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]

1 Potsdam Institute for Climate Impact Research, Energy Transition Lab, Potsdam, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Marlin Arnz

Reviewers:

- @michaelweinold
- @TimoDiepers

Submitted: 15 July 2025
Published: unpublished

Summary

Material flow analysis (MFA) is a core method in industrial ecology that tracks time-dependent material flows within a system, such as a national economy, across all life cycle stages. 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 advice, urban and regional planning, and sustainable product design.

flodym (Flexible Open Dynamic Material Systems Model) is a library of objects and functions needed to build dynamic MFA. Mathematically, a large part of MFA translates to operations between multi-dimensional arrays. *flodym* implements the `FlodymArray` class, which internally manages operations of one or several such arrays. Objects representing flows, stocks, and parameters all inherit from this class. Stocks include lifetime models for dynamic stock modelling, i.e. for calculating the relation of material flows entering a stock and the mass and age structure of that stock over time. The whole MFA system is realized with an abstract parent class including sanity checks for the system, that users can implement a subclass of. *flodym* includes functionality for efficient read-in 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 can be seen as a re-implementation with vastly expanded functionality and improved structuring. As a result, *flodym* enables users to write customized, flexible MFAs, built for maintainability and future expansion.

Statement of need

MFA provides quantitative insights how materials are extracted, transformed, used, and retained within socio-economic systems over time. In academia, it 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, MFA provides an evidence base for resource governance, waste and demand forecasting, and the design of interventions.

MFA is therefore a widespread method in industrial ecology and related fields. Scopus lists 4260 publications including the exact term "Material flow analysis" in title or abstract. Review papers include, for example, (Müller et al., 2014), (Bringezu & Moriguchi, 2018), (Graedel, 2019), and (Streeck et al., 2023).

This makes general and easily accessible MFA tools vital for a large audience in academia, industry and policy-making. Compared to MFA software using a GUI, a code library such as *flodym* allows customizability to users' individual needs, which is vital in academia. It drastically enlarges the range of possible use cases. One example is the coupling to other modelling tools like Life cycle analysis or integrated assessment modelling, where data can be exchanged flexibly through a coded solution.

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. ODYM is therefore widely used in the industrial ecology community and beyond. One of ODYM's strengths is that it builds on an abstraction of the principles and structures of MFAs, such as:

- formalizing a system definition and establishing mass conservation checks
- formalizing dynamic stock models, as described later in Lauinger et al. (2021)
- translating the abstract concepts of processes, stocks, flows, and parameters into a general library, without prescribing any details about the MFA structure, such as its dimensions.

flodym is based on the concepts of ODYM such that its structure, scope and strengths are similar to ODYM. However, there are also aspects in which *flodym* aims to fill gaps and add value, setting it apart from the original:

- ODYM stores dimensionality information in its array objects, but does not harvest the full potential of this information. *flodym* uses dimensionality information for complete internal dimension management in operations of multi-dimensional arrays. For example, the ODYM-based code

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

  reduces to

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

  using `FlodymArray` objects. This allows to write simpler code and reduces errors. For example, dimensions of the same size could simply be switched in the `einsum` statement, which yields wrong results but goes unnoticed by the code. More importantly, it makes the code flexible (hence the name *flodym*) for adaptation and extension. Since the dimensions of each object are not explicitly given for every array operation, but only once in the array definition, dimensions can be added, removed or re-ordered later with minimal changes to the source code.

- Slicing is eased in a similar way. If, for example, only the values of the `waste` array for the `C` (carbon) entry of the `element` dimension are needed, the ODYM syntax

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

  simplifies to

  ```
  waste['C']
  ```

  Again, this allows for adding or removing other dimensions later, or changing the position of the `C` entry in the `element` dimension, without having to change the code. Apart from these functionalities, which are built on Python's magic methods, `FlodymArrays` feature a large range of built-in conventional methods for dimension manipulation, such as `sum_over`, `cast_to` or `get_shares_over`.

- Data read-in and initialization in ODYM prescribes a strict format based on Excel files. There is no data export functionality. In *flodym*, data read-in and export are based on pandas, opening them to a wide range of formats. Users can either use pre-built *flodym* read-in functions, or write their own, and generate objects from data frames. On data read-in, *flodym* performs checks on the data, detecting errors early on. Data read-in is performance-optimized especially for sparse arrays, since the full array size is only used after converting the input pandas data frame to a numpy array. Data is type-checked through the use of pydantic (Colvin et al., 2025), adding robustness to the code.

- ODYM contains the possibility of data export to a non-Python Sankey plotting tool, but no other visualization tools. In *flodym*, general visualization routines are implemented for pyplot (Hunter, 2007) and plotly (Plotly Technologies Inc., 2015) visualization, including plotting of multi-dimensional arrays, and Sankey plots of the MFA system.
- In ODYM, the class for dynamic stock models does not allow for dimensions apart from time. It also does not contain integrated methods for all required computation steps. Moreover, the stock objects which are used in the MFA system do not contain inflow, outflow, and stock, but only one of the three, distinguished by a `Type` attribute. To transfer the results of the dynamic stock model into the MFA, one 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 somewhat cumbersome and a performance bottleneck. In *flodym*, the treatment of material stocks is simplified and integrated with the rest of the MFA. This is realized through `Stock` objects containing `FlodymArray` objects for inflow, outflow and stock arrays, as well as a lifetime model and compute functions. Both stock and lifetime model are multi-dimensional and part of the MFA system class, such that the interaction with them is seamless and the performance gains of numpy array operations are leveraged.
- *flodym* features various smaller functional extensions compared to ODYM. For example, stock models can handle non-evenly-spaced time step vectors, or sub-year lifetimes.
- ODYM features several great application examples, but only a partial API reference, and the API does not always follow PEP 8 naming conventions. The whole *flodym* code incorporates principles of software development (such as PEP 8 formatting, or GitHub actions for tests and documentation building) and clean code, easing future collaboration and extension. The code is extensively documented, including docstrings, type hints, an API reference, how-tos and examples.

The required changes compared to ODYM were so extensive that a refactoring (prior to the functional extension) was estimated to be far more work than a re-write, which is why this path was chosen.

Other existing open MFA packages such as OMAT (Villalba & Hoekman, 2018) or STAN (Cencic & Rechberger, 2008) are different in scope: They are no libraries, but rather comprehensive tools, which eases their use, but limits the flexibility for using them in non-standard ways like *flodym* allows. The same applies to the pymfa (Thiébaud et al., 2019) and PMFA (Kawecki-Wenger, n.d.) packages, which are moreover focused on probabilistic MFA as an extension or special case of MFA.

Software design

A lot of the improvements over ODYM, which are laid out in the previous section, are about software design:

- Internalizing functionality where it is possible, building on abstraction (dimension management in FlodymArray objects). The abstraction also makes applications using this library more sustainable, as less information has to be provided on specific data structures (e.g. array dimensionality)
- Seamless integration of different functionalities which were previously separate (dynamic stock models and MFASystems)

- Providing a coherent, as clean as possible API, which allows writing expressive code (for example through labelled indexing). Here, an emphasis is also put on predictability. For example, numpy nomenclature is used where it is possible.

What's more, *flodym* features different levels of integration: Users can decide whether to only use the most basic data containers and their own customized code, or whether to build on integrated functions provided by the library.

Research impact statement

flodym is a new tool and therefore applications have not reached publication yet, but it's external user base is growing rapidly.

Finished large-scale projects using *flodym* are the in-house REMIND-MFA (Dürrwächter et al., 2025) and the external TRANSIENCE EU MFA (Saurat et al., 2025).

flodym was and will be used for the following teaching events. None of these are organized by the authors of the publication, demonstrating rapid take-up of the library in the community:

- An autumn school on LCA-MFA coupling was heavily based on *flodym* (Mutel, 2025), (*Schools-2025-November-Switzerland*, 2026), resulting for example in the development of several Github repositories (*PAW_MFA_LCA_2025*, 2026), (*DdS_REFLOC*, 2026), (*PVProject*, 2026), (*Dds2025manure*, 2026).
- An invited lecture on flodym at Brightcon 2025 ("Brightcon 2025, hackathon & courses in Grenoble and online," 2026)
- A university class at Leiden University with 150 students.
- The ISIE-SEM Summer School in 2026 (following a similar 2024 event (*ISIE-SEM Summer School*, 2024)).

The repository currently has 22 stars on Github.

AI usage disclosure

AI was used during generation of the *flodym* source code: The auto-complete functionality of Github Copilot was used in VSCode. The Github Copilot coding agent was used in minor development projects with a total of 11 commits with 729 lines. Different AI tools were also used to write parts of the tests.

AI played a minor role in paper writing, assisting in re-wording some of the paragraphs.

All code and text generated or modified by AI was proof-read by humans. AI-generated code was also extensively tested.

Acknowledgements

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

Thank you to Sally Dacie and other non-author contributors to the project.

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

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

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

Cencic, O., & Rechberger, H. (2008). Material flow analysis with software STAN. In *Environmental informatics and industrial ecology*. Shaker Verlag.

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/

dds2025manure. (2026, January). https://github.com/gergosuto/dds2025manure

DdS_REFLOC. (2026, January). https://github.com/MeYiwen/DdS_REFLOC

Dürrwächter, J., Hosak, M., Weiß, B., Zhang, Q., & Ueckerdt, F. (2025). *REMIND-MFA*. https://github.com/pik-piam/remind-mfa

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

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

ISIE-SEM summer school. (2024, January). https://is4ie.org/events/event/isie-sem-summer-school

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

Lauinger, D., Billy, R. G., Vásquez, F., & Müller, D. B. (2021). A general framework for stock dynamics of populations and built and natural environments. *Journal of Industrial Ecology*, *25*(5), 1136–1146. https://doi.org/10.1111/jiec.13117

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

Mutel, C. (2025, July). *DdS autumn school 2025*. https://www.d-d-s.ch/schools/nov-25

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

PAW_MFA_LCA_2025. (2026, January). https://github.com/isabelapi/PAW_MFA_LCA_2025

Plotly Technologies Inc. (2015). *Collaborative data science*. Plotly Technologies Inc. https://plot.ly

PVProject. (2026, January). https://github.com/Tanima-Sharma/PVProject

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

Schools-2025-november-switzerland. (2026, January). https://github.com/Depart-de-Sentier/

Schools-2025-November-Switzerland

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) [Computer software]. 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. *Journal of Industrial Ecology*, *22*(2), 434–442. https://doi.org/10.1111/jiec.12553