprehensive tutorial on the main novel features of the MARIO modelling framework. Check the "Software location: Emulator environment" section below to get access to the Jupiter Notebook related to this example, along with all the supporting Excel files.

For further information and tutorials, we developed an online video-course available at Polimi Open Knowledge platform [19]. You can also visit the Zenodo repository dedicated to our "Input-Output analysis and modelling with MARIO" online course [20].

The following example requires the user to first download the EXIOBASE hybrid supply-use table referred to the year 2011 [18]. The downloaded folder can be stored in any directory. The path to this folder is indicated henceforth with *exio_path*.

To start parsing the downloaded table, it is necessary to have MARIO installed (link to the installation in the documentation [21]), to import it, and to call the save a MARIO Database object called *world* by using the *parse_exiobase* function, indicating the type of table and the type of units. Also, it is possible to filter over the desired environmental extensions. Any of the sheets stored in the "MR_HSUTs_2011_v3_3_18_extensions.xlsx" provided in [18], files can be provided, as a list. In this case, we are going to use only the *Emiss* extensions, referring to the emissions transactions.

```
import mario
world = mario.parse_exiobase(path=exio_
path, table='SUT', unit="Hybrid",
extensions=['Emiss'])
```

To investigate the parsed table, it is often useful to get the list of labels of a desired set or to extract the list of labels containing a determined string within a set. These two features are allowed by using the *get_index* and the *search* functions respectively. Examples of getting a list of all the labels included in the "Activity" set and of getting a list of all the activities including the "gas" string among the same set are reported as follows:

```
world.get_index('Activity')
world.search('Activity','gas')
```

It is also possible to calculate new matrices, indirectly using the above-mentioned *calc_all* function. To get a new, or an already calculated matrix in the baseline scenario, it is enough to use the matrix property as shown for the case of the footprint coefficient matrix **f** (already described in Eq. 4).

```
world.f
```

The first advanced table modification we perform in this tutorial is aggregation. It is possible to export an empty Excel template by calling the *get_aggregation_excel* function and by providing a path *aggr_path*.

```
world.get_aggregation_excel(aggr_path)
```

The Excel file must be filled in at least one of its sheets. Each sheet presents the list of labels of the sets it is dedicated to and the user is required to map the original labels with new desired labels to reduce the numerosity of the desired sets.

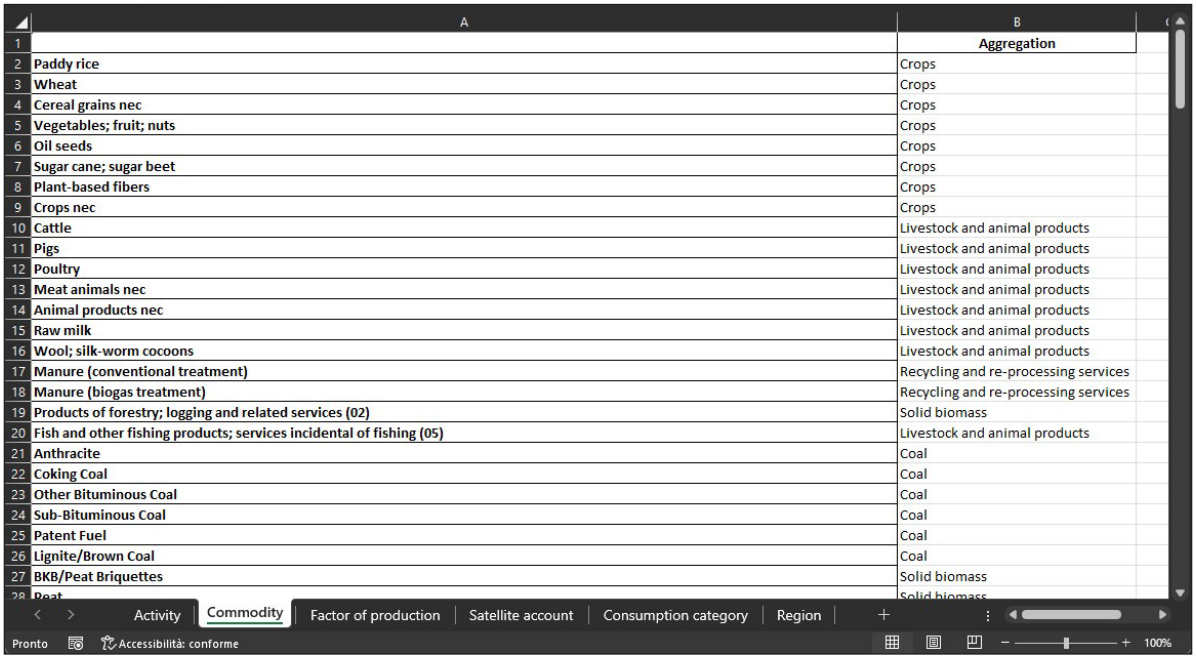
Figure 2 shows the aggregation of the commodities of the raw EXIOBASE table into a new configuration.

It is worth noting that in the case of a hybrid-units table, like the one in this example, the user must pay attention to the homogeneity of the units relative to the commodities that are going to be aggregated. A specific error is raised in case this rule is neglected.

To aggregate the Database, it is just required to read back the filled Excel file by using the *aggregate* function. It is finally possible to check the new dimension of the Database by printing it. The aggregated database has now 5 regions, 50 activities, 44 commodities, and just 1 satellite account, instead of the original 48 regions, 164 activities, 200 commodities, and 66 satellite accounts.

```
world.aggregate(aggr_path)
print(world)
```

The second modification to the table would be the extension of the newly obtained aggregated table to a new supply chain. In this example, we will display how to model a new industrial activity, namely the European supply chain of battery manufacturing. A new activity called "Manufacture of batteries" as well as a new commodity called "Batteries" need to be added to the table. Also in this case, the user can be supported by an integration with dedicated Excel files. In the case of a SUT, two files are requested, one to add the "Batteries" commodity and a second to add the new "Manufacture of batteries" activity. It is first necessary to create a list of *new_activities* including the label of the new activity to be modelled, then the user shall provide a path *add_activity_path* where the template Excel file would be exported. The *get_add_sectors_excel* function can finally be executed, by specifying also that the item which will



	A	B
1		Aggregation
2	Paddy rice	Crops
3	Wheat	Crops
4	Cereal grains nec	Crops
5	Vegetables; fruit; nuts	Crops
6	Oil seeds	Crops
7	Sugar cane; sugar beet	Crops
8	Plant-based fibers	Crops
9	Crops nec	Crops
10	Cattle	Livestock and animal products
11	Pigs	Livestock and animal products
12	Poultry	Livestock and animal products
13	Meat animals nec	Livestock and animal products
14	Animal products nec	Livestock and animal products
15	Raw milk	Livestock and animal products
16	Wool; silk-worm cocoons	Livestock and animal products
17	Manure (conventional treatment)	Recycling and re-processing services
18	Manure (biogas treatment)	Recycling and re-processing services
19	Products of forestry; logging and related services (02)	Solid biomass
20	Fish and other fishing products; services incidental of fishing (05)	Livestock and animal products
21	Anthracite	Coal
22	Coking Coal	Coal
23	Other Bituminous Coal	Coal
24	Sub-Bituminous Coal	Coal
25	Patent Fuel	Coal
26	Lignite/Brown Coal	Coal
27	BKB/Peat Briquettes	Solid biomass
28	Peat	Solid biomass

Figure 2 Aggregation Excel file. Example of commodities aggregation.

be modelled in this template is an activity and in which regions this activity will be implemented.

```
new_activities = ['Manufacture of
batteries']
world.get_add_sectors_excel(
      new_sectors=new_activities,
      regions=world.get_index("Region"),
      item='Activity',
      path=add_activity_path,
      )
```

To properly model a new activity, it is necessary to provide a characterization of the input structure of the new activity in the "input_from" sheet (Figure 3), the unit of measure of its output ("units" sheet), and possibly the related satellite transactions ("Satellite account" sheet). The characterization of the input structure is intended to be performed in the form of a life-cycle inventory [22] and therefore needs to be supported by appropriate data collection as well as by a harmonization process to correctly map the collected literature data with the sets of the adopted table.

Once the template to add the activity is filled, the *add_sectors* function shall be called to add the new activity. It is possible to notice that the Database now has 51 activities by printing the Database object.

```
world.add_sectors(
      new_sectors=new_activities,
      regions=world.get_index("Region"),
      item='Activity',
      io=add_activity_path
      )
print(world)
```

The same procedure needs to be followed for the addition of the new commodity: it is required, therefore, to provide a list of *new_commodities* including "Batteries", and to generate another Excel template (*add_commodity*_path).

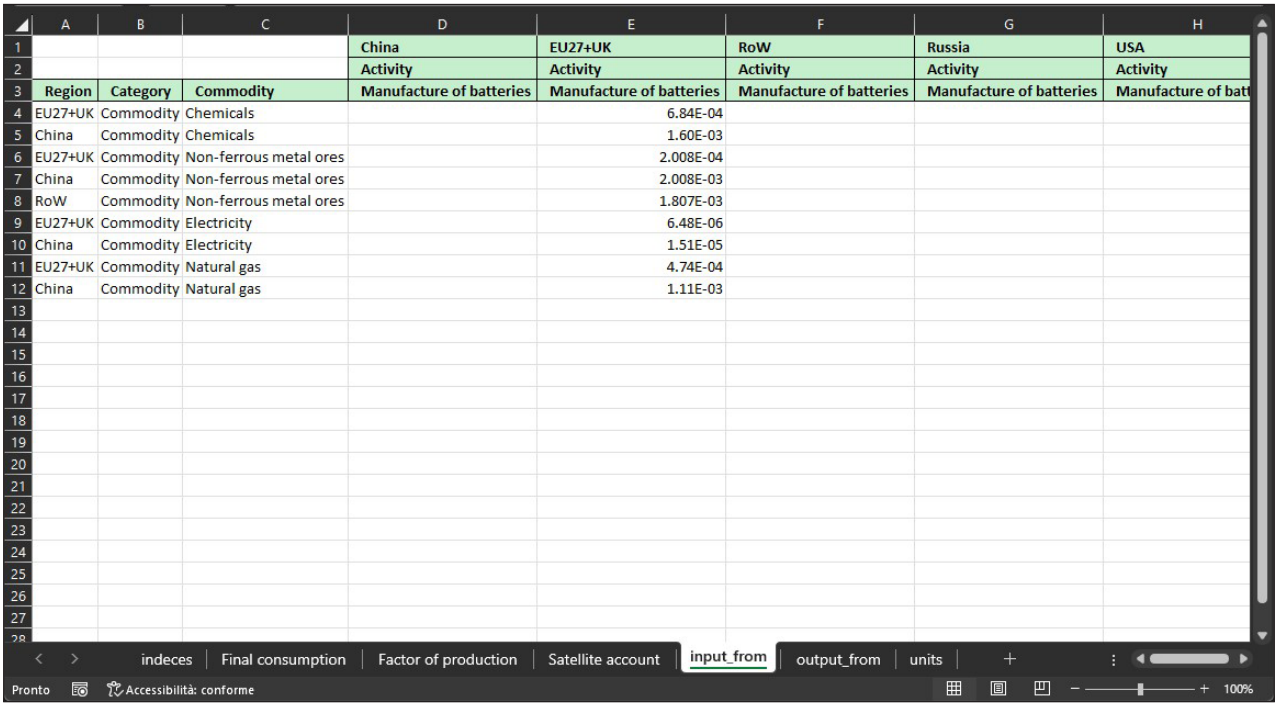
```
new_commodities = ['Batteries']
world.get_add_sectors_excel(
      new_sectors=new_commodities,
      regions=world.get_index("Region"),
      item='Activity',
      path=add_commodity_path,
      )
```

When filling the file, in this case, it is necessary to indicate that the "Batteries" commodity is produced by the new activity in the "output from" sheet (Figure 4) and that the only consumption of batteries is in the European final demand ("Final consumption" sheet). The "Batteries" commodity is measured in "kWh" ("units" sheet).

Once the template Excel to add the commodities is filled it is necessary to call the *add_sectors* function once more. Afterward, it is possible to notice that the Database now has 45 commodities by printing the Database object.

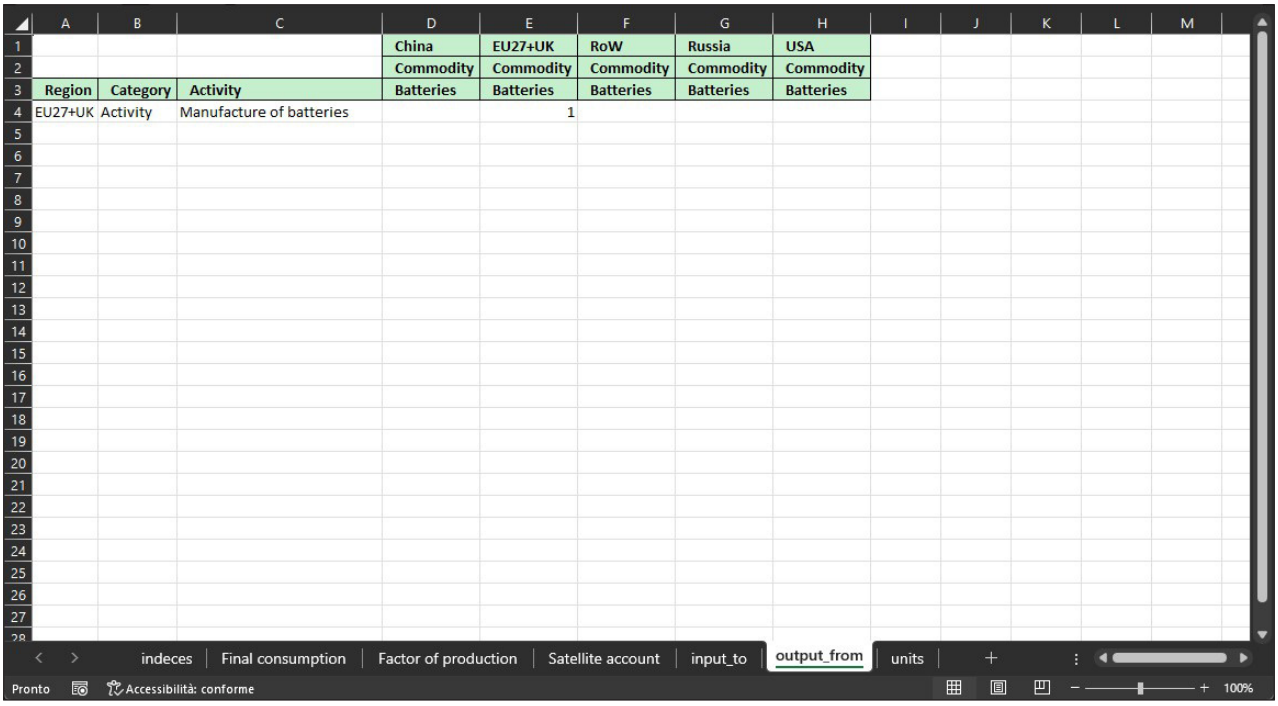
```
world.add_sectors(
      new_sectors=new_commodities,
      regions=world.get_index("Region"),
      item='Commodity,
      io=add_commodity_path
      )
print(world)
```

The tutorial analysis is finalized by implementing a scenario analysis, therefore a shock would be modelled into the table. MARIO allows exporting an empty Excel



	A	B	C	D	E	F	G	H
1				China	EU27+UK	RoW	Russia	USA
2				Activity	Activity	Activity	Activity	Activity
3	Region	Category	Commodity	Manufacture of batteries	Manufacture of batteries	Manufacture of batteries	Manufacture of batteries	Manufacture of batt
4	EU27+UK	Commodity	Chemicals		6.84E-04			
5	China	Commodity	Chemicals		1.60E-03			
6	EU27+UK	Commodity	Non-ferrous metal ores		2.008E-04			
7	China	Commodity	Non-ferrous metal ores		2.008E-03			
8	RoW	Commodity	Non-ferrous metal ores		1.807E-03			
9	EU27+UK	Commodity	Electricity		6.48E-06			
10	China	Commodity	Electricity		1.51E-05			
11	EU27+UK	Commodity	Natural gas		4.74E-04			
12	China	Commodity	Natural gas		1.11E-03			

Figure 3 Characterization of the input structure of the new activity.



	A	B	C	D	E	F	G	H
1				China	EU27+UK	RoW	Russia	USA
2				Commodity	Commodity	Commodity	Commodity	Commodity
3	Region	Category	Activity	Batteries	Batteries	Batteries	Batteries	Batteries
4	EU27+UK	Activity	Manufacture of batteries		1			

Figure 4 Characterization of the supply of the European "Batteries" commodity by the new European "Manufacture of batteries" activity.

template to be filled with the information regarding the desired modifications to be applied to the original table. It is enough to call the *get_shock_excel* function and provide a path (*shock_path*). In this example, the environmental impact of the increase in European final consumption of domestic batteries up to 1 TWh is evaluated.

```
world.get_shock_excel(shock_path)
```

To implement such shock, the user would need to navigate to the **Y** sheet and provide the instructions to update the final consumption of the European "Batteries" commodity in Europe (Figure 5).

Once the shock template is filled, the *shock_calc* function implements the shock in the Database object. It is necessary to provide which matrices are intended to be modified (just **Y** for this particular shock) and the name of the new scenario which will be created ("increased batteries demand" in this case). It is possible to print the Database object to check that two scenarios are present in the instance after this operation.

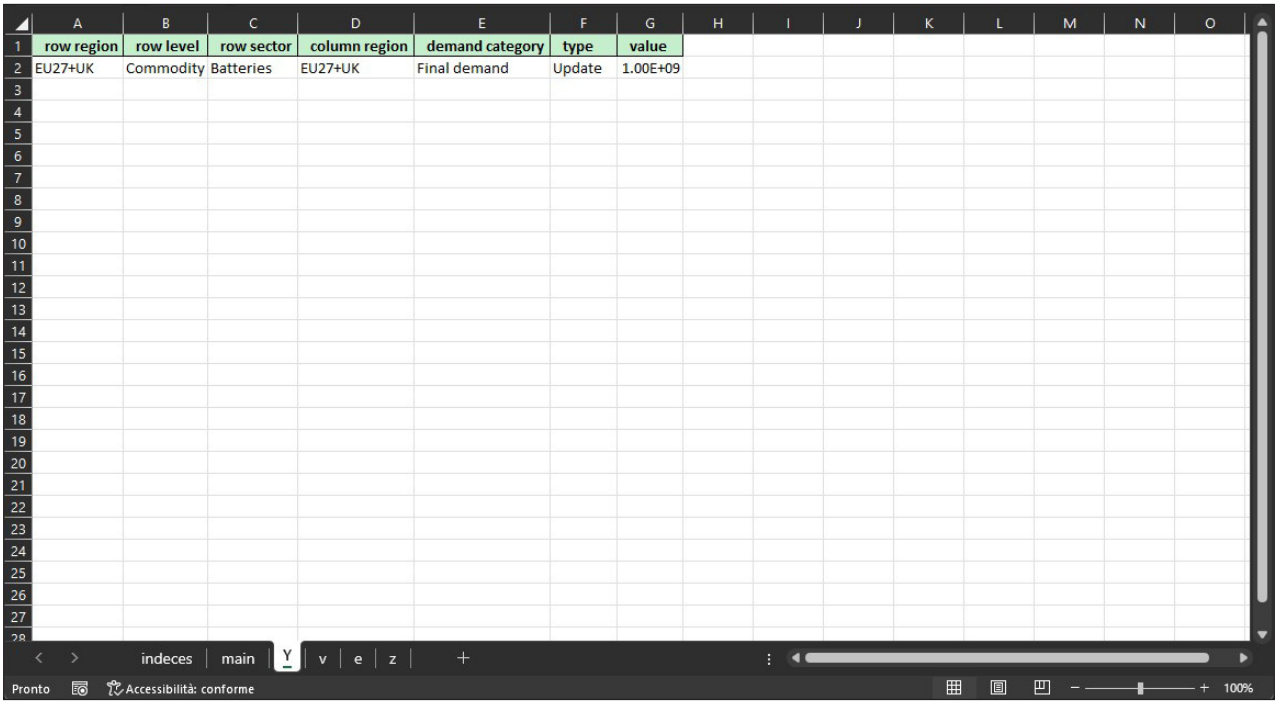


Figure 5 Shock implementation on the Y matrix.

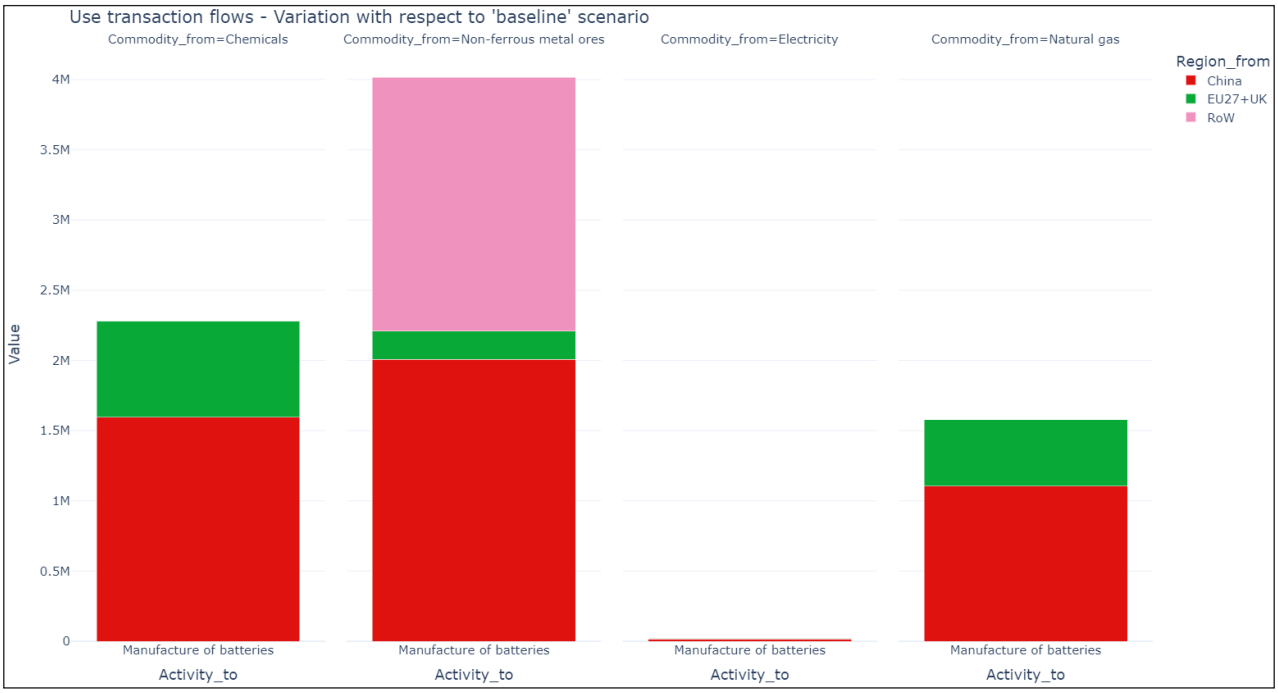


Figure 6 Built-in visualization of selected results. It is worth noting that the values of each commodity are represented following its unit of measure.

```
world.shock_calc(
      io=shock_path,
      Y=True,
      scenario='increased batteries demand,
      )
print(world)
```

To conclude, it is possible to visualize the results, for example, the impact caused by the implemented shock on the consumption of commodities required for the production of batteries. Such impact can be provided with respect to the baseline scenario. To do so, we can use the *plot_matrix* function, which is based on the Plotly library [23], and apply a set of desired filters.

```
world.plot_matrix(
      matrix = 'U',
      item = 'Commodity',
      facet_col='Commodity_from',
      x = 'Activity_to',
```

```
color='Region_from',
path= plot_path,
base_scenario = "baseline",
filter_Activity_to= ['Manufacture of
batteries'],
filter_Region_
from=['China','EU27+UK','RoW'],
filter_Region_to=['EU27+UK'],
filter_Commodity_
from=['Chemicals','Non-ferrous metal
ores','Electricity', 'Natural gas'],
)
```

The resulting plot is an interactive HTML file, as shown in Figure 6, which reports the consumption of the new "Manufacture of batteries" activity of selected commodities (one per subplot, each expressed in its specific unit of measure) coming from different regions (legend colors). The values are expressed as a variation with respect to the baseline scenario.

QUALITY CONTROL

The current version of MARIO has achieved a test coverage of 49%. This coverage includes a comprehensive 100% assessment of the fundamental mathematical engine. Additional tests are currently in active development.

For the unit test process, *pytest* is used (https://docs.pytest.org/en/7.4.x/). Moreover, there are two sample cases embedded in the software which allows the user to test the main functionalities of MARIO for two types of tables that can be handled, namely SUTs and IOTs. The two examples can be loaded using the *load_example* function:

```
from mario import load_example
IOT = load_example("IOT")
SUT = load_example("SUT")
```

The MARIO source code follows the PEP 8 specifications as implemented by the Black code formatting library (https://github.com/psf/black).

The MARIO documentation is built using the Sphinx Python Documentation Generator and hosted on *readthedocs* (https://mario-suite.readthedocs.io/). It includes all the API references, information on the requirements, and the installation guide, a full section is dedicated to the terminology, and two others are devoted to database parsing and examples.

(2) AVAILABILITY

OPERATING SYSTEM

Windows, Mac OS, GNU/Linux, and any other operating systems running Python.

PROGRAMMING LANGUAGE

Python 3.7 or higher.

ADDITIONAL SYSTEM REQUIREMENTS

No specific requirements. Please note that memory usage is strongly affected by the dimension of the parsed input-output table.

DEPENDENCIES

For the current MARIO version 0.2.0

- pandas 1.3.5
- numpy 1.21.6
- plotly 5.14.1
- tabulate 0.9.0

SOFTWARE LOCATION:

Archive

Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.5879382
Licence: GPL v3
Publisher: Mohammad Amin Tahavori
Version published: 0.2.2 and earlier versions.
Date published: 16/09/23 (version 0.2.2)

Code repository

Name: Github (also hosted on PyPI)
Identifier: https://github.com/it-is-me-mario/MARIO
Licence: GPL v3
Date published: 16/09/23 (version 0.2.2)

Emulation environment

Name: Jupiter Notebook of the displayed example application
Identifier: https://github.com/it-is-me-mario/Tutorials/blob/main/JORS%20paper/Tutorial.ipynb
Licence: GPL v2
Date published: 22/05/23

LANGUAGE

English

(3) REUSE POTENTIAL

MARIO's target users are manyfold.

It was primarily designed for academic and other IO professionals in need of a user-friendly, flexible, and all-around comprehensive tool to perform economic and environmental analyses. The software has been currently

adopted in various peer-reviewed scientific publications [24–27] as well as in Horizon [28] and private consultancy projects. In particular, the scope of the applications may range from policy impact assessment to commodities as well as industrial processes footprint evaluation, or to compare environmental or economic repercussions caused by different technological alternatives. As stated in the Introduction section, we believe MARIO's real added value with respect to other available tools is that it collects many features of most of such tools in one unique package, such as parsing SUTs, parsing hybrid-units tables, generating scenarios, and allowing to extend IO tables by implementing new supply-chain/processes. It is finally the intention of the authors to propose MARIO as a bridge among other tools, leaving the users to decide to adopt their preferred tools for some specific features while allowing them to expand their analyses into MARIO in case some other features are absent in their preferred tool. For instance, from version 0.2.0, MARIO allows parsing tables that have been firstly manipulated with pymrio directly importing a pymrio IOSystem object.

The other noticeable added value provided by MARIO is related to its Excel-based API, thanks to which MARIO is easily exploitable for teaching and training activities, in particular, if such activities aim to develop new capacities in the field of Industrial Ecology and/or Life-Cycle Assessment. MARIO has been successfully used in different courses ranging from the ICTP Joint Summer School on Modelling Tools for Sustainable Development [29], to a MOOC course available on the Open Knowledge platform of Politecnico di Milano [30] as well as for a Master of Science degree course [31] and in various other professionals-oriented courses provided by the Department of Energy in Politecnico di Milano.

Regarding communications, updates, and contributions, the preferred channel is the GitHub organization (link available in the Software location section): developers and contributors are invited to adopt the GitHub issues tracking system to report bugs and propose enhancements to the code. Direct emails to the authors of this paper are also welcome in case of any other request.

ACKNOWLEDGEMENTS

The authors want to thank Prof. Maria Cristina Rulli, who unintentionally inspired MARIO's name. A special thanks to all the thesis students and MSc students from the Advanced Thermodynamics and Thermoeconomics and Energy Accounting and Impact Assessment Modelling courses of Politecnico di Milano, who in the last three years used MARIO since the very early stages and helped us reporting bugs, feedbacks and ideas.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Mohammad Amin Tahavori orcid.org/0000-0002-7753-0523
MARIO lead developer and co-designer; eNextGen, Via Principe Eugenio 9, 20155, Milan, Italy; VITO NV – Hoofdkantoor, Boeretang 200, 2400, Mol, Belgium
Nicolò Golinucci orcid.org/0000-0002-8735-499X
MARIO co-developer and lead designer; Department of Energy, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
Lorenzo Rinaldi orcid.org/0000-0003-4667-8653
MARIO co-developer and co-designer; Department of Energy, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
Matteo Vincenzo Rocco orcid.org/0000-0003-3129-3654
Scientific coordination and supervision; Department of Energy, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
Emanuela Colombo orcid.org/0000-0002-9747-5699
Scientific coordination and supervision; Department of Energy, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy

REFERENCES

1. **Miller RE, Blair PD.** "Input-Output Analysis – Foundations and Extensions." Cambridge University Press; 2012. DOI: https://doi.org/10.1017/CBO9780511626982
2. **Duchin F.** "Structural economics : measuring change in technology, lifestyles, and the environment." Washington, DC: Island Press; 1998.
3. **Vicente-Saez R, Martinez-Fuentes C.** "Open Science now: A systematic literature review for an integrated definition." *J Bus Res*. 2018; 88: 428–436. DOI: https://doi.org/10.1016/j.jbusres.2017.12.043
4. **Pfenninger S,** et al. "Opening the black box of energy modelling: Strategies and lessons learned." *Energy Strategy Reviews*. 2018; 19. 63–71. DOI: https://doi.org/10.1016/j.esr.2017.12.002
5. **Pfenninger S, DeCarolis J, Hirth L, Quoilin S, Staffell I.** "The importance of open data and software: Is energy research lagging behind?" *Energy Policy*. 2017; 101: 211–215. DOI: https://doi.org/10.1016/j.enpol.2016.11.046
6. **Pauliuk S, Majeau-Bettez G, Mutel CL, Steubing B, Stadler K.** "Lifting Industrial Ecology Modeling to a New Level of Quality and Transparency: A Call for More Transparent Publications and a Collaborative Open Source Software Framework." *J Ind Ecol*. 2015; 19(6): 937–949. DOI: https://doi.org/10.1111/jiec.12316
7. **Mutel C.** "Brightway: An open source framework for Life Cycle Assessment." *The Journal of Open Source Software*. 2017; 2(12): 236. DOI: https://doi.org/10.21105/joss.00236

8. **Majeau-Bettez G, Agez M.** "ecospold2matrix." https://github.com/majeau-bettez/ecospold2matrix (accessed May 14, 2023).
9. **Majeau-Bettez G, Pauliuk S, Mutel C, Stadler K.** "pySUT."
10. **Stadler K.** "Pymrio – A Python Based Multi-Regional Input-Output Analysis Toolbox." *J Open Res Softw*. 2021; 9(1): 8. DOI: https://doi.org/10.5334/jors.251
11. **Donati F, Aguilar-Hernandez GA, Sigüenza-Sánchez CP, de Koning A, Rodrigues JFD, Tukker A.** "Modeling the circular economy in environmentally extended input-output tables: Methods, software and case study." *Resour Conserv Recycl*. 2020; 152; 104508. DOI: https://doi.org/10.1016/j.resconrec.2019.104508
12. **Donati F,** et al. "Modeling the circular economy in environmentally extended input–output: A web application." *J Ind Ecol*. 2020; 25(1): 36–50. DOI: https://doi.org/10.1111/jiec.13046
13. **Agez M, Wood R, Margni M, Strømman AH, Samson R, Majeau-Bettez G.** "Hybridization of complete PLCA and MRIO databases for a comprehensive product system coverage." *J Ind Ecol*. 2020; 24(4): 774–790. DOI: https://doi.org/10.1111/jiec.12979
14. **Sacchi R.** "unfold: removing the barriers to sharing and reproducing prospective life-cycle assessment databases." *J Open Source Softw*. 2023; 8(83): 5198. DOI: https://doi.org/10.21105/joss.05198
15. "Pandas Official Webpage."
16. **Lenzen M, Moran D, Kanemoto K, Geschke A.** "BUILDING EORA: A GLOBAL MULTI-REGION INPUT–OUTPUT DATABASE AT HIGH COUNTRY AND SECTOR RESOLUTION." *Economic Systems Research*. 2013; 25(1): 20–49. DOI: https://doi.org/10.1080/09535314.2013.769938
17. **Stadler K,** et al. "EXIOBASE 3." Oct. 2021. DOI: https://doi.org/10.5281/ZENODO.5589597
18. **Merciai S, Schmidt J.** "EXIOBASE HYBRID v3 – 2011." May 2021. DOI: https://doi.org/10.5281/ZENODO.7244919
19. **Rocco MV, Golinucci N, Rinaldi L, Tonini F.** "Measuring the impact of the Energy transition." *POLIMI Open Knowledge*; 2023. https://www.pok.polimi.it/courses/course-v1:Polimi+MET102+2023_M7/about (accessed Sep. 17, 2023).
20. **Rinaldi L, Golinucci N, Rocco MV, Colombo E.** "Input-Output analysis and modelling with MARIO." *Zenodo*; Sep. 2023. DOI: https://doi.org/10.5281/zenodo.8308515
21. **Tahavori MA, Golinucci N, Rinaldi L.** "MARIO – Documentation."
22. **International Standard Organization.** "ISO 14040: Environmental Management-Life Cycle Assessment-Principles and Framework." 1997.
23. **Plotly.** "Plotly Python Graphing Library." 2023. https://plotly.com/python/ (accessed May 23, 2023).
24. **Conte M,** et al. "Investigating the economic and environmental impacts of a technological shift towards hydrogen-based solutions for steel manufacture in high-renewable electricity mix scenarios for Italy." in *IOP Conference Series: Earth and Environmental Science*; 2022. DOI: https://doi.org/10.1088/1755-1315/1106/1/012008
25. **Falchetta G, Golinucci N, Rocco MV.** "Environmental and energy implications of meat consumption pathways in sub-saharan africa." *Sustainability (Switzerland)*. 2021; 13(13). DOI: https://doi.org/10.3390/su13137075
26. **Golinucci N,** et al. "Comprehensive and Integrated Impact Assessment Framework for Development Policies Evaluation: Definition and Application to Kenyan Coffee Sector." *Energies (Basel)*. 2022; 15(9). DOI: https://doi.org/10.3390/en15093071
27. **Rinaldi L, Rocco MV, Colombo E.** "Assessing critical materials demand in global energy transition scenarios based on the Dynamic Extraction and Recycling Input-Output framework (DYNERIO)." *Resour Conserv Recycl*. 2023; 191: 106900. DOI: https://doi.org/10.1016/j.resconrec.2023.106900
28. **Nikas A,** et al. "Three responses to the energy crisis – the co-benefits of energy efficiency." *IAM COMPACT*; 2023. [Online]. Available: https://www.iam-compact.eu/sites/default/files/2023-05/%5BIAM%20COMPACT%5D%20Three%20responses%20to%20the%20energy%20crisis.pdf.
29. **Climate Compatible Growth.** "Joint Summer School on Modelling Tools for Sustainable Development." *International Centre for Theoretical Physics*; 2023. https://indico.ictp.it/event/10186#:~:text=The%20Joint%20Summer%20School%20on,from%203rd%20July%20to%2014th (accessed May 22, 2023).
30. **Rocco MV, Golinucci N, Rinaldi L, Tonini F.** "MOOC course: Measuring the impact of the Energy Transition." *Polimi Open Knowledge*; 2023. https://www.pok.polimi.it/courses/course-v1:Polimi+MET102+2023_M1/about (accessed May 22, 2023).
31. **Rocco MV.** "MSc course: Energy Accounting and Impact Assessment Methods." *Manifesto degli Studi, Politecnico di Milano*; 2022. https://www11.ceda.polimi.it/schedaincarico/schedaincarico/controller/scheda_pubblica/SchedaPublic.do?&evn_default=evento&c_classe=809952&__pj0=0&__pj1=5efc1a54cde03c4b8a27a02dee9547b2 (accessed May 22, 2023).

TO CITE THIS ARTICLE:
Tahavori MA, Golinucci N, Rinaldi L, Rocco MV, Colombo E 2023 MARIO: A Versatile and User-Friendly Software for Building Input-Output Models. *Journal of Open Research Software,* 11: 14. DOI: https://doi.org/10.5334/jors.473

Submitted: 27 May 2023 **Accepted:** 11 October 2023 **Published:** 27 October 2023