2011

More carrot than stick: Encouraging computer programming in thermal design projects

Kevin D. Cole
University of Nebraska-Lincoln, kcole1@unl.edu

Published in *Computer Applications in Engineering Education* 19 (2011); doi: 10.1002/cae.20515

Submitted February 17, 2010; accepted November 14, 2010; published online December 29, 2010.

More carrot than stick: Encouraging computer programming in thermal design projects

Kevin D. Cole

Department of Mechanical Engineering, N104 Walter Scott Engineering Center, University of Nebraska–Lincoln, Lincoln, Nebraska 68508-0656; email kcole1@unl.edu

Abstract
Students will do almost anything to avoid using thermal-property tables. In this paper, Matlab-based thermal-property software is described as an enticement for students to do computer programming in the design of thermal systems. Downloaded shareware was used for steam properties in a steam-cycle project, and an air-property package was developed for use with a gas-turbine project. Although the use of computer programming required considerable effort by both instructor and students, most students did gain a better appreciation of the utility of writing computer programs as part of engineering design. Student evaluations of the course were not significantly affected compared to semesters in which computer programming was not encouraged.

Keywords: thermodynamics, power cycle, Rankine, Brayton, thermophysical properties

Introduction

The Mechanical Engineering curriculum at Nebraska includes design content in the course Thermal Systems and Design. In the course project, students are expected to explore several different options for a thermal system (such as a power plant), decide on the best one, and report their findings in a formal written report. In these explorations, a calculation must be repeated several times with a change in only one parameter or component. Many students choose to use a computer for these calculations.

A generation ago, the only way to use a computer for engineering calculations was to write a computer program. Many of today's students, however, expect to mouse-click through every task. My experience has been that some of these students simply ignore homework assignments that require computer programming. Others will create large spreadsheets and embed hard-to-debug expressions in each cell, rather than write a short computer program to do the same task.

Technically speaking, our students know how to write computer programs, as they are required to take a three-credit programming-language course in Matlab, Fortran, or C. Practically speaking, however, many students are unwilling or unable to write computer programs. I do not fault our computer science department, whose instructors are teaching the syllabus we agreed upon. When asked, some of our thermo students tell us that they see no utility in the subject of computer programming. Why do students have this attitude?

All of our thermo students are survivors of calculus and chemistry. Many have part-time jobs, some have families, and all of them are very busy. They also communicate freely with cell phones and social networking sites. If our students see no utility in computer programming, it is because *they believe that they can succeed without it*.

If success is narrowly defined as passing a few college courses, then yes, students may get by without writing computer programs. However, we should strive to imbue our students with a loftier view of success. This is an issue of attitude, not of information, therefore it will not be addressed by outlawing spreadsheet programs. A student's attitude would be best improved by an experience of positive reinforcement associated with the desired behavior. The personal challenge, then, is to ask: what can I do in my design course to give students a positive experience with computer programming?

My answer to this question came from another trend that I have observed in student behavior. Specifically, most thermo students dislike the property tables in the back of the book. They struggle to interpret the column headings ($v_f \times 10^3$ always stumps them), they struggle when the units differ from table to table (such as pressure in bar, kPa, and MPa) and they particularly dislike interpolation. At the same time, students prefer to get their information from the internet, for ev-

erything from word definitions to the air speed velocity of an African swallow. They are masterful at getting information quickly. If a networked computer is at hand, a student will give you a word definition before you can lift the dictionary. Consequently, students see the property tables as slow and completely outdated, like a slide rule. Student dislike of the property tables was my starting point for seeking to change their attitudes about computer programming.

The purpose of this paper is to describe my experiences with encouraging students to use a programming language (Matlab) in their thermal design projects. To encourage them, I provided Matlab routines for obtaining thermodynamic properties of water and air, for use in place of paper-based tables. To discourage hand calculations, I made the projects more elaborate.

My goal was to change attitudes, so although code writing was encouraged, it was not required. A requirement would have only antagonized those students who believe that a spreadsheet is the only general-purpose computer tool they will ever need. Such students, if programming were required, would have given me a poor end-of-semester evaluation; worse, someday when they are captains of industry they will have reason to pass me over for that research contract.

This remainder of the paper is divided into sections on literature review, my experience with two specific projects, discussion, and summary.

Literature Review

In this section the literature will be reviewed in the area of Matlab use in design courses and in the area of software available for evaluating thermophysical properties.

There are many examples of Matlab software used in design courses on diverse subjects, including: robotics [1–3]; control theory [4, 5]; four-bar linkages [6, 7]; signal analysis [8, 9]; and, electrical power generation, transmission, and control [10–12]. There are also several examples of Matlab use in thermal science courses. Mawasha [13] describes a Matlab package developed for predicting the performance of heat pumps, based on manufacturer's performance data for individual components. With this approach, information on the detailed thermodynamic properties of the working fluid is not required. Ettouney et al. [14] describe a comprehensive Matlab package for design and analysis of desalination processes. Several types of desalination hardware are incorporated in the package. All required thermodynamic properties of brine, water, and water vapor are built into the package. Garcia et al. [15] used Matlab to design a set of experiments for use by third-year chemical engineering students. The Matlab program is also used by students as part of their analysis of laboratory data. Liu [16] developed courseware in Matlab intended to help beginning thermodynamic students with small steps in problem solving. The portion associated with thermodynamic properties is limited to leading students to the appropriate paper-based table, depending on the phase (liquid, mixture, vapor). One portion of the courseware can assist students with simple thermodynamic cycles. Benbouzid [17] describes a Matlab-based model of a wind turbine, developed by a student team, that incorporates a turbine, transmission, generator, transformer, and power grid. The model was used by a second group of students to build a prototype wind turbine.

The project described by Rivas et al. [18] is very close to the scope of the present work, however it is based on a spreadsheet (MS Excel) rather than Matlab. The spreadsheet, developed for classroom use, simulates a Rankine steam cycle with three regenerators. Steam properties were obtained from proprietary add-in software. The operating conditions for the steam cycle were optimized using the non-linear regression routines built into the spreadsheet.

There are several commercial Matlab packages available for obtaining thermophysical properties. The package ThermoCalc [19] provides properties for a wide range of substances. It was originally written in Fortran and C and was recently ported to Matlab, and it may be licensed as a toolbox add-in to Matlab. Thermo-Utilities [20] may be downloaded and some functions are available for free, but for complete functionality a software registration key must be purchased. Multiflash [21] is a third party software product designed for the chemical industry that provides equilibrium properties for single phase materials such as hydrocarbons, refrigerants, and gases, as well as multi-component and multi-phase streams. WatSteam [22] provides properties of water and steam, and was developed by a technical consulting firm which also offers packages for properties of other substances.

Free software, if available, is always an advantage for students. However, there are only a few free shareware packages for thermophysical properties written for Matlab. For water/steam properties I found two packages available for free download, both of which are based on the 1997 standards set by the International Association for the Properties of Water and Steam [23]. Package Freesteamplus was developed by an engineering student at a major research university. However, the supervising faculty member has no plans to support the software. Package XSteam [24] was first posted on the Matlab file exchange in 2006, and since then several user comments and reviews have indicated that it is accurate and runs quickly. I chose package XSteam for classroom use with a steam-cycle project.

For air properties, a package called airProp was not chosen because there were only two lukewarm reviews of the software since it was posted in 2006. Consequently, package IdealAir was developed under my supervision at the University of Nebraska, based on data that was originally assembled by Keenan and Kaye [25]. Package IdealAir has been posted on the Matlab file exchange [26], and was used in the gas-turbine project described in the next section. Some months after IdealAir was developed, package HOT became available on the Matlab file exchange [27], for properties of gases and gas mixtures. According to the documentation it has been under development by a research group at Virginia Tech since 2004.

Example Projects

The project in the course Thermal Systems and Design is assigned about the middle of the semester with a detailed handout, which sets the overall scope of the project. Students are expected to work on the project in groups of three, and to submit a formal written report at the end of the semester. Two distinct projects were assigned to students in successive semesters in 2009. The project assignment handouts are available from the author.

Steam-Cycle Project

The steam-cycle project asked students to design a steam-turbine system for the production of electric power. The first step is an ideal Rankine cycle with perfect (lossless) turbine,

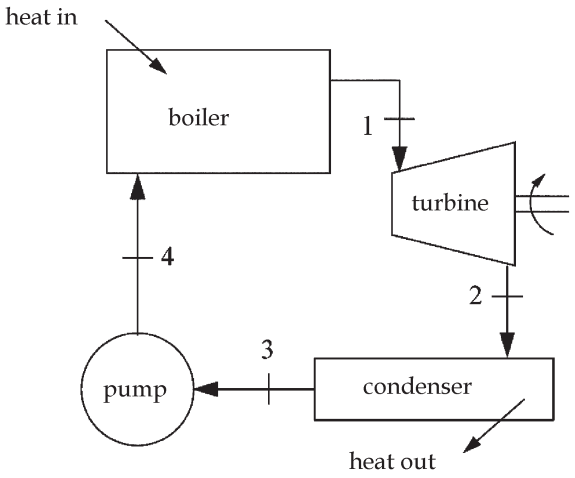

Figure 1. The ideal Rankine cycle is the starting point for the steam-cycle project.

condenser, pump, and boiler. Refer to Figure 1. Students were asked to explore how the performance and cost of the system change as real-world effects are included and as additional components are included.

To set the thermodynamic parameters for the project, students were given the following information: electric production capacity; turbine inlet conditions; cooling water temperature; and, the allowable maximum temperature of the cooling water leaving the condenser. Students were expected to choose the condenser pressure, turbine efficiency, and pump efficiency, and to justify their choices based on citations of at least two acceptable sources. Wikipedia was not an acceptable source.

Students were asked to repeatedly add a component and re-analyze the system, working up incrementally to a system with reheat and three regenerators. Extra credit was offered for analysis of up to five regenerators. For each added component, students were also asked to demonstrate that they had found the best operating pressure for that component. They usually responded by plotting component pressure versus system performance (thermodynamic efficiency). This level of complexity, representative of large real-world systems, would be too much to ask based on hand calculations alone.

Along with the project assignment, the students were given a Matlab subroutine designed to compute the thermal efficiency of a simple Rankine cycle named Rankine.m (available from the author). Subroutine Rankine.m uses package XSteam to obtain the properties of steam. Students are encouraged to alter code Rankine.m to carry out all the calculations in the project. While working with this code, students are exposed to several important programming issues, including: how to pass variables to a subroutine; how send output values to a separate file; how to use vectors to store several values compactly; and, how to use a looping structure to create a multi-row table of output values.

Gas-Turbine Project

The gas-turbine project asked students to design a gas-turbine system for producing electric power. The thermodynamic analysis begins with the ideal Brayton cycle which contains a compressor, combustor, turbine, and heat exchanger. Refer to Figure 2. Students are asked to analyze the basic cycle, and then repeatedly add a component and re-analyze the system. Additional components to consider included a reheater, regenerator, and an intercooler. Information on capital cost, maintenance cost, and fuel cost are also provided so that a cost/benefit analysis can be made at each step in the analysis. Extra credit was given for analysis of up to four intercoolers. The level of complexity was intentionally set high to encourage use of computer programming and to discourage hand calculations.

As with the steam cycle project, a starter code was provided that computed the cycle efficiency for the ideal Brayton cycle named Brayton.m (available from the author). Code Brayton.m uses package IdealAir, developed at Nebraska, for obtaining the properties of air at all states within the cycle. The expectation was that students would alter this program in order to re-analyze the cycle as more components are added.

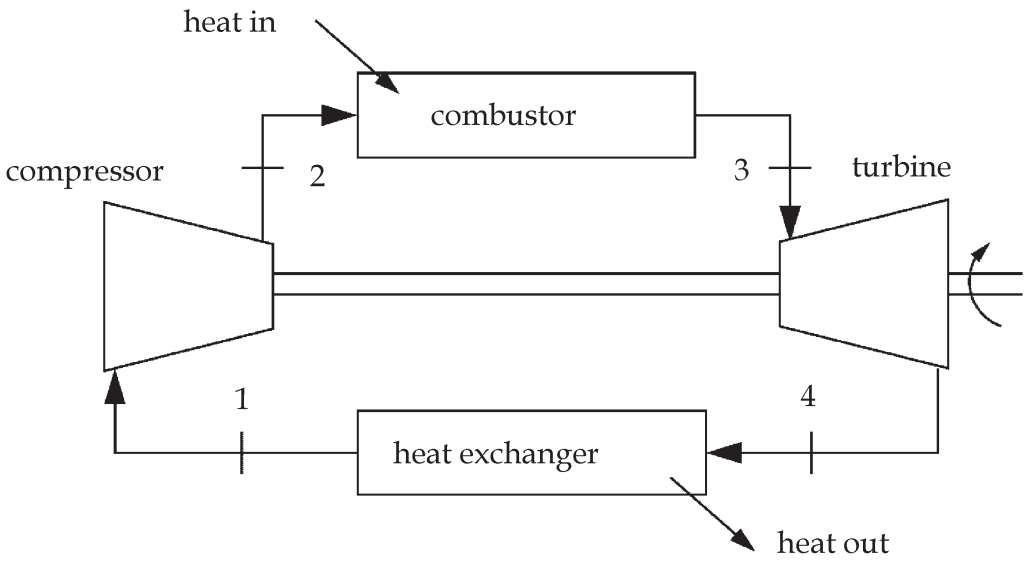

Figure 2. The air-standard Brayton cycle is the starting point for the gas-turbine project.

Table 1. Student evaluations for course thermodynamic systems and design

Semester	Responses per class	Evaluation (1 = low; 5 = high)	Statistics
Before encouraging computer programming			
Spring 2005	24/36	4.62	Average 4.52; std dev. 0.14
Spring 2007	28/42	4.59	
Spring 2008	22/36	4.56	
Fall 2008	21/35	4.31	
After encouraging computer programming			
Spring 2009	18/41	4.50	Average 4.45
Fall 2009	11/38	4.40	

For the last two semesters computer programming was encouraged. Compared to the average from the four previous semesters, the addition of computer programming had a negligible effect on student responses.

Homework to Support the Project

After the design project is assigned, weekly homework is used to guide students through the project. The project handout includes a sequence of tasks that cover some of the technical work needed to complete the project. These tasks are assigned as homework over the 6 weeks that the students work on the project. Students need to be shown that a big project must be broken down into smaller steps before any progress can be made. These homework assignments encourage students to get started early on the project and they provide timely feedback about their technical work. An added benefit of homework based on the project is that students are more likely to have the technical work completed well before formal report is due, giving them more time to write the formal report.

Discussion

The course Thermal Systems and Design is usually the student's first experience working in a team of three students on a project culminating in a formal written report. Thus, the homework based on the project is also a valuable tool for training students in technical communication. For example, I ask that every plot that is turned in has descriptive axis labels, a legend or labeled curves, and a full descriptive caption as to what is shown and what conclusions may be drawn from it. My experience in using this approach is that all of the written reports are handed in on time and most of the technical errors in student work are small ones.

I began encouraging computer programming to my project course primarily because I was looking for new project topics. Adding computer programming allowed me to assign projects more elaborate than ever before. However, adding computer programming required some additional time and effort. Finding and testing routine XSteam took some time; I had a student assistant to develop package IdealAir, but it took time to test it carefully. Extra class time was required for demonstrations of Matlab. There was more student contact time out of class (i.e., during office hours), and at times student frustration levels were high. Grading the weekly assignments based on the project was time consuming.

I chose Matlab for my class over Fortran and C because of the graphical user interface, and because my school already licenses Matlab. On the negative side, because Matlab is proprietary, students had to either work on University computers or buy the software themselves (about $100 for a student license).

So far this discussion has focused on the challenges involved, but there have also been several positive outcomes in this activity. One benefit is that I have become a more confident Matlab programmer and I have begun using Matlab in my research. In the future I intend to find or develop Matlab packages for refrigerants and for other gases so that I can develop new projects in the areas of refrigeration, gas mixtures, or combustion.

Student evaluations of the course were about the same as earlier semesters when the project contained fewer components and computer programming was not emphasized. Table 1 shows the overall student evaluations for several semesters for the same course taught by the author. The average for the first four semesters, before computer programming was introduced, was 4.52/5.0 with a standard deviation of 0.14. The last two semesters, when computer programming was included, the average was 4.45/5.0. Although this is slightly lower, it is only about one-half of one standard deviation below the previously established average. This suggests that the apparent drop in the evaluation is not statistically significant. A comment is in order on the number of student responses for each

Table 2. Verbatim comments from student evaluations from two semesters when computer programming was encouraged

The project that was assigned provided enough direction and progress checks to help us further grasp concepts.
Start the project earlier and have one task due each week. (Note by author: two tasks were due each week.)
The project was worth 15% of the grade and I put in 40+ h working on it. In comparison, an exam was worth 20% of the overall grade and I only needed to study about 10 h to do well on an exam.
My group members and me put in about 40 h apiece on the project. Although this was a nice learning experience, I wonder if it took too much time.
The project was good in theory, the only problem was the Matlab. None of my group really knew how to program in any language so we spent about 6–8 h/week trying to write the programs. This hugely affected our ability to learn anything else in the course at that time.
Doing the large group project made it feel like the things we are learning are actually relevant.

class. Student evaluations are internet based and are carried out on the student's own time, which has meant that the number of responses is always well below the enrollment.

Further information on student evaluations is given in Table 2 which shows verbatim comments entered by students for the two semesters in which the Matlab projects were assigned. Some of the comments are quite critical of the effort required for the computer programming, but some comments are supportive.

Summary

Computer programming is a general-purpose tool that all engineering students are taught, but they need encouragement to use it in new situations. Over two successive semesters I have encouraged students to use a programming language (Matlab) as part of the project in the course Thermal Systems and Design. The carrot was a computer alternative to the dreaded property tables in the book. The stick was a rigorous project with added complexity intended to discourage hand calculations. For the Rankine-cycle project, a downloadable Matlab package was available for the properties of steam. For the Brayton-cycle project, a new Matlab package was developed for air properties. Simple starter programs were also developed for each project that students could alter to do their project calculations. Each project was split into several tasks, assigned as weekly homework, to give students timely feedback on their technical progress before the formal written report was due.

Student evaluations of the course were not significantly affected by adding computer programming to the course project. This suggests that most students had a positive experience. If these students have found utility and value while applying computer programming in my course, my hope is that they will be more likely to use it again. Ideally our students will become lifelong learners as they cope with the technological changes they will surely face in the twenty-first century.

Acknowledgments — The author would like to acknowledge undergraduate assistant Jared Miller who wrote the Matlab package IdealAir and starter program Brayton.m.

References

[1] W. Khaisongkram and D. Banjerdpongchai, MATLAB based GUIs for linear controller design via convex optimization, *Comput Appl Eng Educ* 11 (2003), 13–24.

[2] S. S. Joshi, Development and implementation of a MATLAB simulation project for a multidisciplinary graduate course in autonomous robotics, *Comput Appl Eng Educ* 12 (2004), 54–64.

[3] C. Hamilton, Using MATLAB to advance the robotics laboratory, Comput Appl Eng Educ 15 (2007), 205–213.

[4] M. Koksal and S. E. Hamamci, Program for the design of linear time invariant control systems: CDMCAD, *Comput Appl Eng Educ* 12 (2004), 165–174.

[5] K. Erenturk, MATLAB-based GUIs for fuzzy logic controller design and applications to PMDC motor and AVR control, *Comput Appl Eng Educ* 13 (2005), 10–25.

[6] P. S. Shiakolas, V. Chandra, J. Kebrle, and D. Wilhite, Environment for engineering design, analysis, and simulation for education using MATLAB via the World Wide Web. II. Representative examples— System simulation and planar mechanism synthesis and analysis, *Comput Appl Eng Educ* 10 (2002), 109–120.

[7] G. Galan-Marin, F. J. Alonso-Sanchez, and J. M. Del Castillo-Granados, A methodology to learn designing optimal mechanisms for path generation, *Comput Appl Eng Educ* 18 (2010), 87–92.

[8] D. Baez-Lopez, D. Baez-Villegas, R. Alcantara, et al., Package for filter design based on MATLAB, *Comput Appl Eng Educ* 9 (2001), 259–264.

[9] E. A. Thompson, J. Acierto, and A. Chavis, Development of low-cost data acquisition hardware as an undergraduate capstone senior design project, *Comput Appl Eng Educ* 12 (2004), 198–207.

[10] I. Ngamroo, B. Somritvanitcha, and K. Hongesombut, Incorporating ObjectStab library and fuzzy logic toolbox for design of power system damping controller, *Comput Appl Eng Educ* 16 (2008), 243–255.

[11] V. Badii and H. M. Oloomi, Transmission line application MATLAB Toolbox based on the graphical design methods of the Smith chart, *Comput Appl Eng Educ* 6 (1998), 23–30.

[12] S. Tuncer, Y. Tatar, and H. Guldemir, Design and implementation of an integrated environment for real-time control of power electronic systems, *Comput Appl Eng Educ* 17 (2009), 119–130.

[13] P. R. Mawasha,A math-package assisted approach to model variables influencing heat pumps, *Comput Appl Eng Educ* 8 (2000), 104–112.

[14] H. Ettouney, H. El-Dessouky, H. Al-Fulaij, and A. Al-Ansary, Computer package for design/rating of thermal desalination processes, *Comput Appl Eng Educ* 8 (2000), 80–103.

[15] J. Garcia, R. Garcia, E. Garcia, et al., MATLAB: A powerful tool for experimental design in chemical engineering, *Comput Appl Eng Educ* 21 (2005), 676–682.

[16] Y. Liu, Development of instructional courseware in thermodynamics education, Comput Appl Eng Educ; doi: 10.1002/cae.20297.

[17] M. E. H. Benbouzid and D. Diallo, Development of a Matlab/Simulink-based wind turbine protoyping software through undergraduate student projects, *Comput Appl Eng Educ*; doi: 10.1002/cae.20375.

[18] A. Rivas, T. Gomez-Acebo, and J. C. Ramos, The application of spreadsheets to the analysis and optimization of systems and processes in the teaching of hydraulic and thermal engineering, *Comput Appl Eng Educ* 14 (2006), 256–268.

[19] Q. Chen, et al., Themo-Calc program interfaces and their applications, *Mater Sci Forum* 475–479 (2005), 3145–3148.

[20] Thermo-Utilities 3.0 for PC Matlab, http://www.taftan.com/thermoml.shtml , accessed February 1, 2010.

[21] Mulitflash for Matlab, http://www.khace.com/Products/Multiflash/index.htm , accessed February 15, 2010.

[22] WatSteam, Thermophysical properties for Matlab, http://www.khace.com/products/watsteam , accessed February 15, 2010.

[23] W. Wagner, et al., The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *ASME J Eng Gas Turb Power* 122 (2000), 150–182.

[24] M. Holgren XSteam:, Thermodynamic properties of water and steam, http://www.mathworks.com/matlabcentral/fileexchange/9817 , accessed January 30, 2010.

[25] J. H. Keenan and J. Kaye, *Gas tables.* Wiley, New York, 1945.

[26] J. Miller, IdealAir, Ideal-gas properties of air, http://www.mathworks.com/matlabcentral/fileexchange/25030 , accessed February 15, 2010.

[27] C. Martin, HOT, Thermodynamic Tools for Matlab, http://www.mathworks.com/matlabcentral/fileexchange/26430 , accessed November 10, 2010.

Kevin D. Cole is an Associate Professor of mechanical engineering at the University of Nebraska–Lincoln. His research interests include heat transfer, fluid-flow measurement, and inverse problems. He is the leading author of *Heat Conduction Using Green's Functions* (Taylor and Francis, 2010) and the companion website "Green's Function Library." He studied at Iowa State (BS 1977), University of Minnesota (MS 1979), and Michigan State (PhD 1986). Before moving to Nebraska he worked for Deere and Company (Waterloo, IA) and for Trinity College (Hartford, CT). Dr. Cole is a member of ASEE, ASME, AIAA, and Sigma Xi, and he is a registered professional engineer.