Problem-based Learning Module for Teaching Thermodynamic Cycle Analysis using PYroMat

Dr. Christopher Reed Martin, Pennsylvania State University

Dr. Martin received his PhD in mechanical engineering from Virginia Tech, where his research focused on reduced order modelling of combustion instabilities. He worked for ESAB Welding and Cutting in plasma torch research and development, and has taught at four drastically dissimilar universities over ten years. His primary area of research is in the area of research is thermal fluid control.

Dr. Joseph Ranalli, Pennsylvania State University, Hazleton Campus

Dr. Joseph Ranalli is an Assistant Professor at Penn State Hazleton, and is the Program Option Coordinator for the Alternative Energy and Power Generation Engineering program. He previously earned a BS from Penn State and a PhD from Virginia Tech, both in Mechanical Engineering. Prior to his current appointment, he served as a postdoctoral research fellow at the National Energy Technology Lab in Morgantown, West Virginia. Dr. Ranalli's current research interests include development of tools and methods for solar energy resource assessment and the role of technology in engineering pedagogy.

Dr. Jacob Preston Moore, Pennsylvania State University, Mont Alto

Jacob Moore is an Assistant Professor of Engineering at Penn State Mont Alto. He has a PhD in Engineering Education from Virginia Tech and a Bachelors and Masters in Mechanical Engineering. His research interests include concept mapping, digital textbooks, and additive manufacturing.

Problem-based-learning module for teaching thermodynamic cycle analysis using PYroMat

Abstract

This paper details a problem-based-learning module for addressing student difficulties in thermodynamics. Using the PYroMat open source software platform to automate more basic skills (like table look-ups and interpolation), learners are called on to design a cycle to meet certain criteria. In the module we detail here, students are provided costs and operational data for sets of candidate components from which to construct a Rankine cycle. Students are provided with fuel cost and the market value for electricity, and asked to make recommendations for the system's return on investment. To perform their analysis, students are provided a Python code implementing the PYroMat package, which they modify and run to determine the performance of their design. We provide advice on implementation and resources to support the module in a sophmore-level class.

1 Introduction

In this paper, we propose a Problem-Based-Learning (PBL) module for introducing students to the Rankine cycle in a typical undergraduate thermodynamics course. While PBL has demonstrated substantial merit for motivating students to explore difficult topics, its implementation in thermodynamics is rarely practical due to the time and complexity of realistic calculations. In the proposed module, we employ the Python-based PYroMat [1] thermodynamic property package to allow students to accelerate these more basic tasks.

We select Rankine cycle analysis as a topic since it could be said to represent a worst case; requiring the evaluation (and inversion) of the thermodynamic properties of equilibrium non-ideal gas-liquid mixtures. For a novice, the use of traditional property tables is often sufficiently cumbersome to overwhelm any attempt to apply broader thinking; what makes this system useful, what limits its performance, how would new high temperature materials impact our cycle? By using PYroMat to accelerate the more foundational elements of the analysis, students are free to grapple with problems more closely related to the design of these systems.

It is important to emphasize that it is not our position that students be excused from learning basic skills (like the proper use of tables). To the contrary, we advocate that assignments be crafted to focus students' attention on developing a manageable subset of those skills at a time. This module is intended as a means to focus on the practical design of a thermodynamic cycle with the assumption that the fluid properties are already familiar.

1.1 Background

Research on the potential benefits of software in education has often focused on proprietary compiled codes distributed with text books [2, 3]. Some recent efforts towards specialized free educa-

tional 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.

To Karimi's point, it has been proven again and again that technology alone does not improve educational outcomes, it is only when instructional practices change that we see improvement [6]. When implementing a new technology into the classroom then the focus should be on instructional strategies rather than the technology itself. The technology itself is only an instructional tool that can be used effectively or ineffectively within the full instructional context. For this reason, it is always important to situate the use of a technology within a larger pedagogical strategy.

One pedagogical strategy that has shown promise is Problem Based Learning [7, 8]. In Problem Based Learning, students work in teams to address a central problem or question. The students themselves take charge of the learning process, and engage in self-directed learning as the need arises. Instructors in this system serve as facilitators, both asking questions to get students to think about critical factors and answering student questions when they arise. This strategy situates learning within a context that will mirror what engineers face in practice, and encourages a pattern of lifelong learning. In engineering education in particular, researchers have found that Problem Based Learning approaches had an advantage over more traditional strategies in terms of the development of soft skills (problem solving, teamwork, self-directed learning) without sacrificing gains in factual knowledge [9].

Though problem based learning has its advantages, it also has limitations. In thermodynamics, an obvious choice for the central problem in a PBL unit would be the design of a thermodynamic cycle. Hypothesis formation and testing is a central aspect of Problem Based Learning [7], and would involve trying out different components or conditions to examine their effect on things like power output and efficiency. The problem however is that modeling thermodynamic cycles using property tables or even databases is a very time consuming process, limiting the amount of experimentation the students can reasonably do. A tool such as PyroMat lifts some of the repetitive computational burden from the students, vastly reducing the amount of time needed to implement a change to the model and then view the results. PyroMat unlocks Problem Based Learning as a viable strategy for teaching students about cycle design in a way that was not possible before.

The authors previously implemented PYro (the predecessor of PYroMat) in a classroom [10] with positive preliminary results. One of the underlying questions was whether students and an instructor with no prior training in Python or PYroMat could implement such an educational module productively. While the original implementation was well received, it was very narrowly applied. The present work is a natural extension of those early efforts.

1.2 Approach

The rationale for selecting PYroMat as a platform for instruction is found in the intersection of a number of constraints. We argue an educational platform should be

- free or inexpensive to students,
- portable into their lives outside the university,
- maintainable with only minimal computer familiarity,
- compatible with other widely used analysis tools,
- reliable and accurate enough for professional work,
- sufficiently documented for self-training,
- sufficiently powerful for professional use,
- simple enough for novice use.

There are a number of other excellent resources that fail one or more of these constraints. Proprietary codes are likely to offer sleek and powerful interfaces, but they are expensive and will only port into students' careers if their companies happen to invest in the same software (i.e. [11]). Many open educational platforms are written with a narrow audience in mind [5, 4] and may not port well to a broader audience. Codes packaged with texts may fail most if not all of these tests. Others are extremely powerful, but highly specialized [12]. Web-based interfaces are easy to use and free, but do not interface easily with other analytics, and their usefulness can even be deliberately restricted as a means to promote a proprietary code [13].

We use the Python language as a platform to align with the mounting trend for its use in computing education [14, 15, 16], for its tremendous power in data processing and modeling, for substantial distribution and support network [17, 18], and because its interpreter is free and open. PYroMat is distributed through the Python package index, so it can be installed and maintained automatically and without knowledge of the Python system.

1.3 About PYroMat

PYroMat is a package written in the Python language and distributed through the Python Package Index [18]. It is a command-line utility for retrieving the thermodynamic properties of a wide range of substances. Version 1.4.1 includes 79 substances; including gaseous mixtures (like air) and multi-phase substances (like steam). Here, we will give an overview of PyroMat's steam properties.

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

```
>>> import pyromat as pyro
>>> steam = pyro.get('steam')
```

The object now stored in the variable named 'steam' knows everything it needs to retrieve the properties of steam. 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.

```
>>> steam.h(T=450., p=1.47)
2827.075794818073
>>> steam.cp(450., 1.47)
2.000229350330389
>>> steam.cp()
4.181097326774104
```

In the last example, no arguments are given, so PYroMat 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.

In addition to the typical thermodynamic properties like c_p, h, and s, the steam object can report its saturation properties, and it accepts quality as an argument to its properties.

```
>>> steam = pyro.get('steam')
>>> steam.hs(T=450.)
(749.29333968000344, 2774.4101890593283)
>>> steam.ss(T=450.)
(2.1089462033994271, 6.6092224288876018)
>>> steam.h(T=450., x=0.5)
1761.851764369666
```

Here, we see the the saturation enthalpies and entropies (liquid and vapor) of steam at 450K, and enthalpy of 50% quality steam at 450K. Constants describing the limits of liquid-gas phase change line are accessible with the `triple` and `critical` functions.

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

These return the temperature, pressure pair indicating the triple and critical points of steam.

For cycle design, it is essential to be able to evaluate the properties in reverse; e.g. calculate temperature given pressure and enthalpy. This operation is exposed with specific inverse relations like `T_h()`, `T_s()`, or `p_s()`. These calculate temperature or pressure given enthalpy or entropy.

```
>>> h = steam.h(T=450., p=10.)
>>> print(h)
749.328482186
>>> T = steam.T_h(h=h, p=10.)
>>> print(T)
450.02064419293373
```

The inverse methods can optionally be configured to return quality as well

```
>>> steam.T_h(h=1500., p=10., quality=True)
(453.0356323914666, 0.36601654353242896)
```

Here, we determine that a two-phase mixture of steam at 453K with quality .366 will have enthalpy 1500kJ/kg/K.

PYroMat is integrated with NumPy (Numerical Python) to seamlessly handle arrays. As a result, it is easy to things like generating an isentropic line under the dome.

```
>>> import numpy as np
>>> p = np.linspace(1.,10.,50)
>>> T,x = steam.T_s(s=3., p=p, quality=True)
```

This generates 50-element arrays of pressure, temperature, and quality data.

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 text citing its origins.

2 The PBL Module

We created a problem-based learning (PBL) module involving analysis of a simple Rankine cycle for a sophomore-level engineering thermodynamics course. The module we created incorporates

PYroMat to automate property lookups in order to allow students to focus on how operating condition variations affect the key cycle performance characteristics such as thermal efficiency and power output.

The details of the assignment are provided in Appendix A. Educators may wish to adjust the module we propose to meet their own purposes, but the spirit will be preserved so long as the key elements of PBL [8] are preserved:

- Problems are open-ended and may lack a concrete answer
- Problems serve to organize and stimulate learning
- Students work in teams
- Learning is self-directed and student-centered
- Faculty serves as a mentor or guide

The module asks students to choose from a set of components (pump, turbine, boiler and condenser) for which example performance data are provided. The example data tables included in Appendix A were created for the project using realistic numbers, but were not derived from actual components. In fact, we permit a number of simplifying assumptions to facilitate the process (like independence of efficiency on shaft speed, flow, etc...). For instructors who are so inclined, this can afford a good opportunity for a discussion about how the students might "go the next step" to model real components.

At the end of their work, students are asked to report on system parts chosen, cycle efficiency, power produced, mass flow rate, total system cost, total energy produced each year and five-year payback. Students make selections, sum the pump, boiler, turbine, and condenser component costs to calculate a system cost. The fuel cost and revenue from electricity (both specified in $ per kJ in Appendix A), allows them to compute the return on investment. This structure requires exploration for teams to be successful, but the numbers are chosen such that multiple designs can achieve the goal. To create additional incentive for exploration, the assignment indicates that there will be acknowledgment for the team with the lowest cost system able to produce a positive return on investment, and for the highest profit total.

Because students are supposed to have had no prior training in Python or PYroMat, a sample Rankine cycle code (Appendix B) is provided for the students to use as a baseline for modification. Figure 1 shows a sample output of the code. Instructors monitoring the lab should be prepared to offer a basic level of mentoring with the language. As PYroMat is used to automate the property retrieval process, students are able to rapidly swap out components, adjusting the limits of their simulated cycle. We recommend that students be given three weeks to investigate the assignment. Faculty office hours in a computer lab should be scheduled to provide assistance with Python. We allow student groups to self-select but recommend only 2-3 students per group to prevent crowding at the computer.

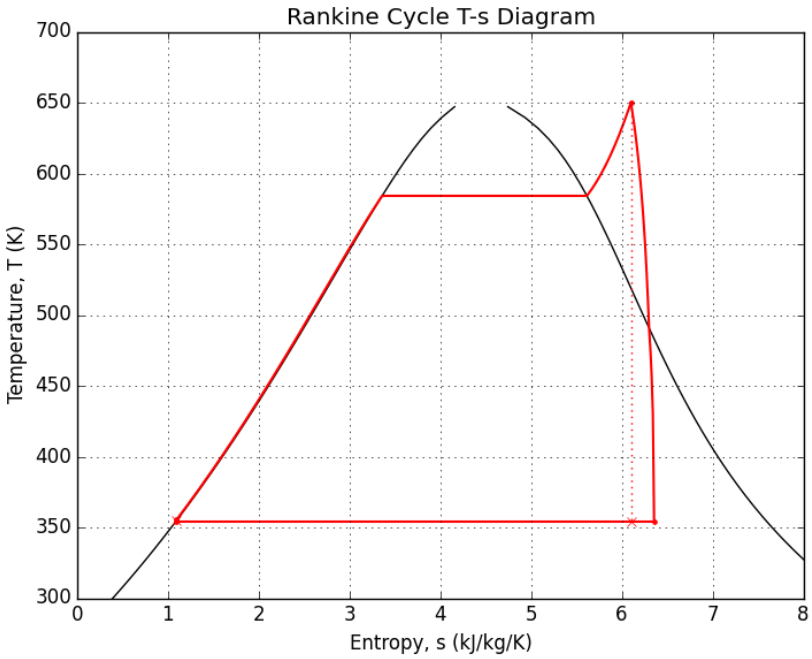

Figure 1: Example output of the module code in Appendix B.

A simple example can illustrate the analysis approach that students might consider. If students simply generate a cycle using the limitations set by the lowest cost items, the efficiency is low enough that the value of energy produced does not exceed the cost of fuel. This results in both an operating loss, as well as a failure to make up the cost of the system. Students would need to look for ways to improve the system efficiency to generate a greater value of energy produced than cost of fuel, producing an operating profit. Even for a system generating an operating profit, situations may be encountered in which the operating profit is sufficiently small that over the five year analysis period, the total profit does not cover the initial cost of the system. Students are now faced with a situation in which they could look for options to generate a larger differential between energy cost and value (improve efficiency), or increase the flow rate such that more energy is produced at the given profit per unit fuel.

Throughout the analysis, students will encounter some variables have a small effect on the cycle efficiency and power produced (e.g. pump isentropic efficiency), while other variables (e.g. pump pressure differential) can be observed to have a much greater impact on the power production. Additionally, several variables in this scheme will produce conflicting effects. For example, reducing the pump inlet pressure increases the cycle efficiency, but will also reduce the turbine outlet quality, meaning that these limitations must be considered in conjunction. On the other hand, mass flow rate acts independently of the other variables, and serves primarily to scale the operating profit produced. To reiterate the motivation for this paper, these analyses would be extremely difficult to achieve without a computerized tool to automate the property lookups.

3 Conclusions and Future Work

With this educational module, we have leveraged the PYroMat package to create opportunities for self-directed exploration that are typically not encountered in a thermodynamics course. PBL activities rely on the ability of students to apply the engineering design process: considering multiple options, investigation of relationships between variables and iteration to achieve a viable solution. The nature of typical undergraduate thermodynamics analysis, relying heavily on property lookups to perform calculations, makes iteration particularly difficult and creates obstacles to this process. By automating the property lookup part of the calculation, we enabled PBL to be employed in this thermodynamics example, and the benefits of this pedagogical approach to be brought to this topic.

Early efforts to implement PYroMat to this effect were quite narrow, and this module is the next step in broadening its implementation. There are important questions for which the authors are still collecting data; (1) can instructors and students with minimal prior training in Python and PYroMat productively use the module; and (2) is the module successful in its attempt to motivate and empower exploration? The authors intend to approach these questions by collecting instructor and student feedback surveys from instructors with both high and low levels of Python proficiency.

A Example Embodiment of the Module

Task: Design a simple Rankine cycle using only a single pump, Boiler, turbine and condenser, subject to the limitations given in the datasheets below.

Objective: Create a system that will pay off its initial investment over a 5 year period. Special acknowledgment will be given to the lowest cost system that has a positive return within 5 years, and the system with the greatest 5 year profit.

Parameters: Fuel for the boiler costs $0.01/MJ. The energy produced is worth $0.08/kWh.

Data Required: System parts chosen, operating pressures, max cycle temperature, mass flow rate, cycle efficiency, net power produced, total system cost, total energy produced each year, payback.

Table 1: Available pumps

Name	Cost ($)	Isentropic Efficiency	Minimum Inlet (kPa)	Maximum Outlet (kPa)	Maximum Flow (kg/s)
A	250,000	0.85	100	9,000	5.5
B	500,000	0.88	50	10,000	5.5

Table 2: Available turbines

Name	Cost ($)	Isentropic Efficiency	Maximum Inlet (°C)	Minimum Quality Out	Maximum Flow (kg/s)
A	250,000	0.85	600	0.8	5.0
B	500,000	0.98	650	0.8	5.5

Table 3: Available boilers

Name	Cost ($)	Heat Xfer Efficiency	Maximum Out (°C)	Maximum Press. (kPa)	Maximum Flow (kg/s)
A	250,000	0.85	625	8,000	5.5
B	500,000	0.90	700	12,000	7

Table 4: Available condensers

Name	Cost ($)	Maximum Flow (kg/s)
A	250,000	5.5
B	500,000	10

B Sample Code

```
import pyromat as pyro
import matplotlib.pyplot as plt
import numpy as np

steam = pyro.get('steam')

# Set the limit conditions based on the components
#Case A (losing)
p1Pa = 0.1e6
p2Pa = 8e6
T3 = 600
mdot = 5.0
syscost = 1000000
FuelEff = 0.85
eta_pump = 0.85
eta_turb = 0.85

#Case B (winning)
p1Pa = 0.05e6
p2Pa = 10e6
T3 = 650
mdot = 5.5
syscost = 2000000
FuelEff = 0.9
eta_pump = 0.88
eta_turb = 0.9

#Convert p to bar
p1 = p1Pa*1e-5
p2 = p2Pa*1e-5

#State 1 is a saturated liquid @ p1
T1 = steam.Ts(p=p1) #Temperature
h1,null = steam.hs(p=p1) #Enthalpy
s1,null = steam.ss(p=p1) #Entropy

#State 2s is isentropic from p1-p2
s2s = s1 #isentropic
#isentropic Temperature
T2s = steam.T_s(p=p2,s=s2s)
#isentropic Enthalpy
h2s = steam.h(p=p2,T=T2s)

#State 2 is found by applying the isentropic efficiency
Wps = h2s-h1 #Isentropic work
Wp = Wps/eta_pump #Actual work
h2 = Wp+h1 #Actual enthalpy at 2
T2 = steam.T_h(p=p2,h=h2) #Actual Temperature
s2 = steam.s(T=T2,p=p2) #Actual entropy

#State 3 is based on known p3 & T3
p3 = p2 #same pressure
h3 = steam.h(p=p3,T=T3) #Enthalpy
s3 = steam.s(p=p3,T=T3) #Entropy

#State 4s is isentropic from p3-p4
p4 = p1 #same as initial pressure
s4s = s3 #isentropic
#isentropic Temperature (including quality)
T4s,x4s = steam.T_s(p=p4,s=s4s,quality=True)
if x4s<0: #superheated vapor quality will be -1
h4s = steam.h(T=T4s,p=p4) #Superheated vapor
else:
h4s = steam.h(x=x4s,p=p4) #Liq/Vap mixture

#State 4 is found by applying the isentropic efficiency
Wts = h3-h4s #Isentropic work
Wt = Wts*eta_turb #Actual work
h4 = h3-Wt #Actual enthalpy at 4
#Actual Temperature (including quality)
T4,x4 = steam.T_h(p=p4,h=h4,quality=True)
if x4<0: #superheated vapor quality will be -1
s4 = steam.s(T=T4,p=p4) #Superheated vapor
else:
s4 = steam.s(x=x4,p=p4) #Liq/Vap mixture

#Find the work and heat transfer terms
Qhi = (h3-h2)
Qlo = (h4-h1)
Wnet = (Wt-Wp)
eff = Wnet/Qhi

#Print some cycle properties
if(x4>0):
print("Turbine Outlet Quality: {:0.2f}".format(x4))
else:
print("Turbine Outlet Quality: Superheated")
print("Efficiency: {:0.2%}".format(eff))
print("Net Power: {:0.2f} kJ/kg".format(Wnet))


# Generate some diagrams
color = 'r'
marker = '.'
smarker = 'x'
# Let figure 1 be a T-s diagram
f1 = plt.figure(1)
ax1 = f1.add_subplot(111)
ax1.set_xlabel('Entropy, s (kJ/kg/K)')
ax1.set_ylabel('Temperature, T (K)')
ax1.set_title('Rankine Cycle T-s Diagram')

# Generate the dome on both plots
Tt,pt = steam.triple()
Tc,pc = steam.critical()
T = np.linspace(Tt,Tc,50)
p = steam.ps(T)
sL,sV = steam.ss(T=T,p=p)
ax1.plot(sL,T,'k')
ax1.plot(sV,T,'k')

# Process 1-2s (isentropic compression of a liquid)
s = np.linspace(s1,s2s,20)
p = np.linspace(p1,p2,20)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color+':',linewidth=1)
ax1.plot(s1,T1,color+marker)
ax1.plot(s2s,T2s,color+smarker)

# Process 1-2 (actual compression of a liquid)
s = np.linspace(s1,s2,20)
p = np.linspace(p1,p2,20)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color,linewidth=1.5)
ax1.plot(s2,T2,color+marker)

#process 2-3 (heat add)
s = np.linspace(s2,s3,200)
p = p2*np.ones(s.shape)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color,linewidth=1.5)
ax1.plot(s3,T3,color+marker)

# Process 3-4s (isentropic compression of a liquid)
s = np.linspace(s3,s4s,20)
p = np.linspace(p3,p4,20)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color+':',linewidth=1)
ax1.plot(s4s,T4s,color+smarker)

#process 3-4 (actual expansion)
s = np.linspace(s3,s4,20)
p = np.linspace(p3,p4,20)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color,linewidth=1.5)
ax1.plot(s4,T4,color+marker)

#process 4-1 (heat rej)
s = np.linspace(s4,s1,40)
p = p4*np.ones(s.shape)
T = steam.T_s(p=p,s=s)
ax1.plot(s,T,color,linewidth=1.5)

ax1.grid('on')
ax1.set_ylim([300,800])
f1.show()

#Pause
input("Press Enter to continue...")
```

References

[1] C. R. Martin. (2016) Pyromat: Thermodynamic computational tools for python. [Online]. Available: http://pythonhosted.org/PYroMat/

[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] T. Russel, *The no significant difference phenomenon*. North Carolina State University, 1999.

[7] C. Hmelo-Silver, "Problem-based learning: what and how do students learn?" *Educational Psychology Review*, vol. 16, no. 3, pp. 235–266, 2004.

[8] H. Barrows, "Problem-based learning in medicine and beyond: a brief overview," *New Directions for Teaching and Learning*, no. 68, pp. 3–12, 1996.

[9] A. Kolmos and E. de Graaff, *Problem-based and project-based learning in engineering education*. Cambridge University Press, 2014, pp. 141–161.

[10] C. R. Martin, J. P. Moore, and J. A. Ranalli, "Teaching the foundations of thermodynamics with pyro," in *2016 IEEE Frontiers in Education Conference (FIE)*, Oct 2016, pp. 1–6.

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

[12] T. C. Project. (2017) Cantera; chemical kinetics, thermodynamics, transport processes. [Online]. Available: http://www.cantera.org/docs/sphinx/html/index.html

[13] National Institute for Standards and Technology. (2013) Nist-janaf thermochemical tables. [Online]. Available: http://kinetics.nist.gov/janaf/

[14] F. Georgatos, "How applicable is python as first computer language for teaching programming in a pre-university educational environment, from a teachers point of view?" Master's thesis, Universiteit van Amsterdam, 2002.

[15] A. Radenski, "Python first: A lab-based digital introduction to computer science," in *Proceedings of the Eleventh Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 06*, University of Bologna, Italy, June 2006, pp. 197–201.

[16] M. Goldwasser and D. Letscher, "A graphics package for the first day and beyond," in *Proceedings of the 40th Annual SIGCSE Technical Symposium on Computer Science Education (SIGCSE)*, Chattanooga, Tennessee, March 2009, pp. 206–210.

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

[18] The Python Package Index. (2017) Pypi. [Online]. Available: https://pypi.python.org/pypi