



Introducing Computational Methods to First-Year Students with Diverse Skills and Interests

By Michael R. Gustafson II, Ph.D., Duke University

Pratt School of Engineering at Duke University requires all students to take Computational Methods in Engineering during their freshman year. The goal is to give students early, practical exposure to engineering concepts so that they can better decide which field of engineering they want to pursue.

To improve the student experience and exposure to engineering, my colleagues and I have changed the course significantly in the past five years. Chief among these changes are the adoption of MATLAB® as the fundamental programming tool and the addition of hands-on data acquisition-focused labs based on Data Acquisition Toolbox. Surveys indicate that these changes have increased the students' satisfaction with the course.

About the Author

Dr. Michael R. Gustafson is Assistant Professor of the Practice, Electrical and Computer Engineering, at the Pratt School of Engineering. His focus is on developing and improving the first- and second-year undergraduate curriculum so that students are equipped with strong laboratory and computer skills. His research interests include linear and nonlinear control systems.

Teaching Challenges

The challenge we face is how to take more than 300 first-year students, teach them a new computing language, offer them a glimpse into different engineering paths, and provide them with basic skills—all while maintaining Pratt's vision of a “bold, personal, and interdisciplinary” education.

The examples used in the class and lab must be approachable by students with little background but still interesting to those who have already taken (and in some cases, mastered) the material.

The students' programming backgrounds present a different kind of challenge. In 2006, 60% of 295 students surveyed reported themselves “complete beginners” with respect to programming. The survey

revealed that while 17.8% had used C/C++, 26.0% had used Java, and 18.1% had used BASIC, only 6.3% had used MATLAB.

The relatively small number of students with MATLAB experience has worked to our advantage, as it means that almost all the students start on an even playing field. Furthermore, having students coming in with some experience in different languages sparks conversations during lecture about the similarities and differences between languages. Again, the key is to make sure that the course is challenging for the students with programming backgrounds but not unapproachable by those new to the field.

Course Objectives

In the first lecture we lay out the learning objectives, which are to give students the tools and experience necessary to:

- Interpret engineering problem statements and solve them using MATLAB
- Model physical systems and optimize parameters using iterative structures
- Solve problems using numerical integration, roots of equations, simultaneous equations, finite difference methods, matrix analysis, linear programming, dynamic programming, and heuristic solutions
- Document engineering design solutions

While these are ambitious goals for an introductory course, the easy-to-use nature of MATLAB has made it possible to accomplish them. Taking advantage of the flexibility of MATLAB, students in the course have generated and measured electrical signals, recorded sounds, calibrated measuring devices, and played games. They have also written solutions to Sudoku puzzles and generated complex fractal trees.

Incorporating Data Acquisition Labs

A common theme in course reviews was that students wanted to see the computer “do something.” As a result, a series of data acquisition (DAQ) labs was added to the course (Figure 1).

Sample Lab Work

One of the first labs that we introduced has students verify the calibration of a pressure sensor and analyze the accuracy and applicability of narrow- and broad-range calibrations. Students are given a 1L graduated “cylinder” (actually, the frustum of a cone, which becomes important in later calculations) with a small hole drilled at the 100 mL mark. A small fill tube connected to a pressure sensor attaches to the hole. The sensor has four lead wires, two to power the sensor and two to take measurements.



Figure 1. Students working in the lab with sensor data.

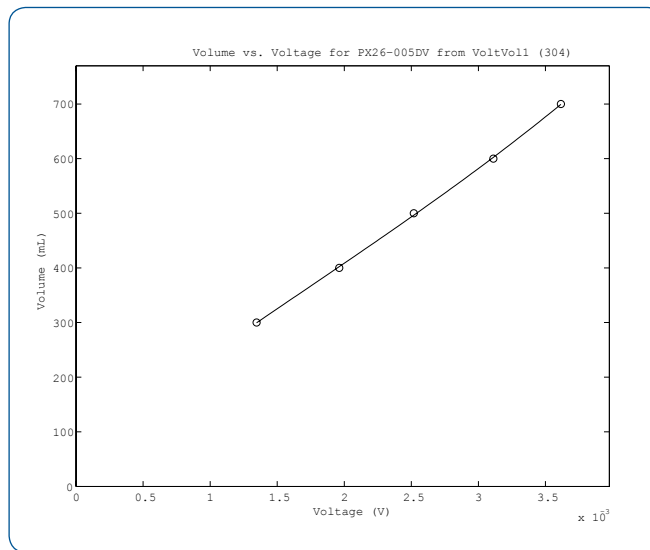


Figure 2. Plot showing volume as a function of voltage.

Students must use knowledge acquired from previous DAQ labs to correctly wire the sensor leads to a National Instruments CB-68LP connector block attached to a National Instrument PCI 6014e Multi-function DAQ card. The card can both power the sensor and read voltages from it.

The students then run four programs using MATLAB and Data Acquisition Toolbox to:

- Acquire data
- Calibrate the sensor
- Read the fluid volume
- Validate the manufacturer's calibration

Acquiring Data

The acquisition program starts with code that initializes the data acquisition process and turns on the sensor. Next, students write a loop that asks the user to enter the current in the cylinder.

Once this value is stored, the program averages and stores 1,000 voltage samples from the sensor. The loop continues until the user specifies that the acquisition phase is over, at which point the acquisition data sets are saved.

This program focuses on early concepts within the class: command window I/O, structured programming, storing data using vectors, and saving data. Students take two sets of data, one over a wide range of volumes and one over a narrow range.

Calibrating the Sensor

The calibration program takes the voltage and volume data saved in the previous step and calculates a fit for volume as a function of voltage (Figure 2). The program then uses the function to calculate the volume of fluid in the cylinder by simply reading a voltage from the sensor.

The pressure sensors exhibit a first-order relationship between pressure and voltage, but because the cylinders aren't truly cylindrical, there is a higher-order relationship between the volume and the height (and thus, the pressure), resulting in a higher-order relationship between the

```

clear

% load data
FileName = input('Enter filename for 3rd order fit coefficients (eg
Vol1Coefs): ', 's');
FileLoad = sprintf('load %s', FileName);
eval(FileLoad)

%% Prepare output
% Create handle to output
AO = analogoutput('nidaq');
% Add channel 0
addchannel(AO, 0);
% Put 10V to channel (powers sensor)
putsample(AO, 10);

%% Prepare input
% Create handle to input
AI = analoginput('nidaq');
% Add channel 1
addchannel(AI, 1);
% Set Input, Sensor, and Units range to [-0.05, 0.05]
set(AI.Channel(1), 'InputRange', [-0.05, 0.05]);
set(AI.Channel(1), 'SensorRange', [-0.05, 0.05]);
set(AI.Channel(1), 'UnitsRange', [-0.05, 0.05]);

%% Plot
figure(1);
hold off;
QPlot = bar(0);
axis([.5 1.5 0 1000]);
ylabel('Volume (mL)');
fprintf('Time (s) Voltage (mV) Volume (mL)\n');
tic
while 1,
    % Get data from transducer
    for k=1:1000
        VolArray(k) = getsample(AI);
    end
    % Find average value of data
    AvgVoltage = mean(VolArray);
    % Use coefficients to determine volume
    Volume = polyval(P, AvgVoltage);
    % Update bar plot
    QPlot = bar(Volume);
    axis([.5 1.5 0 1000]);
    fprintf('%8.2f %12.4f %12.2f\n', toc, AvgVoltage*1000, Volume);
    drawnow
end

```

Figure 3. Excerpt from MATLAB program to read fluid volume based on calibration data.

volume and the voltage. The calibration program takes this into account using a third-order fit, then plots the original data points and the curve of best fit.

This program focuses on later material in the course: loading data, graphically representing data and fit lines, calculating polynomial fits, and creating data sets to represent fit lines. The data for the narrow calibration, which is closely spaced relatively far from the origin, results in an ill-conditioned matrix when the `polyfit` command is used. This opens up discussion of why that happens and how to compensate for it.

Reading the Fluid Volume

The students are given a program to read the fluid volume based on the calibration data (Figure 3).

The program loads calibration information and then continually reads voltages from the sensor. As it does so, a figure window with a graphical representation of a cylinder shows the fluid level, while in the command window, the sensor voltages and equivalent volumes are displayed as a strip chart.

Every semester, a few students fail to realize that the computer is actively monitoring the system and ask, “Why isn’t the level changing?” This leads to the “That’s so cool” moment when I change the water level and the computer representation changes accordingly.

For this program, students manually take volume estimate data using each of the calibrations for several fluid levels between the ranges of the narrow and broad calibrations. Several factors, such as the ill-conditioned matrix and the measurement error, invariably lead to the narrow-range calibration producing major errors in extrapolation. (Students generally report either negative volumes or volumes overflowing the cylinder.)

This program demonstrates the hazards of extrapolation and the importance of creating a model that is not only mathematically correct but also physically realistic.

Validating the Manufacturer's Calibration

In the final program, students use their data to see whether the manufacturer's claim that a sensor with 10V across the power leads will actually show 10 mV of change across the sensing leads for every 1 *psi* change in pressure.

This final program contains code to calculate fluid height as a function of volume, since the conversion from volume to height for the frustum of a cone is both irksome and outside the scope of the course. The program presents a graphical examination of voltage as a function of pressure (Figure 4) and allows the students to either confirm or challenge the manufacturer's specifications.

In four years, every sensor properly attached to the DAQ card has produced calibrations that match the manufacturer's claim exceedingly well.

This feature of the course is especially valuable. Students—especially in introductory courses—enjoy working on an assignment where the results are tangible and verifiable. This approach also builds confidence in the reliability of computational methods, preparing them for those times when the answers are not known a priori.

Advantages of Using MATLAB

This lab demonstrates the importance of MATLAB and Data Acquisition Toolbox to the course. Giving students an easy-to-use but powerful language with which they can obtain real-world data, analyze it, present both the data and the results of the analysis graphically, and verify published specifications of devices starts a process of exploration that we hope will motivate the rest of their undergraduate engineering experience.

Despite their different background experiences and interests, students can easily jump in and start work on practical engineering problems while learning about programming and numerical methods. Using MathWorks tools has significantly improved both the student experience and the concepts and skills covered in the course. ■

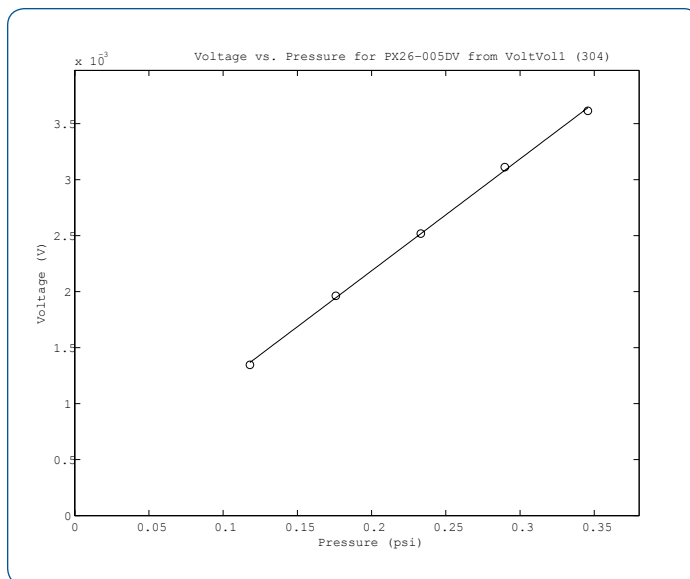


Figure 4. MATLAB plot comparing voltage and pressure.

For More Information

- Pratt School of Engineering, Duke University
www.pratt.duke.edu
- Data Acquisition Toolbox
www.mathworks.com/products/daq

Resources

VISIT

www.mathworks.com/academia

TECHNICAL SUPPORT

www.mathworks.com/support

ONLINE USER COMMUNITY

www.mathworks.com/matlabcentral

DEMOS

www.mathworks.com/demos

TRAINING SERVICES

www.mathworks.com/training

THIRD-PARTY PRODUCTS AND SERVICES

www.mathworks.com/connections

WORLDWIDE CONTACTS

www.mathworks.com/contact

E-MAIL

info@mathworks.com

© 2007 MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks and SimBiology, SimEvents, SimHydraulics, the T-shaped membrane, Embedded MATLAB, and PolySpace are trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.