Teaching the foundations of thermodynamics with PYro

Christopher R. Martin
Penn State University
Altoona Campus
Altoona, PA 16601
Email: crm28@psu.edu

Jacob P. Moore
Penn State University
Mont Alto Campus
Mont Alto, PA 17237

Joseph A. Ranalli
Penn State University
Hazelton Campus
Hazelton, PA 18202

***Abstract*—One of the key skills developed in foundational thermodynamics courses is obtaining property and state data for various substances of interest. Typically, students are instructed to perform this task through the use of tables or computer software. In this paper, we present and evaluate modules for teaching the foundations of thermodynamics using free open-source software intended to port to students' professional lives. The approach introduces the PYro thermodynamic property calculator. PYro is implemented in Python, which is free and available on most widely used platforms. PYro is clearly documented, and all data are readily traceable to reputable sources. Most of the data describe ideal gases from the NIST JANAF database, but there is also support for mixtures (such as air) and multiphase substances (such as steam). The interface design makes the software appropriate to most tasks in introductory and intermediate thermodynamics courses without requiring proficiency in the Python language. While the idea of using software to teach thermodynamics is far from new, commercial software usually comes at a substantial price and places the implementation burden on the instructor. On the other hand, educational software rarely transitions into students' professional lives. This paper proposes a model for productively separating the development of skills (like table look-ups) from knowledge and concepts.**

In addition to an introduction of the tool, this paper provides results of preliminary evaluation conducted within a thermodynamics classroom. The authors developed a learning module demonstrating the use of PYro to compute states for an ideal Brayton cycle. Students were tasked with performing parametric analysis on the cycle, by varying various limiting factors (e.g. combustor pressure, turbine inlet temperature). Students were asked to compare power produced and cycle efficiency computed under these conditions. At the end of the module, students were surveyed about the experience of working with the software. Evaluation is provided in the form of instructor and student feedback from a classroom implementation. We propose that this utilization of the tool demonstrates its ability to promote higher-level cognitive thinking in problem solving, removing the time intensive task of performing table look-ups and allowing them to focus on more holistic questions of cycle performance.

I. Introduction

Instruction in thermodynamics usually requires blending challenging abstract ideas with skills-based topics like interpolation and table look-ups. The inherently multi-dimensional interdependent nature of properties has long made thermodynamics notoriously challenging for engineering students, and the problem is often exacerbated when simultaneously applying the ideas to cycles, psychrometrics, combustion, heat transfer, and other applications common to engineering fields.

We support the idea that by shortening the time it takes to answer the question, "I wonder what happens when…" and by empowering students to answer those questions themselves, it is possible to economically make exploration a part of learning thermodynamics. For example, precious few students are ever motivated enough to dive into property tables to see what a jet engine will do at different altitudes. When posed the question with a tool for answering the question in seconds, we create an environment where students are more likely to ask and answer their own questions beyond what we ask. If their tools also port to their professional lives, then the skills they develop will also have been well invested.

For professional applications, there are countless for-profit packages available with their own specialties, and a number of thermodynamic texts come with calculators of their own. Unfortunately, the professional grade packages seem to be rarely used in education, and educational packages seem to be similarly absent in industry. If a software package is likely to span the needs of both educational and industrial institutions, it must simultaneously be robust, inexpensive, intuitive, and reputable.

Certainly, there are existing for-profit models for offering expensive software at a discount to academic institutions to make such an arrangement possible. However, we focus our attention on software that is "free and open." The GNU Operating System project, sponsored by the Free Software Foundation, currently defines free software as providing the user four freedoms; to use for any purpose, to study and change the software, to distribute copies of the software, and to distribute modifications of the software [1]. This model frees people to modify code to suit their own needs, but also creates the possibility for communities wherein living copies of software can evolve over time.

A. Background

The use of software to help teach thermodynamics is certainly not new. Previous research on the potential benefits of software in education has focused on proprietary compiled codes distributed with text books [2], [3]. Some recent efforts towards specialized free educational software [4], [5] do exist.

Karimi, in particular, warns against treating these systems as "black boxes," lest students learn dependence on the computer for verbatim answers without seeking any deeper insights. Instead, he advocates (as do we) for the integration of software in education in such a way as to empower students to discover the "fundamental physical laws" too often inaccessible to students while they grapple with tables and interpolation. We go further, and argue that students benefit more if they can use the same tools in a professional role that they were using when they first studied the topic.

It is not apparent that even the proprietary codes long distributed with widely used texts have achieved any level of popularity among industrial users. A brief survey of thermodynamic software packages available seems to suggest a few classes of packages; calculators for materials science, chemical reaction calculators, and property calculators. The calculators for materials science focus heavily on predicting crystalline morphology in alloys. Chemical reaction calculators tend to focus more heavily on gas phase properties and chemical kinetic data. Property calculators simply report properties of substances without design for a specific application. While we make some effort to describe the current state of the art, we do not give an exhaustive survey of available packages here. If we did, it would almost certainly exceed eight pages, and may be out of date by the time this document were printed.

There are several mature competing codes for materials science applications that have been available before the start of the millennium [6], [7], [8], [9], [10], [11], [12], some of which are reported to be in wide use today. These codes focus largely on phase equilibrium calculations relevant to crystallography and metallurgy. They are of particular value to the material science community because they predict phase diagrams for multi-component systems.

Chemical reaction calculators date back at least to the 1980s; notably including W. C. Reynolds's Fortran-based STANJAN code [13], the proprietary Chemkin code, and the popular open-source Cantera project they inspired. These codes focus largely on chemical equilibrium calculations in gas-phase reactions (combustion), but their scope has broadened with time to include multiphase phenomena like surface interactions and soot formation.

We classify software that retrieves information about substances, agnostic to the purpose for which it is used, as "property calculators." Of these, there are many; often focusing on a particular class of substances or properties. NIST maintains extensive databases that fit into this category, notably the proprietary NIST-ASME implementation of the properties of steam [14], web-based JANAF tables [15] from which many of PYro's data are taken with permission, and web-based atomic spectra database [16]. We further narrow our focus to software that fits in this category.

We conclude from this preliminary survey that there are excellent packages already available, but many of them are proprietary or highly specialized. We therefore shift our attention to portable packages appropriate for use in both industry and education with a premium on flexibility. We imagine a platform capable of accessing a wide variety of properties of substances and mixtures from incompatibly formatted sources, but with a standard interface.

TABLE I
REQUIREMENTS FOR PROPERTY SOFTWARE FROM EDUCATION AND PROFESSIONAL APPLICATIONS

Requirements for Education	**Requirements for Industry**
Streamlined standardized interface for all data formats	Traceability of original source data to reputable sources
Minimal programming proficiency required	Detailed documentation on all features
Documentation available "in-line"	Regular support and maintenance across system upgrades
Actively maintained modules and examples	Multi-platoform support (e.g. Windows, Linux, Mac-OS, etc…)
Tools for automating challenging activities	Wide variety of data (e.g. fluid, thermochemical, electrical)
Free and open	Tools automating the most common activities

In Table I, we propose a brief list of requirements for such a system were it to serve the needs of both professionals and students. Python is a natural choice for a language in which to provide this kind of service. It is free, widely used in a number of scientific communities, and uses syntax that translates naturally from other languages. A number of proprietary software packages already widely adopted by practitioners are designed with a Python interface (e.g. Matlab, Labview, ANSYS/Fluent). Additionally, Python has excellent means for providing "in-line" documentation; it has support for objects that users can query for documentation at the command line without needing to look up additional resources.

There are currently several thermodynamics codes distributed for Python; perhaps most notably Cantera, CoolProp, and Thermopy. As we have mentioned, Cantera is a mature, actively maintained, widely used, and well documented system for combustion modeling. Thermopy offers steam properties, some ideal gas properties, and psychrometric data in separate modules, and was last updated in 2009. The most applicable to the present topic is CoolProp, which includes multi-phase thermodynamic data for 118 species. PYro distinguishes itself from these alternatives in its capacity for stringing together incompatible data sources with a standard "pythonic" interface. Cantera and CoolProp are by far the most capable packages, but they offer python API hooks into compiled libraries, which can limit the elegance of the interface. That withstanding, the value of these packages is not to be minimized; depending on what instructors are trying to achieve, these may be excellent alternatives.

This is the motivation for the PYro project. What we present here is implementation in a classroom of a young incarnation of the project. The software is intended to grow over time with

the help of students, teachers, and practitioners.

II. PYro Package

The PYro package was released in November of 2015, and has already seen its first update. It is a living project, and there are plans to expand it continuously. That established, old versions of the code are published in perpetuity.

A. Approach

When composing a property calculator, the author must first decide how the data are to be formatted. Will the software interpolated between tabulated values, will it use an equation of state whose coefficients are listed for each substance, or will there be some more sophisticated manner of curve fit? When attempting to knit together data of different types from different sources, this challenge becomes quite impossible without substantial effort.

PYro makes no attempt to store data in a unified format. Instead, the package is organized into individual classes (like `igfit` and `igtab`) that tend to their own data, which need be nothing alike (like curve fit and tabulated data). For the most part, the user never needs to be aware of that distinction. Each object provides its own methods (or functions), and offers documentation on how to use them. As a result, there is no need to realize that thermochemical data for diatomic oxygen is stored as a curve fit, xenon is tabulated explicitly, air only knows what its made of (and nothing more), and steam is a more complicated matter.

B. Interface

To gain access to a substance's properties, users request an object representing that substance. At the command line, this may look like the following:

```
>>> air = pyro.get('air')
```

The object now stored in the variable named 'air' knows everything it needs to retrieve the properties of air. Properties are calculated as a function of temperature and pressure. The interface allows users to call out temperature (T) and pressure (p) explicitly by name or simply pass them in order like in a traditional function call. Here, we calculate the enthalpy (h) and specific heat (c_p) of air at 450K and 1.47bar.

```
>>> air.h(T=450., p=1.47)
149.49651484733164
>>> air.cp(450., 1.47)
1.0226922187797265
>>> air.cp()
1.0034916652356838
```

In the last example, no arguments are given, so PYro defaults to standard values for temperature and pressure (300K, 1.013bar). The interested user can reconfigure those numbers. All of the properties are standardized to a kJ, kg, s, K, bar system. These units were chosen to be mathematically intuitive, while producing conveniently sized numbers for most applications.

Of course, not all substances are the same. For example, air is a mixture, so in addition to its thermophysical properties, users can find out what its constituents are. A mass fraction (Y) is the portion of a mixture's total mass contributed by each of the constituent substances. When we call air's mass fraction function, we retrieve a Python object that names each of the component gases and its mass fraction.

```
>>> air.Y()
{u'Ar': 0.012895662694107133,
 u'CO2': 0.00047710404152028824,
 u'N2': 0.7552050549294919,
 u'O2': 0.23142217833488057}
```

Here, we see that by mass, air is 1.3% argon, .05% CO_2, 75.5% N_2, and 23.1% O_2. The `air.X()` method gives the same constituents in volumetric fractions. Pure substances like oxygen or steam have no need for `Y()` or `X()` methods.

On the other hand, substances that include multiphase data offer up some special options to help users characterize the phase changes.

```
>>> steam = pyro.get('steam')
>>> steam.h(T=450., p=1.47)
2827.075794818073
>>> steam.hs(T=450.)
(749.29333968000344, 2774.4101890593283)
>>> steam.h(T=450., x=0.5)
1761.851764369666
```

Here, we see the enthalpy of steam at 450K and 1.47bar, the saturation enthalpies (liquid and vapor) of steam at 450K, and enthalpy of 50% quality steam at 450K. Gas phase data have no need for `hs()` methods since they do not saturate.

```
>>> steam.triple()
(273.16, 0.00611657)
>>> steam.critical()
(647.096, 220.64)
```

PYro can also report the very special conditions at which water exists in three phases (the triple point), and the point above which the liquid-vapor phase transition disappears (the critical point).

All data come with detailed citations provided by the `info()` function. For example,

```
>>> pyro.info('steam')
```

returns a printout that identifies the data class used, the source file's location on the hard drive, the date it was last modified, and the following text:

> "Multi-phase steam curve fits are taken from the IF-97 report http://iapws.org/relguide/IF97-Rev.html Maintained by the International Association for the Properties of Water and Steam, the Industrial Formulation of 1997 provides curve fits for precisely calculating the properties of water and steam. Detailed citations for original data may be found on the IAPWS website.

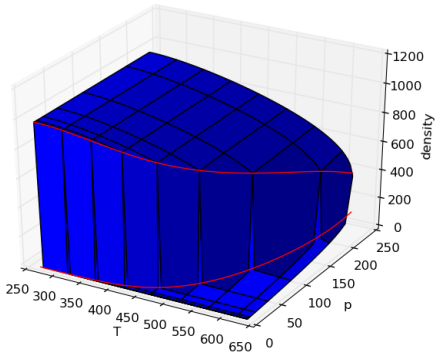

Fig. 1. Density of steam across temperature and pressure; the saturation lines are in red.

> All properties have been validated against the tests recommended by the IF-97 report, the exception being constant-volume specific heat. It was validated by first validating internal energy and numerically differentiating it with respect to temperature at constant volume."

C. Capabilities and Limitations

As of version 1.2, PYro contains 79 substances. Most of those are gas-phase data taken from the JANAF tables or curve fits of those same data. There are also mixtures of gases like air, H35 (a blend of hydrogen and argon), and F5 (a blend of nitrogen and hydrogen). Steam is the first multiphase data type available in PYro with more planned in future releases.

All substances have a basic standard data set defined; specific heats, enthalpy, entropy, molecular weight, and density. Some substances also define additional data like we discuss above. Using tools already available in PYro and sample codes available on PYro's website [17], it is possible to generate multidimensional plots like Figure 1 that students can rotate and manipulate.

The problem often arises that users need to use the properties in reverse. For example, there are a number of problems that require students to find a temperature given pressure and entropy. Version 1.2 also introduced the `psolve()` function, which returns temperature and pressure given any two properties.

```
>>> air.s(T=450.)
7.1115470914719481
>>> air.h(T=450.)
149.49651484733164
>>> air.psolve(s=7.11155, h=149.4965)
(449.99998548211147, 1.0132394288744628)
```

While `psolve()` works quite well with gas data, it is not entirely stable when used with steam. The discontinuity due to the phase change can cause convergence issues. This issue is intermittent, is well understood, and future releases will resolve the problem.

PYro is primarily limited by its data set. Version 1.2 includes no solid phase data, and all of the data are thermophysical. Future releases will add new substances like refrigerants, and electro-mechanical data like thermal conductivity, viscosity, electrical conductivity, permeability, and others.

There are plans to create amalgamated data classes so that different data classes that describe the same substance can be knitted seamlessly together. For example, a data class that describes the paramagnetic properties of oxygen might be jammed together with the thermodynamic data, so that users would still see only a single object representing the single substance.

III. Educational Pilot Testing

The first educational implementation of PYro was in a small thermodynamics class. Version 1.2 of the PYro software was used in a two day lab activity examining the effect of input conditions on the power and efficiency of an ideal Brayton cycle engine. The pilot was conducted in an class with three sophomore engineering students in a small public institution. Prior to the lab, the instructor had used a combination of lookup tables in the course textbook and the web-based NIST database for material properties. The instructor and all students had no prior coding experience with Python. After the experience, students commented on their experience with the software on a survey.

A. Description of the Activity

In the first phase of the lab, the instructor provided a basic shell code with some setup code, initial conditions, and basic comments to guide students in the overall process they needed to follow to find engine power and efficiency. The students were introduced to the Brayton cycle one class period prior to the lab. They had some basic familiarity with the cycle, and had calculated these quantities once before using lookup tables for air in the textbook. Students guided the process discussion, and the instructor guided the students in the required syntax. By the end of the first fifty-minute section, all students had successfully determined the engine power and cycle efficiency for the given input conditions.

On the second day of the lab, the instructor gave the students a pre-built code chunk to create a T-s plot for the cycle. With about ten minutes of debugging, the students were able to see a T-s plot for their cycle with input conditions and outputs notated on the plot. An example of this plot is shown in Figure 2. After students had successfully created the T-s plots, the instructor asked students to begin playing with the inputs (input temperature, pressure, mass flow rate, post combustor temperature, and pressure ratio) and to observe the results on power, efficiency, and the shape of the T-s plot. When using the air tables, determining the impact of such a change would take 20-30 minutes of calculations, but with the code already developed, determining the impact of the change took only seconds (changing the input and then re-running

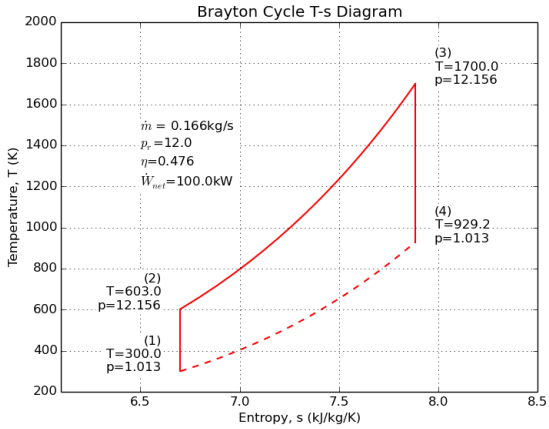

Fig. 2. Temperature-entropy process diagram for a Brayton cycle, generated automatically using a script built on PYro.

the code). After being given some unstructured time to play with these variables, students were asked which inputs did not have a significant impact on power or efficiency (such as input pressure), which impacted engine performance in some ways but not others (such as mass flow rate impacting power but not efficiency) and which impacted both outputs (such as input temperature and pressure ratio). At the conclusion of the lab a short survey was distributed to gather feedback from the students on their experiences with the software.

B. Summary of Survey Results

This pilot was designed to identify any problems in the software and to solicit next steps for documentation, examples, and features. The pilot group consisted of three students, allowing highly individualized reflections rather than substantive empirical conclusions. Students were prompted with two seven-level Likert scale questions:

1) "I feel that the lab helped me better understand the Brayton cycle."
2) "I feel that the PYro software is useful in learning about thermodynamics."

with options 1:Strongly Disagree, 2:Disagree, 3:Mildly Disagree, 4:Neutral, 5:Mildly Agree, 6:Agree, 7:Strongly Agree. Students were also prompted with three free response questions:

3) "What features of PYro do you feel were most useful?"
4) "What features of PYro do you feel could be improved?"
5) "What would you change about the Brayton lab?"

All three students responded with a six (6) to question 1. Two students responded with a seven (7) to question 2, and one student responded with a six (6).

Responses to question 3 universally focused on the ease of obtaining property values quickly and the ability to change parameters of the analysis. Since the exercise focused on that specifically, these comments are not particularly surprising, but the degree success in achieving the perception of ease of use in novice users is an unexpected positive result.

Responses to question 4 included a call for better in-line documentation and functionality already offered in the `psolve()` function. The third student offered no suggestions. In-line documentation is ideal for the intent of the PYro package, since it makes learning through exploration feasible. We interpret the comment about `psolve()` as an implicit indication for the need for a more intuitive way for users to browse the package's capabilities.

In response to question 5, one student reported being satisfied with the lab as it was. Both of the remaining students actually asked for more complexity. One student wanted to learn enough to construct the scripts "from scratch," and the other student wanted to go into more detail modeling the cycle.

IV. Conclusions and Recommendations

Instructors wishing to consider implementing PYro in their course should visit the website [17] and explore whether PYro is appropriate to the course. There is a good chance that it will work well for topics in combustion, heat transfer, cycles, and especially foundational thermodynamics, but that will depend strongly on precisely what is being covered and how.

The responses to question 5 underline an inherent challenge in using activities like these; students who are inspired to dig deeper may not always be motivated in compatible directions. Some students will want to master the programming aspects, other students may be fascinated by the thermodynamics, and other students may simply be satisfied to complete the assignment and move on. That is certainly not a bad thing, so we recommend including additional materials to allow more motivated students to self-educate. There are excellent online tutorials for Python [18] that are regularly updated.

Students and instructors without prior experience in Python can certainly succeed with the package, but that will always limit what can be done comfortably. There is a slowly expanding set of example codes on the PYro website that might be enough for introductory needs. To broaden the audience that will find PYro useful, there are a number of future activities planned:

- Matlab has built-in Python support. Attempts to elegantly integrate PYro with Matlab are planned, but have not yet begun.
- The data set should be broadened to include materials most common to undergraduate courses; refrigerants, liquids, solids, etc.
- Example codes and modules should be broadened and simplified to ease integration for non-Python users.
- `psolve()` should be improved for stability in multi-phase systems.
- Example integration in larger classes should provide broader data sets.
- PYro's distribution should be ported to the built-in `pip` Python packaging system to ease installation and improve visibility on the web.

As a young package, PYro's progress is dependent on developing a community of users and accumulating user feedback. We implore users to contact us with their experiences; good or bad. If PYro is insufficient for a task, we encourage users to contact the author or post ideas on PYro's discussion page on new features. In the meantime, we also encourage users to consider some of the other excellent work represented by the CoolProp and Cantera packages.

REFERENCES

[1] GNU Operating System. (2016) What is free software? [Online]. Available: https://www.gnu.org/philosophy/free-sw.html

[2] A. Karimi, "Use of interactive computer software in teaching thermodynamics fundamental concepts," in *International Mechanical Engineering Congress and Exposition*. ASME, Nov 2005.

[3] N. Mulop, K. M. Yusof, and Z. Tasir, "A review on enhancing the teaching and learning of thermodynamics," *Procedia Social and Behavioral Sciences*, vol. 56, pp. 703–712, 2012.

[4] A. Martin, M. D. Bermejo, F. A. Mato, and M. J. Cocero, "Teaching advanced equations of state in applied thermodynamics courses using open source programs," *Education for Chemical Engineers*, vol. 6, no. 4, pp. 114–121, Dec 2011.

[5] B. Golman, "Transient kinetic analysis of multipath reactions: An educational module using the ipython software package," *Education for Chemical Engineers*, vol. 15, no. 1, pp. 1–18, Apr 2016.

[6] L. Kaufman, "Foreward," *CALPHAD*, vol. 26, no. 2, p. 141, 2002.

[7] J. Tomiska, "Extherm the interactive support package of experimental thermodynamics," *CALPHAD*, vol. 26, no. 2, pp. 143–154, 2002.

[8] H. Yokokawa, S. Yamauchi, and T. Matsumoto, "Thermodynamic database malt for windows with gem and chd," *CALPHAD*, vol. 26, no. 2, pp. 155–166, 2002.

[9] B. Cheynet, P.-Y. Chevalier, and E. Fischer, "Thermosuite," *CALPHAD*, vol. 26, no. 2, pp. 167–174, 2002.

[10] C. Bale, P. Chartrand, S. Degterov, G. Eriksson, K. Hack, R. B. Mahfoud, J. Melancon, A. Pelton, and S. Peterson, "Factsage thermochemical software and databases," *CALPHAD*, vol. 26, no. 2, pp. 189–228, 2002.

[11] R. Davies, A. Dinsdale, J. Gisby, J. Robinson, and S. Martin, "Mtdata - thermodynamic and phase equilibrium software from the national physical laboratory," *CALPHAD*, vol. 26, no. 2, pp. 229–271, 2002.

[12] J.-O. Andersson, T. Helander, L. Höglund, P. Shi, and B. Sundman, "Thermo-calc & dictra, computational tools for materials science," *CALPHAD*, vol. 26, no. 2, pp. 273–312, 2002.

[13] W. Reynolds, "The element potential method for chemical equilibrium analysis: Implementation in the interactive program stanjan version 3," Dept. Mechanical Engineering, Stanford University, Tech. Rep., 1986.

[14] National Institute for Standards and Technology. (2015) Nist/asme steam properties database: version 3.0. [Online]. Available: http://www.nist.gov/srd/nist10.cfm

[15] ——. (2013) Nist-janaf thermochemical tables. [Online]. Available: http://kinetics.nist.gov/janaf/

[16] ——. (2015) Nist atomic spectra database. [Online]. Available: http://www.nist.gov/pml/data/asd.cfm

[17] C. R. Martin. (2016) Pyro: Thermodynamic computational tools for python. [Online]. Available: http://pyro.sourceforge.net

[18] Python Software Foundation. (2016) The python tutorial. [Online]. Available: https://docs.python.org/2/tutorial/