The Journal of Open Source Education

ThermoState: A state manager for thermodynamics courses

Bryan W. Weber[1]

1 Department of Mechanical Engineering, University of Connecticut, Storrs, CT USA 06269

DOI: 10.21105/jose.00033

Software

- Review
- Repository
- Archive

Submitted: 22 September 2018
Published: 24 October 2018

Summary

ThermoState is a Python package that provides easy management of thermodynamic states of simple compressible systems. ThermoState relies on CoolProp (Bell, Wronski, Quoilin, & Lemort, 2014, 2016) and Pint (Grecco & others, 2017) to provide the equations of state and units handling, respectively. ThermoState replaces tables that are typically used in engineering courses to evaluate properties when solving for the behavior of systems.

Statement of Need

In traditional engineering thermodynamics courses, properties of simple compressible systems (such as the temperature, pressure, specific volume, specific enthalpy, etc.) are evaluated from tabulated data. This process often involves an inordinate number of arithmetic calculations, simply to determine the appropriate properties at a given state. The length of the simple calculations inhibits students' ability to recognize patterns in problems; they get lost in the calculations, and fail to see the larger point of a problem. Aside from calculations from a table, students also often get stuck performing unit conversions, especially in the so-called "English Engineering" (EE) units system.

Therefore, there is a need for a software package that simplifies or reduces the number of rote calculations the students must accomplish and removes the burden of improper unit conversion. Existing software packages that solve the equation of state for a substance have APIs that are confusing for students who are learning not only thermodynamics, but Python as well. In addition, packages such as CoolProp (Bell et al., 2014, 2016) do not have a facility to automatically manage units and require all quantities to be in base SI units, which is inconvenient for problems working with EE units.

ThermoState combines the CoolProp (Bell et al., 2014, 2016) and Pint (Grecco & others, 2017) packages to enable easy evaluation of equations of state without needing tables as well as automatic unit conversion. In addition, ThermoState has an easy to understand API that uses instance attributes to store the properties of interest.

Because of the simplicity of use and because students do not have to perform trivial arithmetic, ThermoState also enables students to engage in higher levels of learning by evaluating the performance of systems for a range of input parameters. Packages such as NumPy (Oliphant, 2015) and Matplotlib (Hunter, 2007) can be used to generate ranges of input parameters and plot the output. Students can interpret the plots and understand the behavior of systems for a range of input values. This type of analysis would not be possible with traditional table-based techniques because of the sheer number of calculations required.

Functionality and Usage

The primary interface for students to the ThermoState package is the `State` class. The `State` class constructor takes one mandatory argument, the name of the substance, and two optional keyword arguments, which must be a pair of independent, intensive properties. Two independent, intensive properties are required to solve the equation of state of the substance. These keyword arguments must be instances of the `Quantity` class from Pint (Grecco & others, 2017), and must have the appropriate dimensions for the property being set.

```
from thermostate import State, Q_
T_1 = Q_(100.0, 'degC')
p_1 = Q_(0.5, 'bar')
st_1a = State('water', T=T_1, p=p_1)
```

Alternatively, only the name of the substance can be specified, and the state can be fixed by assigning a value to an instance attribute representing the appropriate pair of properties.

```
st_1b = State('water')
st_1b.Tp = T_1, p_1
```

Once the instance of the `State` class is created, the properties of the substance are available as attributes of the instance. These instance attributes are themselves instances of the `Quantity` class from Pint (Grecco & others, 2017). Internally, the instance attributes are always in SI base units, to simplify passing arguments to the appropriate CoolProp (Bell et al., 2014, 2016) functions. However, the `Quantity` class has a `to` method that converts the units of the quantity, allowing the students to use whatever units are natural for a problem.

```
v_1 = st_1a.v.to('m**3/kg')
u_1 = st_1a.u.to('BTU/lb')
```

Finally, Pint (Grecco & others, 2017) `Quantity` instances can be used in arithmetic statements in Python, and provided that the statement is dimensionally consistent, can be easily used to calculate, e.g., the work and heat transfer during a process.

Recent Uses

I originally wrote ThermoState to use in my Applied Thermodynamics course in the Spring 2016 semester at the University of Connecticut. This course covers the major thermodynamic cycles (Rankine, Brayton, Refrigeration, Otto, Diesel, etc.). Having the students use ThermoState to evaluate properties allowed me to assign problems that would have been impossible without some software assistance. In particular, I expect the students to produce 2-3 design reports of systems that implement one or more of the previously mentioned cycles, and I expect students to perform some level of analysis of the cycle with respect to varying the input parameters.

Since then, I have used ThermoState in two other Applied Thermodynamics courses (Spring 2017 and 2018). I have also used ThermoState in my Thermodynamic Principles course, a prerequisite for Applied Thermodynamics. In all the courses, students had

positive feedback about the use of ThermoState, strongly preferring using the software to using tables.

To avoid having to have students install any software on their personal computers, I use a JupyterHub instance hosted by the University where students can log in and work on their homework and projects. Using Jupyter Notebooks (Kluyver et al., 2016), students can combine their code to solve the problem with Markdown and equations that explains their process.

A tutorial on the basic use of the package and several examples can be found in the `docs` folder of the repository as well as the online documentation.

References

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

Bell, I. H., Wronski, J., Quoilin, S., & Lemort, V. (2016). *CoolProp*. Retrieved from http://coolprop.org

Grecco, H. E., & others. (2017). *Pint*. Retrieved from https://pint.readthedocs.io

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

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., et al. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. (F. Loizides & B. Schmidt, Eds.). IOS Press.

Oliphant, T. E. (2015). *Guide to NumPy* (2nd ed.). CreateSpace Independent Publishing Platform.