The Journal of Open Source Software

flodym: A Python package for dynamic material flow analysis

Jakob Dürrwächter [1¶], **Merlin Hosak** [1], **Bennet Weiss** [1], **and Falko Ueckerdt** [1,2]

1 Potsdam Institute for Climate Impact Research, Energy Transition Lab, Potsdam, Germany **2** Interdisciplinary Transformation University, Energy Transition and Climate Futures, Linz, Austria ¶ Corresponding author

DOI: 10.21105/joss.10105

Software

- Review
- Repository
- Archive

Editor: Marlin Arnz

Reviewers:

- @michaelweinold
- @TimoDiepers

Submitted: 15 July 2025
Published: 30 March 2026

Summary

Material flow analysis (MFA) is a core method in industrial ecology that tracks material flows within a system, such as a national economy, across different life cycle stages (Brunner & Rechberger, 2016). It also accounts for material accumulation in stocks, including materials embodied in products, assets, and infrastructure at a given time. MFA supports resource management, environmental impact assessment, and the evaluation of circular economy strategies, informing policy-making, urban and regional planning, and sustainable product design.

flodym (Flexible Open Dynamic Material Systems Model) is a Python library of objects and functions needed to build dynamic MFA models. From a mathematical perspective, MFA largely involves operations on multi-dimensional arrays. flodym implements the `FlodymArray` class, which internally manages operations of one or more such arrays. Objects representing flows, stocks, and parameters inherit from this class. Stocks include lifetime models for dynamic stock modelling, i.e. for calculating the relationship between material inflows to a stock and the resulting mass and age structure of that stock over time. The whole MFA system is realized with an abstract parent class, which users can subclass. The parent class includes routines for checking mass balance consistency and non-negative flows. flodym offers functionality for efficient data import and export via pandas (McKinney, 2010; The pandas development team, 2020), as well as visualization routines.

flodym is based on the concepts of the Open Dynamic Material Systems Model (ODYM) (Pauliuk & Heeren, 2020). It is a re-implementation with expanded functionality and improved structuring. As a result, flodym enables users to write customized, flexible MFAs, designed for maintainability and future extension.

Statement of need

MFA provides quantitative insights into how materials are extracted, transformed, used, and accumulated in socioeconomic systems over time. In academia, MFA enables research on resource efficiency, environmental impacts, and sustainability transitions. In industry, it informs decisions on material sourcing, product lifetimes, and circular economy strategies. For policy-makers and planners, it provides an evidence base for resource governance, waste management, demand forecasting, and intervention design.

MFA is therefore widely used in industrial ecology and related fields. Several review papers document the development and applications of MFA, including Müller et al. (2014), Bringezu & Moriguchi (2018), Graedel (2019), and Streeck et al. (2023).

The broad use of MFA makes widely applicable and easily accessible MFA tools essential for a large audience in academia, industry, and policy-making. Compared to MFA software using a graphical user interface, a code library such as flodym allows customization to users' individual needs, and substantially expands the range of possible applications, which is vital in academia. One example is the coupling with other modeling tools like life cycle assessment or integrated assessment models, where custom implementations enable flexible data exchange through programmatic interfaces.

State of the field

While there are several existing open source MFA software packages, ODYM (Pauliuk & Heeren, 2020) is, to the knowledge of the authors, the only general and adaptable open-source MFA library, allowing users to write their own MFA with the full range of options that custom code offers. Therefore, this section first describes ODYM's capabilities and potential downsides in detail, followed by an overview of other tools.

ODYM

ODYM is widely used in the industrial ecology community and related research areas. One of its strengths is that it builds on an abstraction of the principles and structures of MFA:

- formalizing a system definition and establishing mass conservation checks
- formalizing dynamic stock models
- translating the abstract concepts of processes, stocks, flows, and parameters into a general library without prescribing a specific MFA structure (e.g., specific dimensions).

However, several aspects leave room for improvement:

- Multi-dimensional array operations are central to MFA implementations. ODYM stores dimensionality information in its array objects, but does not fully utilize it. Each dimension in the code (e.g., time) is assigned an index letter. ODYM stores which dimensions each array has. However, the user still has to manually repeat it for every array operation. For example: The array `end_of_life_products` has dimensions time (t), region (r) and product type (p). The array `waste_share` has dimensions product type (p) and waste type (w). Multiplying them should yield the array `waste` with dimensions time (t), region (r), and waste type (w). For the multiplication, the ODYM user must manually specify the dimension mapping `'trp,pw->trw'`, as in the following code:

  ```
  waste.Values = np.einsum(
      'trp,pw->trw',
      end_of_life_products.Values,
      waste_share.Values
  )
  ```

  This is cumbersome and prone to errors. Also, it reduces readability and makes the code less flexible, because any change in dimensionality requires modifying each operation.

- A similar issue arises with array slicing. Selecting the portion of the `waste` array corresponding to a single region might look like this:

  ```
  waste.Values[:,0,:]
  ```

  This is hard to interpret, and also subject to change if if the array's dimensional structure changes.

- The class for dynamic stock models does not support dimensions other than time. It also does not contain integrated methods for all required computation steps. Moreover, stock objects used in the MFA system represent only one of the three quantities – stock, inflow, or outflow – distinguished by a `Type` attribute. To transfer the results of the

dynamic stock model to the MFA, the user has to loop over all non-time dimensions, run several sub-methods of the scalar dynamic stock model, and transfer the results into the MFA arrays. This is cumbersome and potentially a performance bottleneck.

- Data read-in and initialization requires a strict input format based on Excel files, which limits flexibility. There is no data export functionality.
- ODYM features several comprehensive application examples, but only a partial API reference is available, and the API does not consistently follow PEP 8 naming conventions.

Other tools

Other MFA packages such as STAN (Cencic & Rechberger, 2008) or OMAT (Villalba & Hoekman, 2018) are different in scope: They are standalone software tools rather than programming libraries. This eases their use, but limits the flexibility for using them in non-standard ways, as flodym allows. A similar limitation applies to the pymfa (Thiébaud et al., 2019) and PMFA (Kawecki-Wenger, 2019) packages, which primarily focus on probabilistic MFA as a specific extension of MFA.

Software design

flodym is based on the concepts of ODYM, and therefore shares a similar structure, scope, and core strengths. However, flodym also addresses several limitations and introduces additional functionality:

- flodym uses dimensionality information for automatic internal management of array dimensions in operations on multidimensional arrays. The multiplication example from the previous section simplifies to

  ```
  waste[...] = end_of_life_products * waste_share
  ```

  using `FlodymArray` objects. This allows users to write simpler code and reduces errors. For example, in the `einsum` statement in the previous section, dimensions of the same size could accidentally be swapped, which would produce incorrect results without raising an error. Additionally, the flodym syntax makes the code flexible for adaptation and extension (hence the name flodym). Because dimensions are defined once during array initialization rather than specified in each operation, dimensions can later be added, removed, or reordered with minimal changes to the source code.

- Array slicing is simplified in a similar way. For example, the values of the `waste` array corresponding to the `CHA` (China) entry of the `region` dimension can be accessed using the syntax

  ```
  waste['CHA']
  ```

  This again allows additional dimensions to be added or removed later, or changing the ordering of the entries within a dimension, without having to change the code. Apart from these functionalities, which are built on Python's special methods (methods with names in double underscores allowing operator overloading), `FlodymArray` objects provide a wide range of built-in methods for dimension manipulation, such as `sum_over`, `cast_to` or `get_shares_over`.

- In flodym, the handling of material stocks is simplified and integrated with the rest of the MFA. This is implemented through `Stock` objects that contain `FlodymArray` instances for inflow, outflow and stock arrays, together with a lifetime model and associated computation methods. Both stocks and lifetime models are multidimensional and attributes of the MFA system class, enabling seamless interaction with flows and the performance gains of NumPy array operations. As novel functionality, stock models can handle unevenly spaced time step vectors, or sub-year lifetimes.

- In flodym, data import and export are based on pandas, allowing support for a wide range of formats. Users can either use pre-built flodym import functions, or write their own routines to generate objects from pandas DataFrames. On data read-in, flodym performs checks on the data, helping detect errors early. Data import is performance-optimized for sparse arrays: the full array size is only used after converting the input pandas DataFrame to a NumPy array. Data types are validated using pydantic (Colvin et al., 2025), adding robustness to the code.
- flodym provides functionality for data export and visualization.
- The whole flodym code follows established software engineering practices (such as PEP 8 formatting, or GitHub Actions for automated testing and documentation building) and clean code, facilitating future collaboration and extension. The code is extensively documented, including docstrings, type hints, an API reference, how-tos and examples.

The required modifications relative to ODYM were sufficiently extensive that refactoring the existing codebase – prior to implementing the new functionality – was estimated to require more effort than developing a new implementation. For this reason, a reimplementation approach was chosen.

Research impact statement

flodym has already been adopted in several research projects and teaching activities.

Finished large-scale projects using flodym include the in-house REMIND-MFA model (Dürrwächter et al., 2026) and the externally developed TRANSIENCE EU MFA model (Saurat et al., 2025).

flodym has been used for the following teaching events. None of these events were organized by the authors of the publication, demonstrating independent adoption of the library by the community:

- An autumn school on LCA–MFA coupling was largely based on flodym (Départ de Sentier, 2025), which resulted, for example, in the development of several GitHub repositories[1][2][3][4][5].
- An invited lecture on flodym at the Brightcon 2025 conference (Départ de Sentier, 2026).
- A university class at Leiden University attend by approximately 150 students.

Further dissemination through conferences and teaching events is planned. Based on the current uptake, the software has the potential to support a wide range of future research and teaching activities in the community.

AI usage disclosure

AI was used during generation of the flodym source code: The auto-complete functionality of GitHub Copilot was used in VS Code. The GitHub Copilot coding agent was used in minor development projects, resulting in 11 commits comprising 729 lines of code. Different AI tools were also used to write parts of the test suite.

AI-assisted language editing was used to improve wording in parts of the manuscript.

All code and text generated or modified by AI was proofread by the authors. AI-generated code was also extensively tested.

[1] https://github.com/Depart-de-Sentier/Schools-2025-November-Switzerland
[2] https://github.com/isabelapi/PAW_MFA_LCA_2025
[3] https://github.com/MeYiwen/DdS_REFLOC
[4] https://github.com/Tanima-Sharma/PVProject
[5] https://github.com/gergosuto/dds2025manure

Acknowledgements

We thank Stefan Pauliuk and the other contributors to ODYM (Pauliuk & Heeren, 2020), which forms the conceptual basis for flodym.

We thank Sally Dacie and other contributors to the project who are not listed as authors.

We gratefully acknowledge funding from the TRANSIENCE project, grant number 101137606, funded by the European Commission within the Horizon Europe Research and Innovation Programme; from the Kopernikus-Projekt Ariadne through the German Federal Ministry of Education and Research (grant no. 03SFK5A0-2); and from the PRISMA project funded by the European Commission within the Horizon Europe Research and Innovation Programme under grant agreement No. 101081604 (PRISMA).

References

Bringezu, S., & Moriguchi, Y. (2018). Material flow analysis. In P. Bartelmus & E. K. Seifert (Eds.), *Green Accounting* (pp. 149–166). Routledge. https://doi.org/10.4324/9781315197715-6

Brunner, P. H., & Rechberger, H. (2016). *Handbook of material flow analysis* (2nd ed.). CRC Press. https://doi.org/10.1201/9781315313450

Cencic, O., & Rechberger, H. (2008). Material flow analysis with software STAN. In A. Moeller, B. Page, & M. Schreiber (Eds.), *Environmental informatics and industrial ecology*. Shaker Verlag. ISBN: 978-3832273132

Colvin, S., Jolibois, E., Ramezani, H., Garcia Badaracco, A., Dorsey, T., Montague, D., Matveenko, S., Trylesinski, M., Runkle, S., Hewitt, D., Hall, A., & Plot, V. (2025). *Pydantic* (Version v2.11.7). https://docs.pydantic.dev/latest/

Départ de Sentier. (2025, July). *DdS Autumn School 2025*. https://www.d-d-s.ch/schools/nov-25

Départ de Sentier. (2026, January). *Brightcon 2025: Hackathon & courses in Grenoble and online*. https://indico.d-d-s.ch/event/1/page/13-courses-for-beginners-and-advanced-users-sold-out

Dürrwächter, J., Hosak, M., Weiß, B., Schweiger, L., Zhang, Q., & Ueckerdt, F. (2026). *REMIND-MFA* (Version v0.2.0). https://remind-mfa.readthedocs.io

Graedel, T. E. (2019). Material flow analysis from origin to evolution. *Environ. Sci. Technol.*, *53*(21), 12188–12196. https://doi.org/10.1021/acs.est.9b03413

Kawecki-Wenger, D. (2019). *PMFA: Base functions for performing a probabilistic MFA using R*. https://github.com/empa-tsl/PMFA

McKinney, Wes. (2010). Data structures for statistical computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Müller, E., Hilty, L. M., Widmer, R., Schluep, M., & Faulstich, M. (2014). Modeling metal stocks and flows: A review of dynamic material flow analysis methods. *Environ. Sci. Technol.*, *48*(4), 2102–2113. https://doi.org/10.1021/es403506a

Pauliuk, S., & Heeren, N. (2020). ODYM — An open software framework for studying dynamic material systems: Principles, implementation, and data structures. *J. Ind. Ecol.*, *24*(3), 446–458. https://doi.org/10.1111/jiec.12952

Saurat, M., Lotz, T. M., Bußmann, S., & Holtz, G. (2025). *TRANSIENCE-EU-MFA*. https://transience-eu-mfa.readthedocs.io

Streeck, J., Pauliuk, S., Wieland, H., & Wiedenhofer, D. (2023). A review of methods to trace material flows into final products in dynamic material flow analysis: From industry shipments in physical units to monetary input-output tables, part 1. *J. Ind. Ecol.*, *27*(2), 436–456. https://doi.org/10.1111/jiec.13380

The pandas development team. (2020). *Pandas-dev/pandas: pandas* (latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

Thiébaud, E., Alexandru, C., Badat, R., Kohler, D., & Hilty, L. (2019). *pymfa2: Ein Werkzeug zur Analyse von Materialflüssen in Python 2.7* (Version 2.1). https://bitbucket.org/Xeelk/pymfa2/src/master/

Villalba, G., & Hoekman, P. (2018). Using web-based technology to bring hands-on urban material flow analysis to the classroom. *J. Ind. Ecol.*, *22*(2), 434–442. https://doi.org/10.1111/jiec.12553