SoftwareX 15 (2021) 100784

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

DIETERpy: A Python framework for the Dispatch and Investment Evaluation Tool with Endogenous Renewables

Carlos Gaete-Morales *, Martin Kittel, Alexander Roth, Wolf-Peter Schill

German Institute for Economic Research (DIW Berlin), Mohrenstr. 58, D-10117 Berlin, Germany

ARTICLE INFO

Article history:
Received 30 September 2020
Received in revised form 16 March 2021
Accepted 27 July 2021

Keywords:
Power sector modeling
Open-source modeling
GAMS
Python
Energy storage
Flexibility options
Sector coupling
Renewable energy integration

ABSTRACT

DIETER is an open-source power sector model designed to analyze future settings with very high shares of variable renewable energy sources. It minimizes overall system costs, including fixed and variable costs of various generation, flexibility and sector coupling options. Here we introduce DIETERpy that builds on the existing model version, written in the General Algebraic Modeling System (GAMS), and enhances it with a Python framework. This combines the flexibility of Python regarding pre- and post-processing of data with a straightforward algebraic formulation in GAMS and the use of efficient solvers. DIETERpy also offers a browser-based graphical user interface. The new framework is designed to be easily accessible as it enables users to run the model, alter its configuration, and define numerous scenarios without a deeper knowledge of GAMS. Code, data, and manuals are available in public repositories under permissive licenses for transparency and reproducibility.

Code metadata

Current code version	v0.3.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_75
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python, GAMS
Compilation requirements, operating environments & dependencies	Python 3.6, GAMS +24.8
Link to developer documentation/manual	https://diw-evu.gitlab.io/dieter_public/dieterpy
Support email for questions	Carlos Gaete cgaete@diw.de

Software metadata

Current software version	v0.3.3
Permanent link to executables of this version	https://pypi.org/project/dieterpy/0.3.3/
Legal Code License	MIT
Computing platform/Operating Systems	Linux, Microsoft Windows
Installation requirements	Python 3.6, GAMS +24.8, conda
Link to developer documentation/manual	https://diw-evu.gitlab.io/dieter_public/dieterpy
Support email for questions	Carlos Gaete cgaete@diw.de

1. Motivation and significance

Mitigating climate change calls for a decarbonization of economies around the world. Power sectors are among the most CO_2-intensive sectors, and renewable energy sources play a major role in decarbonizing them [1]. Wind and solar energy sources are abundant, but come with specific characteristics, such as high

* Corresponding author.
E-mail addresses: cgaete@diw.de (Carlos Gaete-Morales), mkittel@diw.de (Martin Kittel), aroth@diw.de (Alexander Roth), wschill@diw.de (Wolf-Peter Schill).

https://doi.org/10.1016/j.softx.2021.100784

fixed and low variable costs, and temporally variable production patterns. Their cost-efficient large-scale integration into energy systems calls for numerical analyses. As to how they can replace fossil fuels and become the backbone of future energy systems particularly raises questions on the role of temporal flexibility options and sector coupling.

DIETER is a power sector model developed to address such questions of temporal flexibility options and sector coupling in future scenarios with high shares of variable renewable energy. While the model was first used to investigate the role of electricity storage, it has subsequently been extended to also consider various sector coupling options, such as battery-electric vehicles (BEV), power-to-heat, green hydrogen, as well as solar prosumage.

Here we introduce a new major development of the model, DIETERpy. Its core model code builds on previous versions of DIETER, written in the General Algebraic Modeling System (GAMS), to which we refer as DIETERgms from now on. DIETER has been first introduced to the literature in the context of analyzing long-term electricity storage needs for variable renewable energy sources [2,3]. The functionalities added and enhanced by DIETERpy provide new tools for simpler and more comprehensive scenario runs, facilitate a more convenient configuration of the model, and enhance accessibility for users. In DIETERpy, the original algebraic GAMS model is embedded in a Python framework, or wrapper, that is responsible for the (1) configuration of the model, (2) definition of the scenarios to be investigated, and (3) pre- and post-processing of the data.

Python, as an interpreted language, has been used as a wrapper for many highly efficient programs in their respective fields, such as C/C++, Fortran, or GAMS. In the literature there are several examples of applications in energy and climate change, for instance, the Python interface for the Dutch Atmospheric Large Eddy Simulation, or the Python Wrapper for System Advisor Model SAM [4,5]. Other examples for GAMS models with Python wrappers in the energy modeling space include IIASA's MESSAGEix modeling framework and the dispatch model DispaSET [6,7]. To our knowledge, DIETERpy is the first capacity expansion power sector model that uses the GAMS API for Python. We also go one step further than previous approaches of Python model wrappers by developing a tool that enables the creation of scenarios by modifying parameter values, variable bounds and (de-)activation of constraints without the need for altering the original model code. DIETERpy also allows running model instances in parallel by using several processor cores and the implementation of the GUSS tool, an advanced GAMS feature that reduces the compilation times of similar scenarios through model instances. Not least, embedding our legacy model DIETER in a Python framework substantially enhances the functionalities of the previous pure GAMS-based version, which may be useful for many current and future users of the model.

DIETERpy can easily be installed via Python package managers, and no extensive knowledge of GAMS is needed for initial runs. The model can be configured either by a browser-based graphical user interface or by `CSV` files. For standard scenario runs, the GAMS-based core model code does not have to be altered by the user, but only for more fundamental model changes. DIETERpy has a data post-processing routine that collects the results of multiple model runs and makes them accessible for further analysis in different output formats, allowing users to proceed with their tools of choice. Basic model results can also be visualized within the browser interface.

2. Software description

DIETER is a power sector capacity expansion model based on linear equations. It minimizes total system costs in a long-run equilibrium setting under perfect foresight. In economic terms, it takes a social planner perspective. Its objective function is the sum of investment costs into various generation and flexibility technologies as well as transmission, and related variable costs for a given time period (usually a full year). DIETER minimizes these costs, given exogenous techno-economic input parameters and time series such as demand and renewable availability which are provided in an hourly resolution. Hence, the solution is an optimal portfolio of generation and flexibility technologies such as electricity storage, as well as their optimal dispatch for the given time period. The model is solved for all consecutive hours of a target year.

A range of equations implements constraints with respect to generation capacities, renewable shares, CO_2 emissions, renewable availability, and balancing reserves. Further, there are inter-temporal restrictions related to various types of storage and sector coupling. The model does not include a detailed representation of the underlying transmission grid infrastructure. Within regions, e.g., European countries, it makes a copper-plate assumption; between regions, DIETER uses a simple transport model approach with Net Transfer Capacities (NTCs). Different versions of DIETER have been developed and applied in several research projects, which has led to a growing number of publications in relevant field journals and several ongoing working papers (see Section 4).

From the beginning, DIETER was designed as a lean, tractable and flexible open-source tool. The model applications that were published so far hardly required high-performance computing resources, but could largely be solved on a standard computer. The new DIETERpy framework aims to further increase the model's accessibility and usefulness also for casual users and practitioners.

DIETERpy includes some basic data in its installation, which the user may have to adjust, depending on the types of scenario analyses to be investigated. We provide a range of techno-economic input parameters for the available technology portfolio. This includes time-invariant costs parameters related to investment in various generation and flexibility technologies as well as cross-border transfer capacities, fuel costs, flexibility costs, and variable costs of different technologies, as well as technology-specific efficiencies, technical life-times, and upper/lower capacity expansion bounds. Further, we include geographical characteristics, as well as time series of renewable availability and demand for electricity and heat. All of the data is freely available and sources are listed within the respective spreadsheets.

2.1. Software architecture

DIETERpy is a software package written in the programming languages Python and GAMS. It can be installed via the Python package manager *pip*. Running the model requires a license for the GAMS Base Module and an LP solver (we use CPLEX).[1] The left panel of Fig. 1 provides an overview of the software architecture of DIETERpy. Data pre- and post-processing as well as the management of different scenario runs are carried out with Python, only the optimization itself is done within a GAMS module that is initiated via Python. The user can choose between different data output formats to further analyze the results.

[1] The GAMS Software GmbH told us that potential users who want to explore DIETERpy can ask for a test license, sales@gams.com.

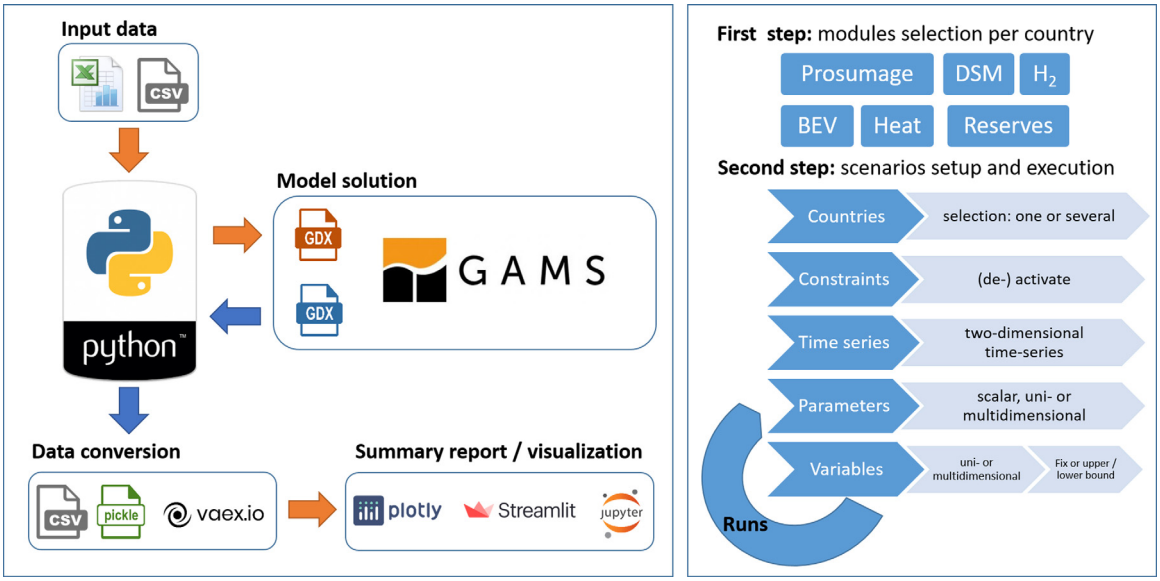

Fig. 1. Graphical overview of DIETERpy.
Source: Own illustration.

The right panel of Fig. 1 summarizes the two steps required before running optimization with DIETERpy. The first step consists of selecting modules. The basic module contains all variables and constraints for a default power sector dispatch and investment model. Additional model features, such as demand-side management (`dsm`), battery-electric vehicles (features `ev_endogenous` and `ev_exogenous`), balancing reserves (`reserves`), solar prosumage (`prosumage`) and power-to-heat (`heat`), can be enabled. In a second step, multiple scenario runs can be specified. Within a scenario run, it is possible to modify parameters, time series and variable bounds, and select different constraints and sets of countries. Parameters and variables have to be indicated with their domain unless it is a scalar or a dimensionless variable. The domain must contain set names or set elements. Unidimensional and multidimensional parameters and variables are supported. For variables, we can establish upper and lower bounds, as well as fixed values.

2.2. Software functionalities

In DIETERpy, the user can select modules and other settings using an easy-to-access browser-based graphical user interface, or editing the corresponding configuration `CSV` files. In order to get access to the configuration files, a project has to be created. This can be done by calling the `create_project` function after typing `dieterpy` in the command line. This will result in a new folder that entails all configuration files and input data. Fig. 2 shows the folders and the file tree of a project.

Out-of-the box, we allow the user to alter:

1. basic program settings (`project_variables.csv`),
2. modules (`features_node_selection.csv`), and
3. the scenario table (`iteration_table.csv`).

Basic program settings include, for instance, the definition of input files, and the output file formats; here, the user can choose between `CSV`, `Pickle`, and `VAEX`. As DIETERpy can solve scenarios in parallel, we also allow to adapt specific settings concerning parallelization. In the "basic program settings", the user may further switch between an "investment & dispatch model" and a "dispatch only model". Cross-border electricity flows can also be easily switched on and off (see Table 1).

Table 1
Example of the project_variables.csv file.
Source: Own illustration.

Variable	Value
scenarios_iteration	Yes
skip_input	No
skip_iteration_data_file	No
base_year	2030
end_hour	h8760
dispatch_only	No
network_transfer	Yes
no_crossover	Yes
infeasibility	No
GUSS	Yes
GUSS_parallel	Yes
GUSS_parallel_threads	0
data_input_file	static_input.xlsx
time_series_file	timeseries_input.xlsx
iteration_data_file	iteration_data.xlsx
gdx_convert_parallel_threads	0
gdx_convert_to_csv	No
gdx_convert_to_pickle	Yes
gdx_convert_to_vaex	No
report_data	Yes

DIETERpy is a generic model which can be adapted to any geographic setting. In its default version, the calibration is most refined for Germany, yet eleven other countries (France, Denmark, Belgium, Netherlands, Poland, Czech Republic, Austria, Switzerland, Spain, Italy, and Portugal) can also be activated. Adding more countries or applying the model to a different geographic region can be done relatively easily if respective input data on electricity demand as well as meaningful bounds for generation and transfer capacity are available. Different modules can be switched on for different countries, depending on the research question and model size restrictions, as shown in Table 2.[2]

DIETERpy further provides an easy way to change single parameter values or define and control entire scenario runs. The program has an easy-to-modify scenario table

[2] By the time of publication of this article, input data is still missing for several country-module combinations. A green hydrogen module initially was available in the pure GAMS version (DIETERgms) only [8], but it will be released in DIETERpy soon after the publication of this article.

Table 2
Example of the `features_node_selection.csv` file. Zeros (0) deactivate features, ones (1) activate them. Here, we deactivated all optional features. `dsm` refers to demand side management, `ev_endogenous` and `ev_exogenous` to endogenous and exogenous optimization of BEV. For additional information, please refer to the online documentation.
Source: Own illustration.

Module	Node											
	DE	FR	DK	BE	NL	PL	CZ	AT	CH	ES	IT	PT
dsm	0	0	0	0	0	0	0	0	0	0	0	0
ev_endogenous	0	0	0	0	0	0	0	0	0	0	0	0
ev_exogenous	0	0	0	0	0	0	0	0	0	0	0	0
Reserves	0	0	0	0	0	0	0	0	0	0	0	0
Prosumage	0	0	0	0	0	0	0	0	0	0	0	0
Heat	0	0	0	0	0	0	0	0	0	0	0	0

(`iteration_table.csv`) that allows the user to edit (a) the set of countries, (b) specific constraints specifications, (c) entire time series (such as demand or renewable capacity factors), (d) single parameter values and (e) variable bounds (either fixing it or providing upper or lower bounds). The scenario table is an easy-to-use, transparent tool for quickly running the model multiple times in different specifications, where every row in that table refers to one scenario (see Table 3). The parametrization is established by adding column headings to the table that are dedicated to special features. To modify input parameters, we add a column heading that matches the parameter name in the `model.gms` file. For variables, we can set fixed values as well as upper and lower bounds.

Once the configuration files have been edited and adapted for a desired case study, the optimization is ready to start. The excel files that contain the input data and the configuration `CSV` files are loaded, and then a Python routine creates the optimization- and GAMS-compatible `GDX` files. Via the Python-API, the model (`model.gms`) as well as scenario table is passed to GAMS.

For solving, DIETERpy uses the Python-API of GAMS to build a model instance. This model instance exists until a solver in GAMS returns the solutions. We have developed three options to run optimization problems with DIETERpy. One option consists of running the scenarios with independent model instances that compile every time they are built. With this method, the scenarios are solved sequentially. This may lead to long execution times when several scenarios are solved due to the compilation time incurred for each model instance. This problem is solved in part with the second option, the Gather-Update-Solve-Scatter (GUSS) tool in GAMS. It allows running several scenarios with only one model instance by updating the variables, parameters and equation values from one scenario to another. The third option takes advantage of multiprocessing in Python. It enables to run the GUSS tool in several processor cores, in which several scenarios are solved in parallel. This option is the fastest one; yet for complex problems, it may lead to a high usage of memory. Therefore, we added an option to choose the number of cores that should be used in the optimization. To provide an indicative idea about the solving times of DIETERpy, we ran a 4-scenario problem with two countries (Germany & France) for a full year (8760 h). All additional modules are deactivated, thus only investment into power plant capacities, storage and net transfer capacities are possible. Between the four scenarios, we only vary the share of renewable energy in both countries (50%-60%-70%–80% in Germany and 40%-50%-60%–70% in France). We used a computer that runs Windows 10, has an Intel i5-3320M (2 cores, 4 threads), and 6 GB of RAM. The pure solving times (without any pre- and post-processing of the data) are as follows: sequential solving (no GUSS) 15.0 min, GUSS (sequential) 9.6 min, GUSS (parallel, 4 threads) 8.4 min.[3]

[3] Practitioners can reproduce this example following the instruction in the documentation for example2.

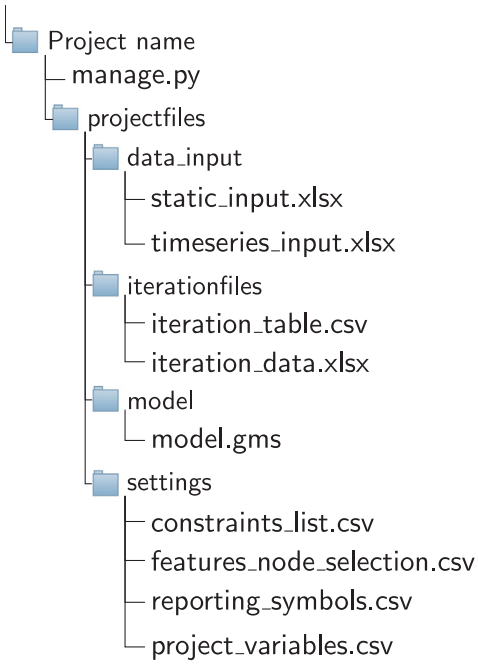

Fig. 2. Folders and files tree of a project.
Source: Own illustration.

GAMS provides the solutions of individual scenario run in `GDX` files, which contain all resulting symbols (parameters, variables, equations and sets). When numerous scenario runs have been solved, collecting and combining all symbols can be computationally intensive. Thus, we have designed three solutions: i) the user can select the symbols to be extracted from the `GDX` files by editing the corresponding file `reporting_symbols.csv`; ii) the number of cores can be selected to extract the symbols in parallel; iii) in case many scenarios and symbols are needed, the *VAEX* library consists of streaming algorithms, memory-mapped files and a zero memory copy policy to explore datasets larger than memory [9]. It is advantageous when dealing with resulting multidimensional variables, and processing such variables for several scenarios increases the risk of running out of memory. Hence, *VAEX* reduces the amount of memory needed, although it increases physical storage usage (about 10 times the *GDX* file size). The selected symbols of each scenario run are extracted and saved as separate `CSV` files or in an individual file per scenario run in case of `Pickle` and `VAEX`. As DIETERpy uses `Pickle` files to proceed with reporting and visualization, it is the recommended format.

DIETERpy has been endowed with Python objects that allow the user to make operations between symbols such as addition,

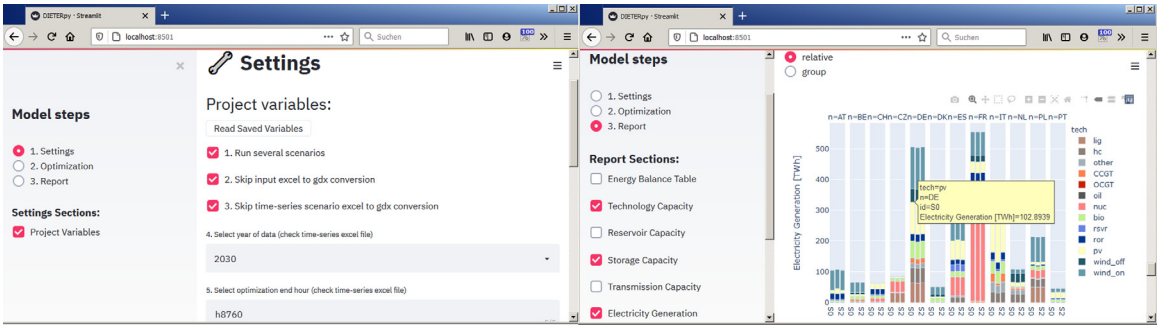

Fig. 3. The graphical user interface. The left panel shows a selection of possible project settings, the right panel model outcomes. *Source:* Own illustration.

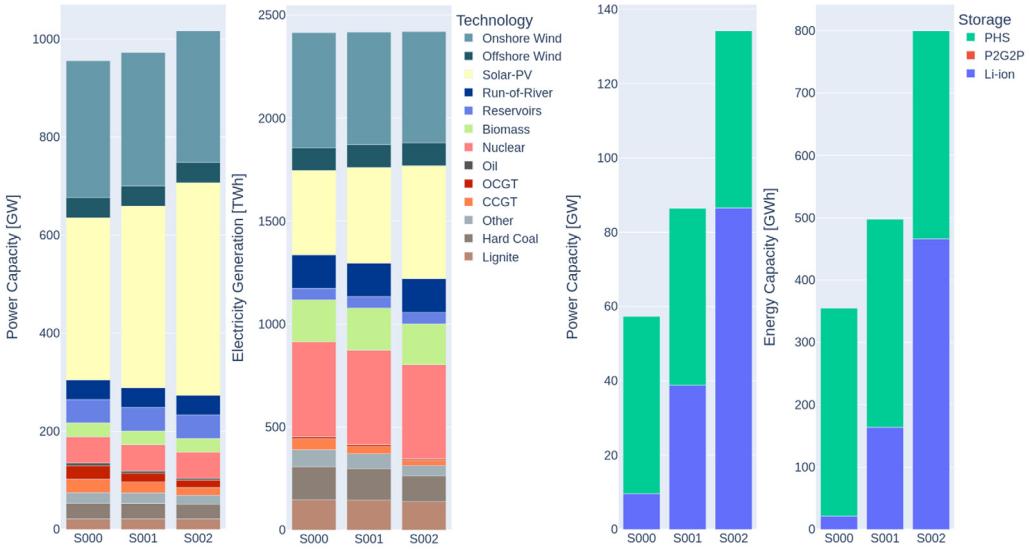

Fig. 4. Installed capacity and total generation of conventional and renewable generators (left panel) and storage (right panel) for three exemplary scenarios. *Source:* Own illustration.

subtraction, multiplication and division, while taking care of symbols' dimensions: the `SymbolsHandler` and `Symbol` classes help the users to analyze the results more efficiently.

Finally, DIETERpy provides a browser-based graphical user interface to interactively visualize the results.[4] Fig. 3 shows two exemplary screenshots. Additional information on how to use the GUI is provided in the online model documentation.

3. Illustrative example

In this section, we provide an exemplary application of DIETERpy, varying the costs of stationary Li-ion battery storage in a mid-term future central European setting. In doing so, we highlight some novel Python features introduced by DIETERpy compared to the previous GAMS-only version DIETERgms. The example can be reproduced after installing DIETERpy and creating a project folder by typing in the terminal `dieterpy create_project -n <project name> -t example1`, where -t stands for *template*.

We adapt `project_variables.csv` to change some basic options (see Table 1). For example, we activate the scenario iteration feature (`scenario_iteration` set to `yes`); we decide to run a full investment and dispatch model (set `dispatch_only` is `no`) with activated electricity exchange between the nodes (`network_transfer` set to `yes`); and we use the maximum number of processor cores for the optimization via the GUSS tool with parallel processes (`GUSS` is `yes`, `GUSS_parallel` is `yes` and `GUSS_parallel_threads` is 0).

To (de-)activate certain model-related modules, we edit the file `features_node_selection.csv` (Table 2). In our example, we deactivate all additional modules for simplicity by setting all entries to 0. Accordingly, only the basic module is used.

We analyze the effects of lower-cost Li-ion storage with three different scenario runs. Model run S000 represents baseline storage cost assumptions; the scenario S001 reflects a cost decrease of both energy- and power-related Li-ion investments of 50%; and scenario S002 assumes a further storage cost reduction to 25% of S000 values. We specify Table 3 accordingly. DIETERgms defines these parameters as `c_i_sto_e(n,sto)`

Table 3
Example of the `iteration_table.csv` file with different assumptions on the annualized costs of Li-ion storage.
Source: Own illustration.

Run	c_i_sto_e(n,'Li-ion') [EUR/MWh]	c_i_sto_p(n,'Li-ion') [EUR/MW]
S000	20029	15021
S001	10014	7511
S002	5007	3755

[4] By the time of writing, this interface is under ongoing development and improvement.

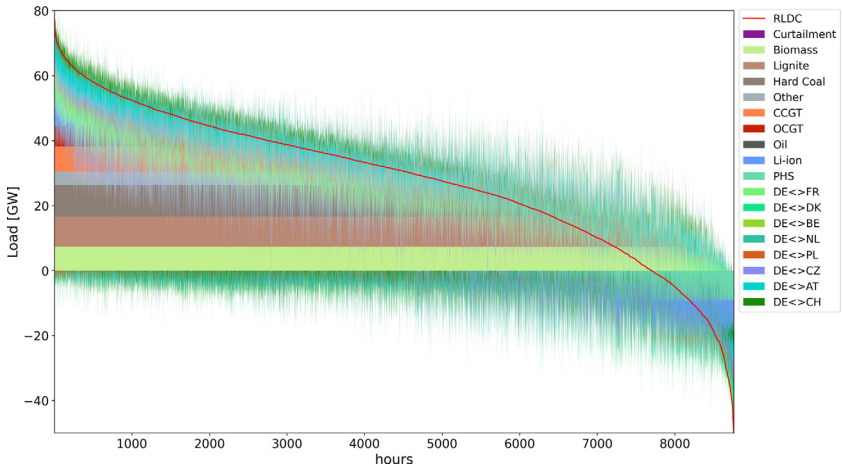

Fig. 5. Residual load duration curve, generation of dispatchable technologies and storage, and transmission flows. This example represents scenario S002 for Germany. *Source:* Own illustration.

and `c_i_sto_p(n,sto)` for energy and power components, respectively. First, as we do not provide the special features `country_set` as a column heading, we choose the entire set of *n* countries to be included in the runs. Second, the set *sto* in the model consists of three elements: "Li-ion" (stationary battery storage), "PHS" (pumped hydro storage, which is fixed in this example) and "P2G2P" (power-to-gas-to-power). As we want to vary only the costs of Li-ion batteries, we provide it literally in the column heading: `c_i_sto_e(n,'Li-ion')` and `c_i_sto_p(n,'Li-ion')`.

Here we only present a snapshot of the results of the exemplary application, using figures generated by our built-in visualization tool. Cheaper batteries generally increase the use of solar PV, both regarding installed capacity and yearly electricity generation (left panel of Fig. 4). This is because short-term battery storage is particularly suited to balance daily PV fluctuations. If the costs of Li-ion batteries decrease as assumed here, they would be deployed with comparable energy and power capacity as existing European pumped hydro storage (right panel of Fig. 4).

Fig. 5 shows an exemplary residual load duration curve (RLDC), combined with the generation of various technologies in respective hours. The typical role of different generation technologies can be seen, e.g., base-load use of lignite and peak-load use of open cycle gas turbines (OCGT). Storage charging mainly occurs in periods of renewable surplus generation, but partly also in other periods.

4. Impact & conclusion

DIETER has been designed to investigate research questions that are highly relevant in current energy transition research. This includes, for example, the exploration of infrastructure requirements for least-cost integration of variable renewable energy sources in the power sector, and the effects of flexibly using wind and solar energy also in other sectors. The model's prosumage module further allows investigating specific aspects of decentralized self-supply with PV-battery systems, which is usually not covered by other energy models.

The new features of the DIETERpy model version presented here will increase the efficiency of numerical model analyses applied to respective research questions. Compared to previous model versions, the new Python implementation allows to more easily specify, run, and evaluate the outcomes of numerous scenarios, and to make use of a great variety of Python libraries for data pre- and post-processing and for visualizing results.

Despite the new Python functionalities, DIETERpy in its core remains a parsimonious model which can be flexibly applied to different parametrizations and research questions. Most of the applications we published so far could be solved on standard desktop computers. Only very large-scale applications covering many regions and multiple sector coupling options may require larger computational resources. Accordingly, DIETERpy may not only be used by highly specialized research groups, but also by students and practitioners.

The Python framework presented here further allows model users without deeper knowledge of programming in either GAMS or Python to run a large range of model applications. In particular, the browser-based graphical user interface increases the ease of use and the accessibility of the model. More expert users are free to adjust and expand any part of DIETERpy as they wish, given the open-source nature of the model and the permissive MIT license.

Thus, we expect that the DIETERpy version presented here will contribute to the further usage and application of the model, especially outside the domain of the current development team. In the past, most publications based on DIETER were co-authored by DIW Berlin researchers. Two articles introduce the basic model version and investigate optimal electrical storage capacity in scenarios with high shares of renewable energy sources [2,3]. Reduced model versions are used for more general reflections of the economics of electrical storage [10] and its changing role in settings with increasing renewable penetration [11]. Three papers on solar prosumage focus on power sector effects for Germany [12] and Western Australia [13], and on how tariff design impacts PV-battery investments [14]. Further model applications analyze the power sector impacts of electric vehicles [15], flexible electric heating [16], and green hydrogen [8]. A demand-side management feature we developed for DIETER was published separately [17]. Recently. we also noticed spin-off versions of the model that have been developed without any involvement of the initial DIETER team. These include, for example, generation cost projections developed for the Australian Energy Market Operator [18], as well as analyses of demand-side management in India [19] and methodological aspects of modeling long-term storage [20]. We expect that the DIETERpy improvements described here will spur more of such activities.

Because of the permissive license, DIETERpy can also be used in commercial settings. Yet we assume that it is more likely to be used in academic and not-for-profit research applications. In fact, the model and its applications have been the backbone of several research grants acquired by us, including such of different German federal ministries and the European Commission.

To draw a more general conclusion, with DIETERpy we show how to embed an existing GAMS-based model into a modern Python framework. Now, the model combines the best of the two programming languages. On the one hand, GAMS is a well-established language for energy models which is proofed to work. It offers a straightforward algebraic formulation and the use of efficient solvers. Maintaining the GAMS core also allows easier linking with other GAMS-based models. On the other hand, Python offers more flexibility, a wide choice of additional libraries, and better opportunities for data pre- and post-processing as well as visualization of results. Using Python also enables an easy-to-use and flexible scenario tool which would be difficult to implement in GAMS. This gives the user the opportunity to run the model in many different scenarios by flexibly altering the model parametrization or its constraints. We believe that DIETERpy could serve as an example also for other established energy models to make them future-proof and more accessible to a wider audience.

Several model improvements and upgrades are currently ongoing or planned. This concerns both the algebraic GAMS core, where we aim for a more complete and more consistent coverage of different sector coupling options, and further advancements of the Python framework. We particularly aim to improve and extended the user interface and the post-processing and data visualization functionalities.

CRediT authorship contribution statement

Carlos Gaete-Morales: Conceptualization, methodology, software (Python), visualization, writing – original draft, review and editing. **Martin Kittel:** Methodology, software (Python), writing – review and editing. **Alexander Roth:** Methodology, software (Python), writing – original draft, review and editing, Online documentation. **Wolf-Peter Schill:** Funding acquisition, methodology, software (GAMS), writing – original draft, review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by research grants of the German Federal Ministry of Education and Research via the projects "Future of Fossil Fuels in the wake of greenhouse gas neutrality", FKZ 01LA1810B, and "Ariadne", FKZ 03SFK5NO.

References

[1] de Coninck H, Revi A, Babiker M, Bertoldi P, Buckeridge M, Cartwright A, et al. Strengthening and implementing the global response. Tech. rep., Intergovernmental Panel on Climate Change; 2018, https://www.ipcc.ch/site/assets/uploads/sites/2/2019/02/SR15_Chapter4_Low_Res.pdf.

[2] Zerrahn A, Schill W-P. Long-run power storage requirements for high shares of renewables: Review and a new model. Renew Sustain Energy Rev 2017;79:1518–34. http://dx.doi.org/10.1016/j.rser.2016.11.098.

[3] Schill W-P, Zerrahn A. Long-run power storage requirements for high shares of renewables: Results and sensitivities. Renew Sustain Energy Rev 2018;83:156–71. http://dx.doi.org/10.1016/j.rser.2017.05.205.

[4] van den Oord G, Jansson F, Pelupessy I, Chertova M, Grönqvist JH, Siebesma P, et al. A Python interface to the Dutch Atmospheric Large-Eddy Simulation. SoftwareX 2020;12:100608. http://dx.doi.org/10.1016/j.softx.2020.100608.

[5] Gilman P, Janzou S, Guittet D, Freeman J, DiOrio N, Blair N, et al. PySAM (Python Wrapper for System Advisor Model "SAM"). 2019, http://dx.doi.org/10.11578/dc.20190903.1.

[6] Huppmann D, Gidden M, Fricko O, Kolp P, Orthofer C, Pimmer M, et al. The MESSAGEix Integrated Assessment Model and the ix modeling platform (ixmp): An open framework for integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development. Environ Model Softw 2019;112:143–56. http://dx.doi.org/10.1016/j.envsoft.2018.11.012.

[7] Kavvadias K, Gonzalez IH, Zucker A, Quoilin S. Integrated modelling of future EU power and heat systems - The Dispa-SET v2.2 open-source model (EUR 29085 EN). Tech. rep., Luxembourg: European Commission; 2018, http://dx.doi.org/10.2760/860626.

[8] Stöckl F, Schill W-P, Zerrahn A. Optimal supply chains and power sector benefits of green hydrogen. Sci Rep 2021;11:14191. http://dx.doi.org/10.1038/s41598-021-92511-6.

[9] Breddels, Maarten A, Veljanoski, Jovan. Vaex: Big data exploration in the era of Gaia. Astron Astrophys 2018;618:A13. http://dx.doi.org/10.1051/0004-6361/201732493.

[10] Zerrahn A, Schill W-P, Kemfert C. On the economics of electrical storage for variable renewable energy sources. Eur Econ Rev 2018;108:259–79. http://dx.doi.org/10.1016/j.euroecorev.2018.07.004.

[11] Schill W-P. Electricity storage and the renewable energy transition. Joule 2020;4:2059–64. http://dx.doi.org/10.1016/j.joule.2020.07.022.

[12] Schill W-P, Zerrahn A, Kunz F. Prosumage of solar electricity: Pros, cons, and the system perspective. Econ Energy Environ Policy 2017;6:7–31. http://dx.doi.org/10.5547/2160-5890.6.1.wsch.

[13] Say K, Schill W-P, John M. Degrees of displacement: The impact of household PV battery prosumage on utility generation and energy storage. Appl Energy 2020;276:115466. http://dx.doi.org/10.1016/j.apenergy.2020.115466.

[14] Günther C, Schill W-P, Zerrahn A. Prosumage of solar electricity: Tariff design, capacity investments, and power sector effects. Energy Policy 2021;152:112168. http://dx.doi.org/10.1016/j.enpol.2021.112168.

[15] Schill W-P, Niemeyer M, Zerrahn A, Diekmann J. Bereitstellung von Regelleistung durch Elektrofahrzeuge: Modellrechnungen für Deutschland im Jahr 2035. Z Energ wirtsch 2016;40:73–87. http://dx.doi.org/10.1007/s12398-016-0174-7.

[16] Schill W-P, Zerrahn A. Flexible electricity use for heating in markets with renewable energy. Appl Energy 2020;266:114571. http://dx.doi.org/10.1016/j.apenergy.2020.114571.

[17] Zerrahn A, Schill W-P. On the representation of demand-side management in power system models. Energy 2015;84:840–5. http://dx.doi.org/10.1016/j.energy.2015.03.037.

[18] Graham PW, Hayward J, Foster J, Story O, Havas L. GenCost 2018. Tech. rep., CSIRO, Australia; 2018, http://dx.doi.org/10.25919/5c587da8cafe7.

[19] Ershad AM, Pietzcker R, Ueckerdt F, Luderer G. Managing power demand from air conditioning benefits solar PV in India scenarios for 2040. Energies 2020;13(9):2223. http://dx.doi.org/10.3390/en13092223.

[20] de Guibert P, Shirizadeh B, Quirion P. Variable time-step: A method for improving computational tractability for energy system models with long-term storage. Energy 2020;213:119024. http://dx.doi.org/10.1016/j.energy.2020.119024.