The Journal of Open Source Software

TESPy: Thermal Engineering Systems in Python

Francesco Witte[1, 2] **and Ilja Tuschy**[1, 2]

1 Center for Sustainable Energy Systems, Flensburg **2** Flensburg University of Applied Sciences

DOI: 10.21105/joss.02178

Software

- Review
- Repository
- Archive

Editor: Kyle Niemeyer

Reviewers:

- @arosen93
- @corentin-dev

Submitted: 18 February 2020
Published: 21 May 2020

Summary

TESPy (Thermal Engineering Systems in Python) provides a powerful simulation toolkit for thermal process engineering, for instance power plants, district heating systems or heat pumps. With the TESPy package it is possible to design plants and simulate stationary operation. From that point, part-load behavior can be predicted using underlying characteristics for each of the plant's components. The component based structure combined with the solution method offer very high flexibility regarding the plant's topology and its parametrisation. An extensive online documentation is provided at Witte (2020). Several examples and tutorials on how to use the software are included, too.

Motivation

Thermal process simulation is a fundamental discipline in energy engineering. Consequently, there are several well known commercial software solutions available in this field, for example EBSILON Professional or Aspen Plus, mainly used in the industrial environment. A similar approach was carried out by Casella & Leva (2003) with the open source library "ThermoPower" written in the Modelica language. The software provides dynamic modeling and focuses on steam power plants, but has not been updated since 2014 (Politecnico di Milano, 2014).

Here we propose a new solution for simulating thermal processes, since ThermoPower is not currently maintained and also limited to one field of thermal engineering. In order to open the software to a wide (scientific) audience an open source solution in the widespread programming language Python is implemented.

Package Architecture

TESPy is built up in modular structure with three main modules:

- networks: The networks module represents the container for every simulation. Initialisation, preprocessing, solving and postprocessing are handled by the network class.
- components: All components are part of the components module. The package provides basic components, for example different types of turbomachinery, heat exchangers, pipes, valves and mixers or splitters. On top of that, advanced components like a drum, an internal combustion engine or a combustion chamber are available. The individual properties of each component are defined in the respective class.
- connections: Connections are used to connect components and hold the fluid property information. Thus, they represent the plant's topology and its state.

Method

To simulate a specific plant an individual model is created connecting the components to form a topological network. Steady state operation of the plant is determined by the fluid's state on every connection (and therefore its change within components) between the plant's individual components. Thus, based on the components and parameters applied, TESPy generates a set of nonlinear equations representing the plant's topology and its parametrisation. By formulating the equations implicitly parameters and results are generally interchangeable offering high flexibility in the user specifications. The system of equations is numerically solved with an inbuilt solver applying the multi-dimensional Newton-Raphson method to determine the mass flow $\dot{m}$, pressure p, enthalpy h and fluid composition (defined by mass fraction x of each fluid) of every connection. This way, it is possible to solve for both thermal as well as hydraulic state of the plant.

To achieve this, TESPy implements balance equations based on standard literature, for example Baehr & Stephan (2016) or Epple, Leithner, Linzer, & Walter (2012), for every component regarding

- mass flow and fluid composition as well as
- energy (covering thermal and hydraulic properties).

On top of the basic equations presented in this paper, there are many component-specific equations that can be applied by the user. The full list of components, its parameters and the respective equations is documented online in the API section (Witte, 2020). For calculation of fluid properties TESPy uses CoolProp (Bell, Wronski, Quoilin, & Lemort, 2014), which supports a wide range of pure and incompressible fluids. In case of gaseous mixtures the software calculates the fluid properties by the laws of ideal mixtures (Baehr & Stephan, 2016; Herning & Zipperer, 1936).

In steady state, the total mass flow into a component must be equal to the total mass flow leaving the component (eq. 1). Additionally, the mass balance of a specific fluid fl is applied (eq. 2). The energy balance of all components is derived from the stationary energy balance of open systems with multiple inlets and outlets (eq. 3).

In thermal engineering applications the change in kinetic and potential energy due to differences in flow velocity c and height z are usually neglected as these are relatively small compared to change in enthalpy, subsequently the equation can be simplified (eq. 4).

$$0 = \sum_i \dot{m}_{\mathrm{in},i} - \sum_o \dot{m}_{\mathrm{out},o} \tag{1}$$

$$0 = \sum_i \dot{m}_{\mathrm{in},i} \cdot x_{fl,\mathrm{in},i} - \sum_o \dot{m}_{\mathrm{out},o} \cdot x_{fl,\mathrm{out},o} \; \forall fl \in \text{network fluids} \tag{2}$$

$$\begin{aligned} 0 = &\sum_o \dot{m}_{\mathrm{out},o} \cdot \left(h_{\mathrm{out},o} + \mathrm{g} \cdot z_{\mathrm{out},o} + \frac{c_{\mathrm{out},o}^2}{2} \right) \\ &- \sum_i \dot{m}_{\mathrm{in},i} \cdot \left(h_{\mathrm{in},i} + \mathrm{g} \cdot z_{\mathrm{in},i} + \frac{c_{\mathrm{in},i}^2}{2} \right) - P - \dot{Q} \end{aligned} \tag{3}$$

$$0 = \sum_o \dot{m}_{\mathrm{out},o} \cdot h_{\mathrm{out},o} - \sum_i \dot{m}_{\mathrm{in},i} \cdot h_{\mathrm{in},i} - P - \dot{Q} \tag{4}$$

The values of heat $\dot{Q}$ and power P transferred, depend on the individual component properties. For example,

- a pipe does not transfer power, thus only heat may be transferred ($P = 0$).
- turbomachinery is considered adiabatic, thus it only transfers power ($\dot{Q} = 0$).

If chemical reactions take place, the corresponding chemical mass balance is taken into account instead of equation 2. On top of that, the energy balance is different, as the reaction enthalpy has to be considered. Furthermore, it is necessary to compensate for the different zero point definitions of enthalpy in the fluid properties of the reaction components by defining a reference state ref. For example, equation 5 is implemented for adiabatic combustion chambers using the fuel's lower heating value LHV as reaction enthalpy.

$$0 = \dot{m}_{\mathrm{out}} \cdot (h_{\mathrm{out}} - h_{\mathrm{out,ref}}) - \sum_i \dot{m}_{\mathrm{in},i} \cdot (h_{\mathrm{in}} - h_{\mathrm{in,ref}})_i - \dot{m}_{\mathrm{in}} \cdot \sum_j LHV_j \cdot x_j \tag{5}$$

With respect to hydraulic state, the pressure drop $p_{\mathrm{out}} - p_{\mathrm{in}}$ in a pipe can be calculated by the Darcy-Weisbach equation 6 from its dimensions (length L, diameter D) and the Darcy friction factor λ calculated from the pipe's roughness k_{s}, its diameter and the Reynolds number Re. A simpler approach is to specify the pressure ratio pr.

$$0 = p_{\mathrm{out}} - p_{\mathrm{in}} + \frac{\rho \cdot c^2 \cdot \lambda\left(\mathrm{Re}, k_{\mathrm{s}}, D\right) \cdot L}{2 \cdot D} \tag{6}$$

$$0 = pr - \frac{p_{\mathrm{out}}}{p_{\mathrm{in}}} \tag{7}$$

After designing a specific plant, part-load performance can be determined. For this, design specific component parameters are calculated in the design case, for example: the area independent heat transfer coefficient kA of heat exchangers. The heat transfer at a different operation point is calculated from the kA value and the logarithmic temperature difference $\Delta\vartheta_{\log}$ in equation 8.

$$0 = \dot{Q} - kA \cdot \Delta\vartheta_{\log} \tag{8}$$

In general, the design parameters (kA in case of the heat exchanger) can be adjusted using lookup table functions to match the model behavior to measured data. This is especially useful if components with well known characteristics should be implemented in a different plant or at different operating conditions. To further improve the representation of an actual plant, the modular structure of TESPy easily allows the addition of new equations and characteristic functions to existing components or even new components.

Previous Implementations

The core strength of TESPy lies in the generic and component based architecture allowing to simulate technologically and topologically different thermal engineering applications.

For example, as the increasing share of renewable energy sources to mitigate climate change will result in a significant storage demand, underground gas storage is considered a large scale energy storage option (International Energy Agency - IEA, 2014; Pfeiffer, Beyer, & Bauer, 2017). Due to the feedback regarding the physical parameters of the fluid exchanged between the geological storage and the above-ground plant, an integrated simulation of the storage and the power plant is necessary for a detailed assessment of such storage options (Pfeiffer, Witte, Tuschy, & Bauer, 2019). Another important task in energy system transition is renewable heating: Heat pumps using subsurface heat to provide heating on household or even district level show analogous feedback reactions with the underground. As their electricity

consumption highly depends on the heat source temperature level, simulator coupling provides valuable assessment possibilities in this field, too. Additionally, TESPy has been coupled with OpenGeoSys (Naumov et al., 2020) for pipeline network simulation of borehole thermal energy storage arrays (Chen, Witte, Kolditz, & Shao, 2020).

Acknowledgments

This work is supported by University of Applied Sciences and the Center for Sustainable Energy Systems in Flensburg. It is part of the open energy modeling framework (oemof), most known for its software package oemof.solph (Hilpert et al., 2018). Many thanks to all contributors.

Key parts of TESPy require the following scientific software packages: CoolProp (Bell et al., 2014), NumPy (Walt, Colbert, & Varoquaux, 2011), pandas (McKinney, 2010). Other packages implemented are tabulate and SciPy (Virtanen et al., 2020).

References

Baehr, H. D., & Stephan, K. (2016). *Thermodynamik*. Springer Berlin Heidelberg. doi:10.1007/978-3-662-49568-1

Bell, I. H., Wronski, J., Quoilin, S., & Lemort, V. (2014). Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library coolprop. *Industrial & Engineering Chemistry Research*, *53*(6), 2498–2508. doi:10.1021/ie4033999

Casella, F., & Leva, A. (2003). Modelica open library for power plant simulation: Design and experimental validation. *Proc. 3rd Modelica Conference*, 41–50.

Chen, S., Witte, F., Kolditz, O., & Shao, H. (2020). Shifted thermal extraction rates in large borehole heat exchanger array a numerical experiment. *Applied Thermal Engineering*, *167*, 114750. doi:10.1016/j.applthermaleng.2019.114750

Epple, B., Leithner, R., Linzer, W., & Walter, H. (Eds.). (2012). *Simulation von Kraftwerken und Feuerungen*. Springer Vienna. doi:10.1007/978-3-7091-1182-6

Herning, F., & Zipperer, L. (1936). Calculation of the viscosity of technical gas mixtures from the viscosity of the individual gases. *Gas-und Wasserfach*, *79*, 69–73.

Hilpert, S., Kaldemeyer, C., Krien, U., Günther, S., Wingenbach, C., & Plessmann, G. (2018). The open energy modelling framework (oemof) - a new approach to facilitate open science in energy system modelling. *Energy Strategy Reviews*, *22*, 16–25. doi:10.1016/j.esr.2018.07.001

International Energy Agency - IEA. (2014). Technology roadmap - energy storage. https://www.iea.org/reports/technology-roadmap-energy-storage.

McKinney, W. (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). doi:10.25080/Majora-92bf1922-00a

Naumov, D. Y., Fischer, T., Bilke, L., Rink, K., Lehmann, C., Watanabe, N., Wenqing, et al. (2020). Ufz/ogs 6.2.2. Zenodo. doi:10.5281/zenodo.591265

Pfeiffer, W. T., Beyer, C., & Bauer, S. (2017). Hydrogen storage in a heterogeneous sandstone formation: Dimensioning and induced hydraulic effects. *Petroleum Geoscience*, *23*(3), 315–326. doi:10.1144/petgeo2016-050

Pfeiffer, W.-T., Witte, F., Tuschy, I., & Bauer, S. (2019). Test cases for a coupled power-plant and geostorage model to simulate compressed air energy storage in geological porous media. *International Conference on Applied Energy, 12.-15.08.2019, Västeras*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*, 261–272. doi:10.1038/s41592-019-0686-2

Walt, S. van der, Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30. doi:10.1109/mcse.2011.37

Witte, F. (2020). TESPy. https://tespy.readthedocs.io/.

Politecnico di Milano. (2014). ThermoPower. https://build.openmodelica.org/Documentation/ThermoPower.html.