The Journal of Open Source Software

sfctools - A toolbox for stock-flow consistent, agent-based models

Thomas Baldauf [1]

1 German Aerospace Center (DLR), Institute of Networked Energy Systems, Curiestr. 4, 70563 Stuttgart, Germany

DOI: 10.21105/joss.04980

Software

- Review
- Repository
- Archive

Editor: Sebastian Benthall

Reviewers:

- @npalmer-professional
- @alanlujan91

Submitted: 11 October 2022
Published: 06 July 2023

Summary

One of the most challenging tasks in macroeconomic models is to describe the macro-level effects from the behavior of meso- or micro-level actors. Whereas in 1759, Adam Smith was still making use of the concept of an 'invisible hand' ensuring market stability and economic welfare (Rothschild, 1994), a more and more popular approach is to make the 'invisible' visible and to accurately model each actor individually by defining its behavioral rules and myopic knowledge domain (Castelfranchi, 2014; Cincotti et al., 2022). In agent-based computational economics (ACE), economic actors correspond to dynamically interacting entities, implemented as agents in a computer software (Axtell & Farmer, 2022; Klein et al., 2018; Tesfatsion, 2002). Such agent-based modeling (ABM) is a powerful approach utilized in economic simulations to generate complex dynamics, endogenous business cycles and market disequilibria. For many research topics, it is useful to combine agent-based modeling with the stock-flow consistent (SFC) paradigm (Caiani et al., 2016; Caverzasi & Godin, 2015; Nikiforos & Zezza, 2018). This architecture ensures there are no 'black holes', i.e. inconsistent sources or sinks, in an economic model. SFC-ABM models, however, are often intransparent and rely on very peculiar, custom-built data structures, thus hampering accessibility (Bandini et al., 2009; Hansen et al., 2019). A tedious task is to generate, maintain and distribute code for ABM, as well as to check for the inner consistency and logic of such models.

Statement of need

sfctools is an ABM-SFC modeling toolbox, which i) relies on transparent and robust data structures for economic agents, ii) comes along with a simple descriptive modeling approach for agents, iii) provides an easy project builder for Python, making the software runnable and accessible on a large number of platforms, and iv) is also manageable from a graphical user interface (GUI) for ABM-SFC modeling, shipped as part of the toolbox, assuring analytical SFC-check and double accounting consistency. The package is shipped in the form of an open-source project.

sfctools was designed to be used by both engineering-oriented and economics-oriented scholars who have basic education in both fields. It can be used by a single developer or by a small development team, splitting the work of model creation in terms of consistency and economic logic from the actual programming and technical implementation. This allows software solutions from rapid prototyping up to more sophisticated, medium-sized ABMs. **sfctools** is therefore a versatile virtual laboratory for agent-based economics.

Unlike more generic frameworks like *mesa* or *AgentPy*, it concentrates on agents in economics. Table 1 shows the different feature coverage of selected modeling frameworks and of **sfctools**.[1]

[1] A more detailed review about ABM and simulation tools can be found in (Abar et al., 2017).

Most other available software packages focus on enabling (multi-) agent-based frameworks at a generic level. However, a standard implementation of accounting balances and the possibility to design and manipulate them in an agent-based environment has not yet been covered. The strength of **sfctools** is therefore three-fold: it tackles the SFC modeling aspect, includes the ABM aspect and provides a graphical interface.

Table 1: Comparison of **sfctools** with other modeling frameworks.

	SFC aspect	ABM aspect	GUI available	Language	Scientific Area
sfctools	x	x	x	Python	Economics
Mesa (Masad & Kazil, 2015)		x	(x)*	Python	Generic
AgentPy (Foramitti, 2021)		x	(x)*	Python	Generic
ABCEconomics (Taghawi-Nejad, 2017)		x	x	Python	Economics
Foundation (Stephan Zheng, 2020)		x		Python	Economics, AI
SFC Models (Romanchuk, 2016)	x			Python	Economics
NetLogo (Gooding & Gooding, 2019)		x	x	(own)	Generic
LSD (Valente, 2008)		x	x	C++	Economics, Generic
FLAME (Chin et al., 2012)		x	x	Java	Generic
FAME (Schimeczek et al., 2023)		x	x	Java, Python	Energy Economics, Generic
JABM (Phelps, 2012)		x		Java	Generic

*) via plotting interface

Basic structure

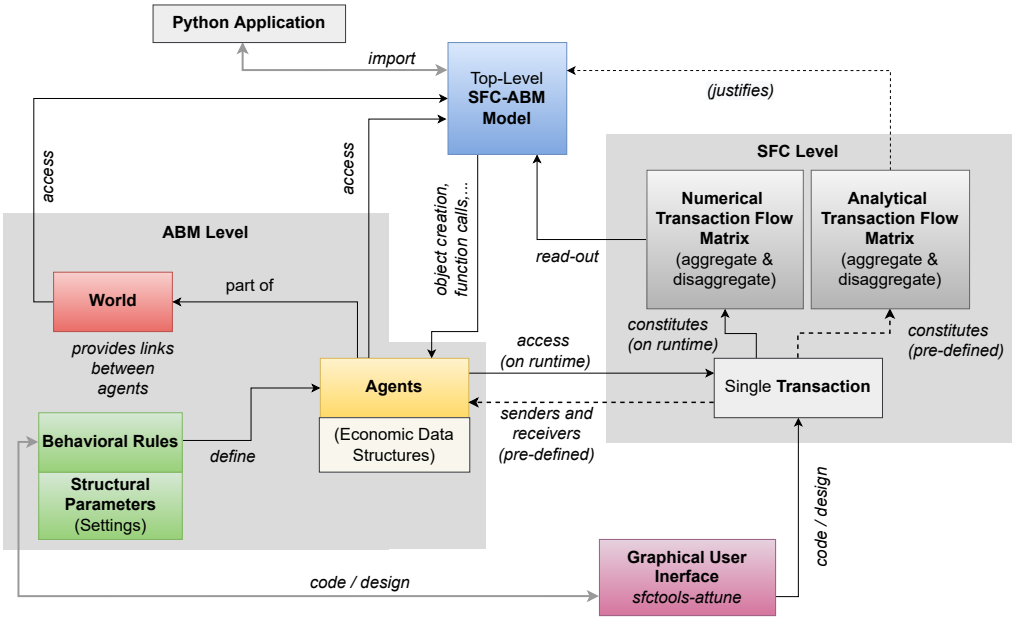

Figure 1: Overview of the **sfctools** toolbox.

Figure 1 shows the basic structure of the modeling framework. The framework supports an efficient, yet powerful SFC-ABM model creation and execution workflow. Users can either program their models directly, using the **sfctools** Python package, or can use the graphical user interface (**sfctools-attune**) to design their models at all aggregation levels. This refers to the implementation of behavioral rules and structural parameters (green boxes), and the design of a set of individual balance sheet transactions (plain gray box). Once the basic model setup is created, the users can check for stock-flow consistency by analytically examining the transaction flow matrix (TFM), taking all theoretically allowed transactions and changes in stocks into account. When running the model, the aggregate and disaggregate TFM is available also as a numerical result. The same is true for data structures on the individual agent level (yellow box): the balance sheets, income statements and cashflow statements of individuals can be consistently logged and accessed on runtime or ex-post. In the background, the **sfctools** core framework will take care of all computational operations and thereby assure stock-flow consistency at all times.

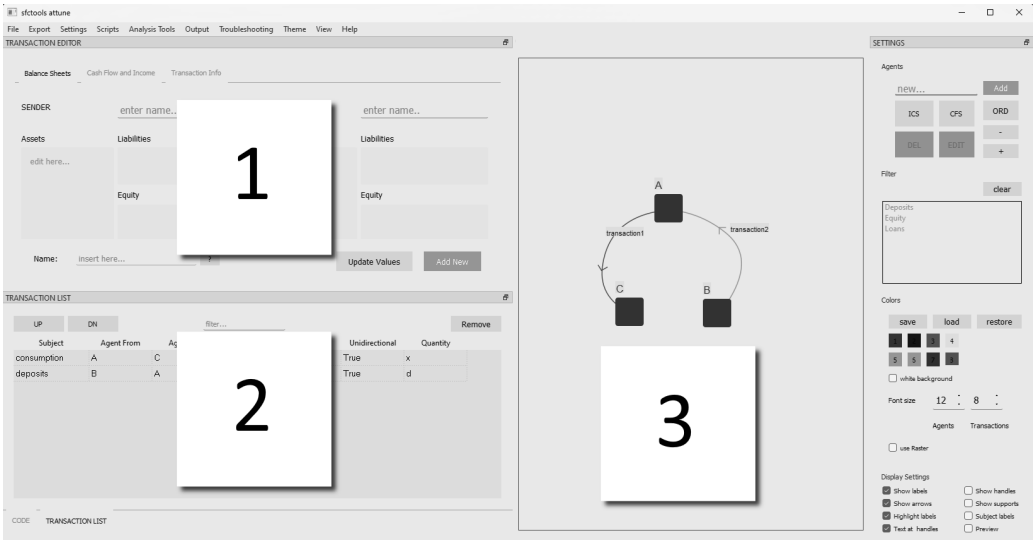

Figure 2: Screenshot of the user interface. 1: Transaction editor, 2: Transactions overview, 3: Graph view.

Figure 2 shows a screenshot of the user interface **attune** (**a**gen**t**-based **t**ransaction and acco**un**ting graphical user interface) shipped along with the **sfctools** framework. In this simple graphical interface, transactions can be edited, sorted and graphically analyzed. Also, structural parameters are edited in form of a yaml-styled summary. The GUI allows for several development productivity tools, such as the analytical pre-construction of the TFM. The main window of the GUI consists of three sub-panels: First, it shows the transaction editor panel (1). Here, the user can directly access the balance sheets of the 'sender' and the 'receiver' agent, which are both equipped with a double entry balance sheet system. Also, the entries for income and cash flow can be manually set in this transaction, and the user can define which flows and stocks are addressed. Second, the user can access already created transactions within the transactions overview panel (2). Here, entries can be edited, deleted or sorted, and distinguished features are given as an overview. Third, a graphical representation of the agents is generated in the graph view (3), where different flows and relations are visualized and can be filtered by balance sheet items or by involved agents in an interactive way. The user can freely lay out and colorize the created agent relations in a graph structure.

Model example

Let us consider a simple model example, namely a three-agent model consisting of a consumer agent (A), a bank (B) and a consumption good producer (C). We employ two transactions (labeled 1 and 2). In the first transaction, B grants a loan to A. Subsequently, A uses its bank deposits to obtain some goods at C. In this simple model, the first transaction only affects the stocks, whereas the second transaction (consumption) is an actual flow.

The model creation workflow is as follows

1. Set up the agents (boxes in the graph view): We add the three agents A, B and C to the model graph. Each node will contain a construction plan for an agent.
2. Set up the transactions (arrows between boxes): Agent A is a consumer and is affected both by both transactions, agent B is a bank and is affected only by transaction 1, agent C is a consumption good producer and is affected only by transaction 2. Once the transactions are registered in the project, they can be deliberately used during the simulation by importing them from an automatically-generated *transactions.py* file.

3. Generate the TFM: To ensure our model is fully stock-flow consistent, we check if all rows and columns of the TFM matrix sum up to zero.

	A	B	C	Total
Consumption	-x	0	+x	0
Δ Deposits	-d+x	+d	-x	0
Δ Loans	+d	-d	0	0
Total	0	0	0	0

4. By exporting our model to Python code via saving the project from the GUI, we automatically generate a fully consistent model, usable in any Python script.

Thanks to the user friendliness of **sfctools**, there is little work to be done in terms of coding. In the GUI, we have the possibility to code the three agents in a custom-designed agent parametrization language for **sfctools-attune**. The code is complemented by a simple Python script to finally run the model. The full example can be found on the project documentation page[2] under *Examples*.

Acknowledgements

I want to thank several people who directly or indirectly got in touch with this software for their constructive remarks: Ardi Latifaj during his master thesis work for his extensive feature requests, Franscesco Lamperti and researchers at Scuola Superiore Sant'Anna for their critical remarks about the framework concept, Patrick Mellacher for his pre-release feedback, Joel Foramitti for his advice on agent-based open-source development, Jonas Eschmann and Luca Fierro for their feedback on the graphical interface and Patrick Jochem for scientific advice on projects being developed using the framework. Karsten Müller and Philipp Harting helped to test the examples. Special thanks go to Benjamin Fuchs for extensive code reviews during the pre-release phase and co-maintenance of the repository. Also, many DLR colleagues supported the project. Last but not least, I am highly grateful to the two reviewers for their excellent constructive feedback on the paper and the code, which has improved the accessibility of the framework for a broad community. All remaining errors are mine.

This work was entirely funded by the German Aerospace Center (DLR).

References

Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. (2017). Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, *24*, 13–33. https://doi.org/10.1016/j.cosrev.2017.03.001

Axtell, R. L., & Farmer, J. D. (2022). Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature*.

Bandini, S., Manzoni, S., & Vizzari, G. (2009). Agent based modeling and simulation: An informatics perspective. *Journal of Artificial Societies and Social Simulation*, *12*(4), 4.

Caiani, A., Godin, A., Caverzasi, E., Gallegati, M., Kinsella, S., & Stiglitz, J. E. (2016). Agent based-stock flow consistent macroeconomics: Towards a benchmark model. *Journal of Economic Dynamics and Control*, *69*, 375–408. https://doi.org/10.2139/ssrn.2664125

Castelfranchi, C. (2014). Making visible "the invisible hand": The mission of social simulation. In *Interdisciplinary applications of agent-based social simulation and modeling* (pp. 1–19). IGI Global. https://doi.org/10.2139/ssrn.2741107

[2] sfctools-framework.readthedocs.io/en/latest/

Caverzasi, E., & Godin, A. (2015). Post-keynesian stock-flow-consistent modelling: A survey. *Cambridge Journal of Economics*, *39*(1), 157–187. https://doi.org/10.1093/cje/beu021

Chin, A., Worth, A., Greenough, A., Coakley, S., Holcombe, M., & Kiran, M. (2012). Flame: An approach to the parallelisation of agent-based applications. *Work*, *501*, 63259.

Cincotti, S., Raberto, M., & Teglio, A. (2022). Why do we need agent-based macroeconomics? *Review of Evolutionary Political Economy*, *3*(1), 5–29. https://doi.org/10.1007/s43253-022-00071-w

Foramitti, J. (2021). AgentPy: A package for agent-based modeling in python. *Journal of Open Source Software*, *6*(62), 3065. https://doi.org/10.21105/joss.03065

Gooding, T., & Gooding, T. (2019). Netlogo. *Economics for a Fairer Society: Going Back to Basics Using Agent-Based Models*, 37–43.

Hansen, P., Liu, X., & Morrison, G. M. (2019). Agent-based modelling and socio-technical energy transitions: A systematic literature review. *Energy Research & Social Science*, *49*, 41–52. https://doi.org/10.1016/j.erss.2018.10.021

Klein, D., Marx, J., & Fischbach, K. (2018). Agent-based modeling in social science, history, and philosophy. An introduction. *Historical Social Research/Historische Sozialforschung*, *43*(1 (163), 7–27.

Masad, D., & Kazil, J. (2015). MESA: An agent-based modeling framework. *14th PYTHON in Science Conference*, *2015*, 53–60.

Nikiforos, M., & Zezza, G. (2018). Stock-flow consistent macroeconomic models: A survey. *Analytical Political Economy*, 63–102. https://doi.org/10.1111/joes.12221

Phelps, S. (2012). Applying dependency injection to agent-based modeling: The JABM toolkit. *Centre for Computational Finance and Economic Agents (CCFEA): Colchester, UK*.

Romanchuk, B. (2016). *SFC models package introduction*. https://github.com/brianr747/SFC_models

Rothschild, E. (1994). Adam smith and the invisible hand. *The American Economic Review*, *84*(2), 319–322.

Schimeczek, C., Deissenroth-Uhrig, M., Frey, U., Fuchs, B., Ghazi, A. A. E., Wetzel, M., & Nienhaus, K. (2023). FAME-core: An open framework for distributed agent-based modelling of energy systems. *Journal of Open Source Software*, *8*(84), 5087. https://doi.org/10.21105/joss.05087

Stephan Zheng, S. S., Alexander Trott. (2020). *The AI economist: Improving equality and productivity with AI-driven tax policies*.

Taghawi-Nejad, D. et al. (2017). *abcEconomics the agent-based computational economy platform that makes modeling easier*. https://abce.readthedocs.io/en/master/

Tesfatsion, L. (2002). Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, *8*(1), 55–82. https://doi.org/10.2139/ssrn.305080

Valente, M. (2008). *Laboratory for simulation development: lsd*. LEM Working Paper Series.