ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

New Orleans Jazzed about Engineering Education

Paper ID #17130

Freshman Engineering Problem Solving with MATLAB for All Disciplines

Prof. Roche de Guzman, Hofstra University

Dr. Roche de Guzman obtained his Ph.D. in Biomedical Engineering at Wayne State University in Detroit, MI in 2008. He had postdoctoral trainings at the Wake Forest Institute for Regenerative Medicine (Winston-Salem, NC) and at Virginia Tech (Blacksburg, MI) prior to becoming an Assistant Professor at Hofstra University (Hempstead, NY) in 2014. He is currently teaching and has taught ENGG 010 (Computer Programming for Engineers), ENGG 081 (Bioengineering), ENGG 118 (Biomaterials), ENGG 108 (Biomaterials Lab), ENGG 199 (Research), and ENGG 143G (Senior Design). His research interests are: Biomaterials and Mathematical Modeling. He is an active member of the Biomedical Engineering Society (BMES) and Society for Biomaterials (SFB).

Dr. John Carmine Vaccaro, Hofstra University

John Vaccaro grew up on Long Island in Levittown, New York. After graduating with a B.S. in mechanical engineering from Hofstra University ('06), Dr. Vaccaro went on to earn his Ph.D. in aeronautical engineering in 2011 from Rensselaer Polytechnic Institute. His area of research is in the field of experimental fluid mechanics and aerodynamics with a focus on wind tunnel testing. Specifically, he has collaborated with the Northrop Grumman Corporation researching the use of flow control in aggressive engine inlet ducts. After graduation, Dr. Vaccaro held a lead engineering position with General Electric Aviation in Lynn, Massachusetts. There, he designed the fan and compressor sections of aircraft engines. He frequently returns to General Electric Aviation as a consultant. Currently, he is an Assistant Professor of Mechanical Engineering at Hofstra University in Hempstead, New York where he teaches Fluid Mechanics, Compressible Fluid Mechanics, Heat Transfer, Heat Transfer Laboratory, Aerodynamics, Measurements and Instrumentation Laboratory, and Senior Design in addition to conducting experimental aerodynamics undergraduate research projects.

Dr. Alexander Hans Pesch, Hofstra University

Alexander H. Pesch was born and raised in northeastern Ohio. After graduating from Ohio University, he spent time in the jet engine overhaul industry before pursuing graduate studies at Cleveland State University. During his time studying at Cleveland State, he also taught undergraduate classes and participated in research at the Center for Rotating Machinery Dynamics and Control. Currently, Dr. Pesch is an assistant professor of engineering at Hofstra University. His duties include teaching undergraduate classes, engaging in scholarly research, and participation in the Hofstra University Robotics and Advanced Manufacturing Laboratory and Hofstra University Center for Innovation which grow the knowledge base of New York in the area of mechatronics in modern manufacturing and bridge the gap between university and industry development.

Dr. Kevin C. Craig, Hofstra University

Kevin Craig graduated from the United States Military Academy, West Point, NY, with a B.S. degree and a commission as an officer in the U.S. Army. He received the M.S., M.Phil., and Ph.D. degrees from Columbia University, NY. He worked in the mechanical-nuclear design department of a major engineering firm in NYC and taught and received tenure at both the U.S. Merchant Marine Academy and Hofstra University. While at Hofstra, he received the 1987 ASEE New Engineering Educator Excellence Award, a national honor. From 1989-2008, as a tenured full professor of mechanical engineering at Rensselaer Polytechnic Institute, he developed the mechatronics teaching and research program focusing on human-centered, model-based design with a balance between theory and industry best practices. He collaborated extensively with the Xerox Mechanical Engineering Sciences Laboratory (MESL), an offshoot of Xerox PARC, during this time. At Rensselaer, he graduated 37 M.S. students and 20 Ph.D. students, and authored over 30 refereed journal articles and over 50 refereed conference papers. In 2006 at RPI, he received the two highest awards conferred for teaching: the RPI School of Engineering Education Excellence Award and the RPI Trustees' Outstanding Teacher Award. Over the past 20 years, he has conducted hands-on,

integrated, customized, mechatronics workshops for practicing engineers nationally and internationally, e.g., at Xerox, Procter & Gamble, Rockwell Automation, Siemens Healthcare Diagnostics, Fiat, Tetra Pak, Johnson Controls, and others. He is a Fellow of the ASME and a member of the IEEE and ASEE. In January 2008, he joined the faculty of the Marquette University College of Engineering as Professor of Mechanical Engineering and the Robert C. Greenheck Chair in Engineering Design, a $5M endowed chair. He was given the 2013 ASEE North-Midwest Best Teacher Award and the 2014 ASME Outstanding Design Educator Award, a society award. In the fall of 2014, he returned to the Hofstra University School of Engineering and Applied Science as a tenured full professor of mechanical engineering. He is the Director of the $1M Robotics and Advanced Manufacturing Laboratory, and also the Director of the Center for Innovation, a new center created to collaborate with business and industry to foster innovation where all intellectual property (IP) belongs to the sponsor.

Freshman Engineering Problem Solving with MATLAB & Simulink

Abstract

The current freshman engineering computer programming course, which utilizes MATLAB programming language, is being experimentally redesigned to incorporate and highlight activities focused on engineering problem solving and system investigation processes. These methods hope to develop the students' critical-thinking and analytical skills that are more suited and applicable in real-world engineering. Course description and sample problems are presented. Results will be shown in a follow-up study comparing the standard computer program syntax-based approach to this pilot course which employs Simulink model-based designs and hardware demonstrations.

1 Introduction

In 1969, one of the authors was a plebe (freshman) at West Point, engineering was the required course of study, and a slide rule (Figure 1) was standard issue. In his first engineering class, there was a 10-foot-long working slide rule hanging from the ceiling to aid in instruction. He never once thought that his slide rule was going to solve an engineering problem he was facing; it would just make his calculations easier. A decade later, he also never thought that his hand-held calculator was going to solve his engineering problems, but now he could solve easily many more types of engineering problems without having to resort to punch cards and mainframe computers. However, his ability to estimate orders-of-magnitude was diminished.

Figure 1. **A slide rule.**

The early 1980s saw the rise of the personal computer, and now every entering engineering student at most universities has a laptop computer fully-loaded with the latest technical software. When confronted with a problem before the desktop/laptop computer era, the engineering student would develop the problem solution by hand, with pencil, paper, and much thought, and only then was the slide rule or calculator taken out of its case, or, if needed, a computer program written and cards punched. Today, entering freshmen have the perception that the solutions to engineering problems are somewhere in the computer and just have to be found, when in fact the solutions are where they have always been – in the minds of the engineers!

Freshman engineering students in all disciplines usually take some computing class (such as C, Java, or MATLAB programming) – hopefully, learn about pseudo codes and flowcharting, and then solve simple problems developed primarily to make use of some features of the programming language just learned. It is certainly extremely valuable for an engineer to be able to develop an algorithm to implement a solution to an engineering problem and then turn that algorithm into computer codes (e.g., MATLAB syntax). But it is the process that is most valuable: the process of thoroughly understanding the problem, making simplifying assumptions to develop a physical model of the problem, applying the laws of nature to the physical model to create a mathematical model, and then solving the mathematical equations, usually by creating an algorithm, and then programming that algorithm on a computer to gain insight and understanding. But, once that is done, in engineering practice today, most complex engineering problems are solved using pre-written programs in MATLAB or LabVIEW, for example, and only in special situations will an engineer write a computer program from scratch specifically tailored to solve a complex problem. Even in real-time computer applications, auto-code generation programs (e.g., Simulink and LabVIEW) are widely available and are quite reliable. It would be most valuable if freshman engineering students were first exposed to the types of engineering problems real engineers in any discipline face 90% of the time and appreciate this process. It certainly would put their laptop computer, computer software, and computer programming in the proper context. All engineers today, and in the future, must be able to model multidisciplinary engineering physical systems, predict how they will behave when built, optimize their design, validate their predictions and designs with engineering measurements, and see a design through to prototyping and manufacturing, with sustainability considerations paramount throughout.

Let's fast forward four years. The engineering student is now graduating and interviewing for a job. How many piano tuners are there in the city of Chicago? Estimation of rough but quantitative answers to unexpected questions about many aspects of the natural world was frequently used by Enrico Fermi, the legendary physicist, to gage one's power over his/her theoretical and experimental studies. These types of questions draw upon a deep understanding of the real world and upon everyday experience. Many companies use problem-solving questions in job interviews to judge the intelligence and flexibility of their applicants. Examples of "Fermi Questions" used are: What does it really cost to drive a car? How many golf balls does it take to fill a 747 airplane? Anyone can make up a Fermi question. These types of questions serve as a test of applicants' abilities to think on their feet and to apply their mathematical skills to real-world problems. There is no single correct way to analyze these types of questions; there are many paths to the answer. All you need to answer these questions is a willingness to think! Here is another example: Your chance of winning the Mega Millions lottery is one in 100 million. If you stacked up all the possible different lottery tickets, the height of the stack would be greater than Mt. Everest. True? This ability is directly related to the modeling and analysis of engineering systems. The very crux of engineering analysis and the hallmark of every successful engineer is the ability to make shrewd and viable approximations which greatly simplify the system and still lead to a rapid, reasonably accurate prediction of its behavior.

Engineers are problem solvers and the only way to learn problem solving is to do it! Only a human can solve problems; the computer is a tool. Design problems are the heart of engineering and to solve them requires creativity, teamwork, and broad knowledge. The approach to solving an engineering problem should proceed in an orderly, stepwise fashion, but often problem solving is an iterative procedure. To become a good problem solver, an engineer must have – knowledge, experience, learning skills, motivation, and communication skills. The ability to logically break a problem into pieces is most important.

It is the mindset these Fermi questions engender and the challenges engineers face in modern practice that motivates the reinventing of the standard Engineering Programming course found in some form at every engineering school in the country. With all this in mind, a pilot course is now being taught that attempts to instill excitement and relevance into a course that, desperately needs revision. This approach to engineering programming is not new and has been proposed previously[1-3].

2 Course Description

ENGG 010: Computer Programming for Engineers is a three-credit (3-hour per week) freshman engineering course offered at 5-6 sections per year (or 2-3 sections per semester). There are approximately 25-30 students in a class taught in a computer lab equipped with various softwares including MATLAB and LabVIEW. The main programming language of choice is MATLAB. A typical syllabus include topics in assigning values to variables, creating scalars, vectors and matrices, writing scripts and functions, utilizing mathematical, relational and logical operators, matrix indexing and manipulation, plotting, solving linear systems, programming constructs: if statement, for loop and while loop, animation, and building GUI programs. A pilot course is now being taught to two ENGG 010 sections at the present time.

This pilot freshman engineering course (Figure 2) emphasizes the engineering problem solving process and the engineering system investigation process, and applies both to the physics of everyday life as experienced by the students. Fundamentals of feedback control are introduced due to its pervasiveness in the human body, nature, and all engineering systems. The differences between the analog and digital world, including sampling, aliasing, and quantization, are fundamental and emphasized. The course applies various computer tools, essential in all subsequent engineering courses and professional practice, in sufficient detail for the students to be able to begin to apply them in real-world problem solving and model-based design. MATLAB, Simulink, and MuPAD are used for – engineering computation, matrix algebra, numerical integration and differentiation, equation solving, plotting, interpolation and curve-fitting, and m-file programming; graphical programming using Simulink to predict dynamic system behavior; symbolic mathematical analysis using MuPAD; and real-time microcontroller programming using auto-code generation. Measurement with LabVIEW is also introduced.

The importance of the process of engineering problem solving (Figure 3) is highlighted to the students at the beginning of the course. Every engineering problem must be properly assembled and analyzed, then solutions are calculated and presented. Specifically, the process involves subdividing into: Given, Find, Diagram, Basic Laws, Assumptions, Analysis, Numbers, Check, and Label. Every assignment given must be completed following this process. This is continued throughout all the courses in the engineering curricula.

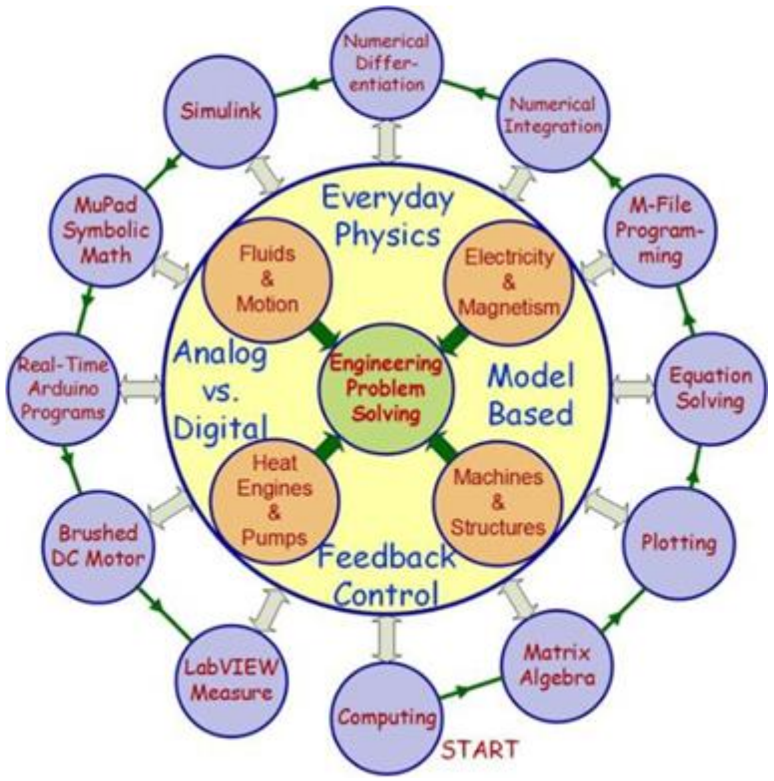
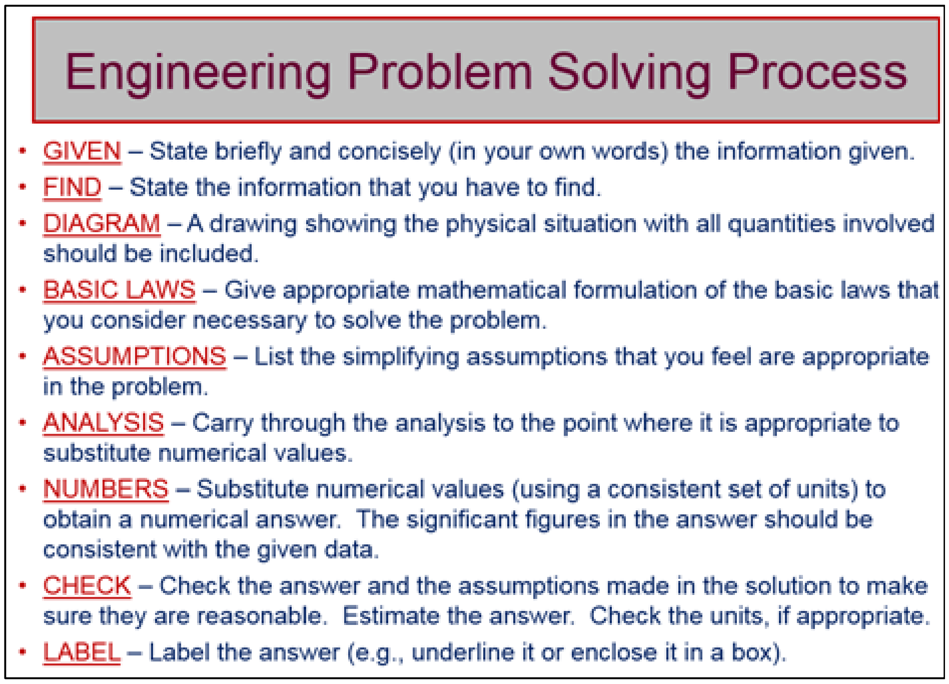


Figure 2. Pilot course topics centered on engineering problem solving.



Engineering Problem Solving Process

- GIVEN – State briefly and concisely (in your own words) the information given.
- FIND – State the information that you have to find.
- DIAGRAM – A drawing showing the physical situation with all quantities involved should be included.
- BASIC LAWS – Give appropriate mathematical formulation of the basic laws that you consider necessary to solve the problem.
- ASSUMPTIONS – List the simplifying assumptions that you feel are appropriate in the problem.
- ANALYSIS – Carry through the analysis to the point where it is appropriate to substitute numerical values.
- NUMBERS – Substitute numerical values (using a consistent set of units) to obtain a numerical answer. The significant figures in the answer should be consistent with the given data.
- CHECK – Check the answer and the assumptions made in the solution to make sure they are reasonable. Estimate the answer. Check the units, if appropriate.
- LABEL – Label the answer (e.g., underline it or enclose it in a box).

Figure 3. Engineering problem solving process.

The engineering system investigation process, shown in Figure 4, is a procedure an engineer follows to thoroughly investigate, i.e., understand, predict, and experimentally verify, how a dynamic engineering system or device performs, no matter how simple or complex the system may be. It is an iterative process, as understanding how the system performs requires simplifying assumptions initially. These initial simplifying assumptions may later be relaxed or changed as understanding develops through comparison of analytical predictions with experimental observations. Comparing the predicted with the actual measured dynamic behavior is the key step in the investigation process. It is important to note that the steps in this process should be applied not only when an actual physical system exists and one desires to understand and predict its behavior, but also when the physical system is a concept in the design process that needs to be analyzed and evaluated. After recognizing a need for a new product or service, one uses past experience (personal and vicarious), awareness of existing hardware, understanding of physical laws, and creativity to generate design concepts. Modeling and analysis in the design process has never been more important. These design concepts can no longer be evaluated by the build-and-test approach because it is too costly and time consuming. Validating the predicted dynamic behavior in this case, when no actual physical system exists, then becomes even more dependent on one's past hardware and experimental experience.

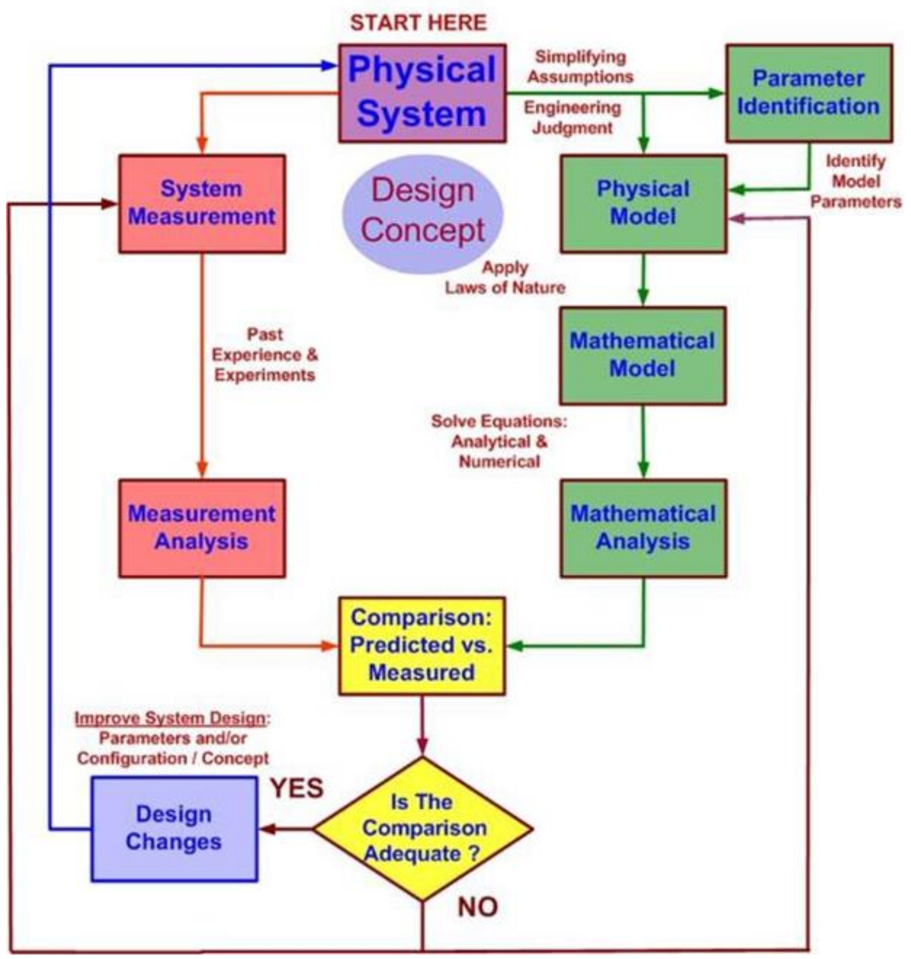


Figure 4. Engineering system investigation process.

3 Sample Course Exercises

Several exercises are taught in the class to demonstrate and stress the importance of engineering problem solving and system investigation processes integrated with contemporary computer programming tools. Some of them are:

a) Projectile motion problem solved by MATLAB m-file script using for/while loops then modeled using interactive GUI.
b) Projectile motion problem solved using Simulink. Nonlinear equations can be solved with Simulink or by assuming constant acceleration over small increments of motion and writing a complex m-file. Both approaches are presented.
c) A feedback control system (the plant is a basic spring-mass-damper rotational system) where the control is proportional, proportional + derivative, and proportional + derivative + integral. Students see how the control modes effect rise time, overshoot, and steady-state error.
d) A studio exercise using the Arduino microcontroller with Simulink auto-code generation to demonstrate aliasing and pulse width modulation.
e) A studio exercise using the Arduino microcontroller with Simulink auto-code generation to control in real time the speed of a brushed dc motor.
f) A writing exercise to explain to a lay person the laws of nature involved in a physical observation.

3.1 a & b - Projectile motion and effect of air resistance

This problem requires the student to apply basic high school physics principles (uniform motion, uniformly accelerated motion, free-body diagrams, Newton's 2nd Law) to a projectile (baseball in this case) and solve the problem first without air resistance, by writing a MATLAB script with loops (for/while), and then with air resistance, by solving the nonlinear equations numerically using Simulink.

Background – We have discussed the motion of projectiles (e.g., baseballs, golf balls, tennis balls, etc.) in air, first neglecting air resistance, and then including air resistance. We know from practical experience that air resistance has a great effect on the trajectory, range, and speed of a projectile in air. In analyzing the motion of a projectile in air, we assume that the acceleration due to gravity, g, is constant and equal to 9.81 m/s^2. We also assume that the mass of the projectile is constant and that the motion is planar (x-y plane). Then we either neglect air resistance or include air resistance in our analysis. In the absence of air resistance, the only force acting on a projectile is the gravitational force, its weight, $m \cdot g$. The projectile motion equations neglecting air resistance for the horizontal (x) and vertical (y) motion of the projectile are:

$$\begin{cases} a_x = 0 \\ v_x = (v_x)_0 \\ x = x_0 + (v_x)_0 t \end{cases} \quad \begin{cases} a_y = -g \\ v_y = (v_y)_0 - gt \\ y = y_0 + (v_y)_0 t - \tfrac{1}{2} g t^2 \end{cases} \tag{1}$$

Here, a is acceleration and v is velocity with the subscripts denoting directions as illustrated in Figure 5a. The variable t is, as usual, time in s.

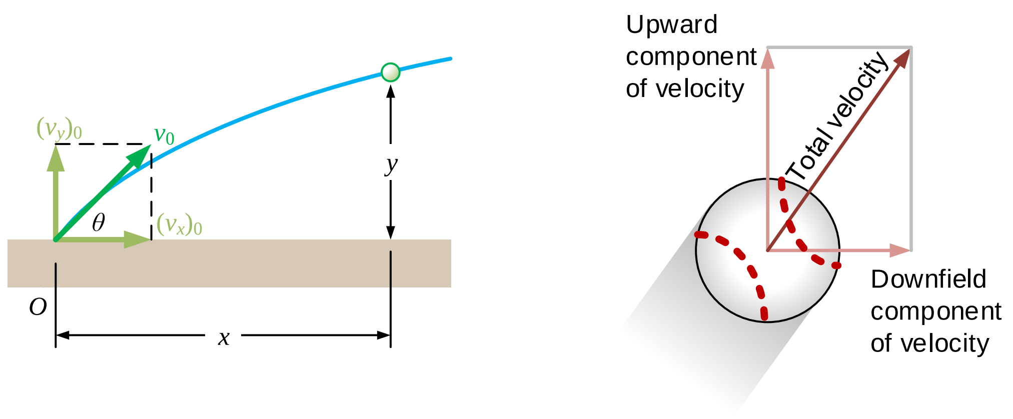


Figure 5a. Projectile problem components, global (left) and local (right).

The initial position of the projectile is x_0 and y_0. The launch angle of the projectile with the horizontal is given by the angle, θ, and the initial projectile velocity is v_0 (with $(v_x)_0$ and $(v_y)_0$ components). The air drag vector force acts opposite to the projectile velocity vector and is proportional to the square of the projectile speed.

$$\begin{cases} F_{\text{drag},x} = -D(v_x)^2 \\ F_{\text{drag},y} = -D(v_y)^2 \end{cases} \tag{2}$$

Therefore, the projectile acceleration vector has the following components:

$$\begin{cases} a_x = -\tfrac{D}{m}(v_x)^2 \\ a_y = -g - \tfrac{D}{m}(v_y)^2 \end{cases} \tag{3}$$

The constant, D, depends on the density of air, ρ, the silhouette area of the body, A, and the drag coefficient constant, C, that depends on the shape of the body. Typical values of C for baseballs and tennis balls are in the range 0.2 to 1.0.

$$D = \tfrac{\rho C A}{2} \tag{4}$$

Requirements – The radius of a baseball is $r = 0.0366$ m and its mass is $m = 0.145$ kg. The drag coefficient $C = 0.5$, appropriate for a batted ball or a pitched fastball. The density of air is $\rho = 1.2\ \text{kg/m}^3$, appropriate for a ballpark at sea level. The initial velocity of the baseball $v_0 = 50$ m/s at an angle of $\theta = 35°$ above the horizontal. Answer the following questions by following the engineering problem solving process (Figure 3), documenting all steps.

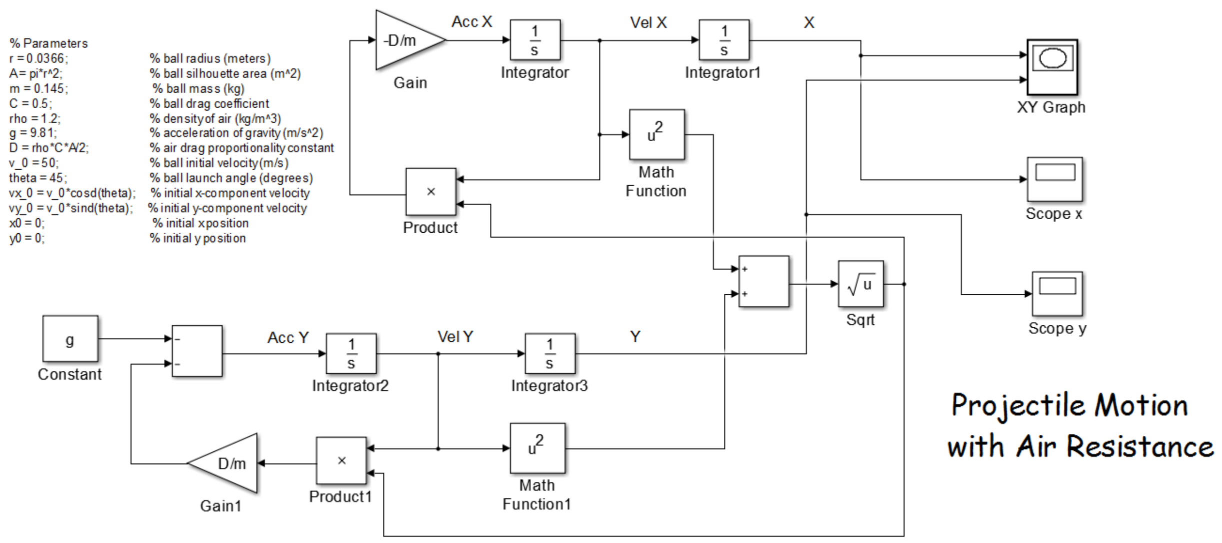
Tasks –

1) Analyze the motion of the baseball without air resistance by writing an m-file program, using an explicit for loop, to plot the range (x) vs. time (t), the height (y) vs.

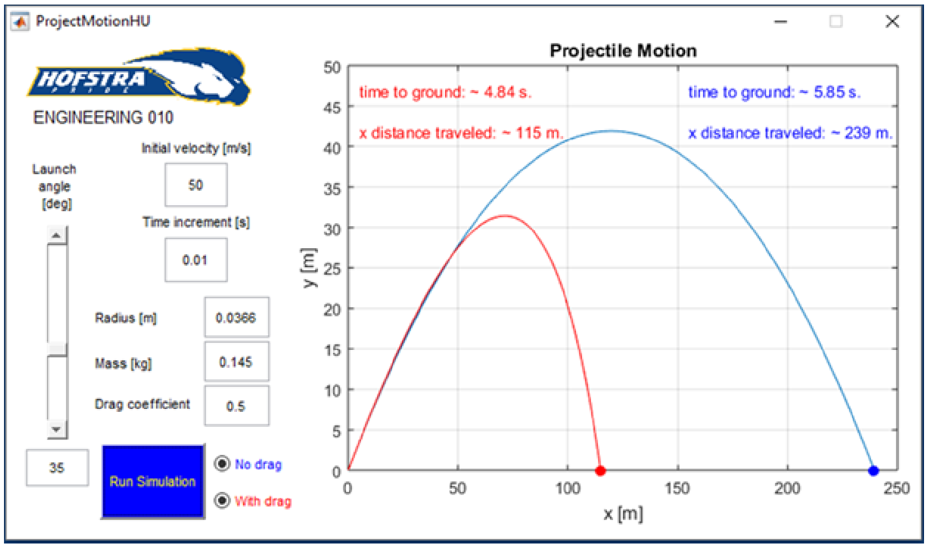
t, *x* vs. *y*, and velocity (*v*) vs. *t* from time *t* = 0 until the ball hits the ground. Include the MATLAB script with the generated figure plots well-labeled.

2) Analyze the motion of the baseball with air resistance using Simulink to solve the equations of motion. Show plots of *x* vs. *t*, *y* vs. *t*, *x* vs. *y*, and *v* vs. *t* from *t* = 0 until the ball hits the ground. Include with the plots, well-labeled, a hand drawing of the Simulink block diagram, along with the actual Simulink block diagram file well-annotated.
3) Compare the flight of the baseball, *x* vs. *y* and *v* vs. *t*, across level ground, both without air resistance and with air resistance. What are your observations?

Example deliverables are shown in Figure 5b which includes a Simulink block diagram for performing the analysis and a GUI for multiple user inputs and plotting the resulting animation.



```
% Parameters
r = 0.0366;                     % ball radius (meters)
A= pi*r^2;                      % ball silhouette area (m^2)
m = 0.145;                       % ball mass (kg)
C = 0.5;                        % ball drag coefficient
rho = 1.2;                      % density of air (kg/m^3)
g = 9.81;                       % acceleration of gravity (m/s^2)
D = rho*C*A/2;                  % air drag proportionality constant
v_0 = 50;                       % ball initial velocity (m/s)
theta = 45;                     % ball launch angle (degrees)
vx_0 = v_0*cosd(theta);         % initial x-component velocity
vy_0 = v_0*sind(theta);         % initial y-component velocity
x0 = 0;                          % initial xposition
y0 = 0;                         % initial y position
```



ProjectMotionHU
HOFSTRA
ENGINEERING 010
Run Simulation
Projectile Motion
time to ground: ~ 4.84 s.
x distance traveled: ~ 115 m.
time to ground: ~ 5.85 s.
x distance traveled: ~ 239 m.
y [m]
x [m]

Initial velocity [m/s]: 50
Launch angle [deg]: 35
Time increment [s]: 0.01
Radius [m]: 0.0366
Mass [kg]: 0.145
Drag coefficient: 0.5
No drag
With drag

Figure 5b. Projectile motion problem and sample solutions. Simulink block diagram (top), simulation animation with axes to scale (middle), and interactive GUI with animation and user inputs (bottom).

3.2 c - Satellite antenna tracking problem

You wish to control the elevation of the satellite-tracking antenna such as the kind shown in Figure 6a.

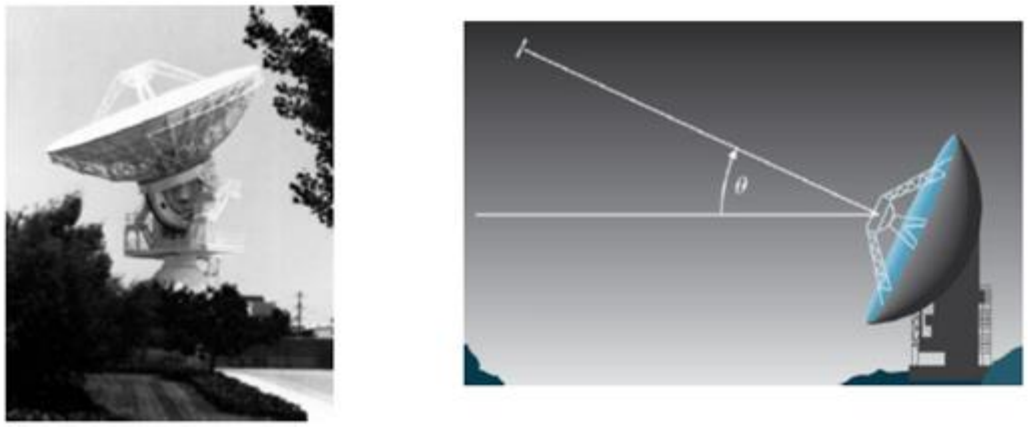


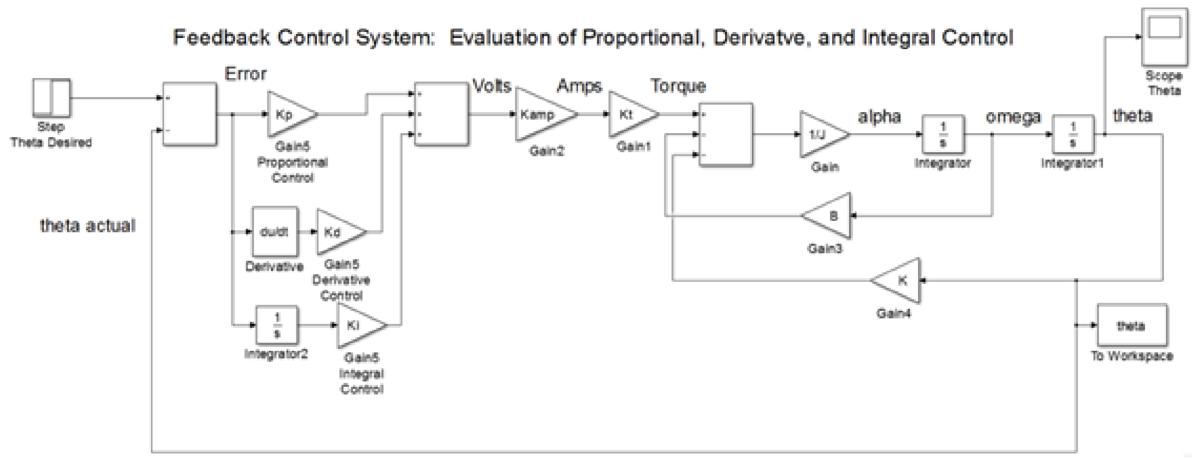
Figure 6a. Example of a satellite tracking antenna.

The antenna and moving parts have a combined moment of inertia J (60,000 kg-m^2) and an angular viscous damping coefficient B (20,000 N-m-s). The viscous damping term captures the energy-dissipation contributions from bearing and aerodynamic friction. For this exercise, we will neglect actuator and sensor dynamics, and any parasitic effects, e.g., time delay, backlash, and saturation. However, there is a compliance, a "springiness", in the mechanism that affects the performance. It can be quantified as a torsional spring with a stiffness K (N-m/rad) equal to 3000.

Tasks –

1) A picture and sketch of the physical system are shown. Draw a picture of the physical model along with a list of all simplifying assumptions.
2) Draw a free-body diagram of the physical model. How many degrees-of-freedom does the model have? Apply Newton's 2nd Law to the free-body diagram to obtain the mathematical model, i.e., the equation of motion.
3) A feedback control system must be designed. Draw the block diagram of the feedback control system showing the plant and controller blocks. Sensor and actuator dynamics are being neglected in this initial investigation.
4) A PID controller is proposed. Draw a Simulink diagram of the feedback control system. Use Simulink simulations to show how each of the PID control gains, K_p, K_d, and K_i (proportional, derivative, and integral, respectively), contribute to: rise time, overshoot, and steady-state error.
5) Pick values for K_p, K_d, and K_i to give a rise time less than 5 s, an overshoot less than 10%, and zero steady-state error.

Example deliverables are shown in Figure 6b which includes a Simulink block diagram for performing the analysis and several time responses for different prospective derivative gains.



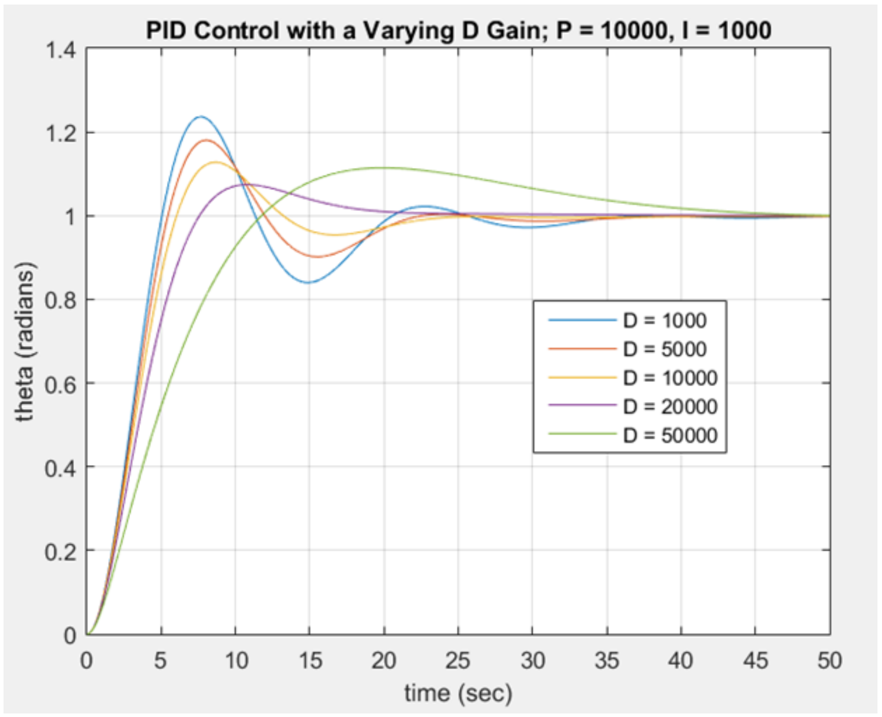
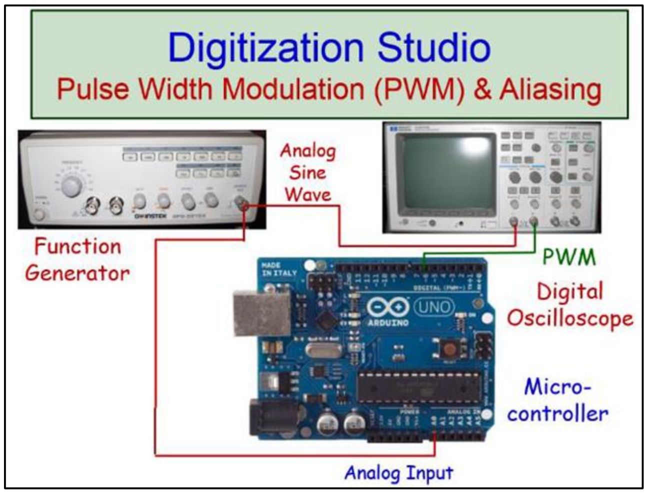


Figure 6b. Satellite antenna tracking problem Simulink block diagram (top) and sample time response solutions (bottom).

3.3 d - Studio exercise to demonstrate aliasing and PWM

A coding exercise which includes a tangible, hands-on, component is to observe the aliasing effect in PWM signals. The student is given a micro-controller and tasked with tracking an analog signal with a PWM equivalent. Through programming of the micro-controller and observation of the resulting behavior on an oscilloscope, the student can observe the aliasing effect when the sampling rate is too low and experience the consequences of proper or improper coding on the physical world. Figure 7 shows example hardware and the graphical coding scheme which has been used in this studio exercise. The following is a discussion for the students of the significance of the theoretical significance of the sampling rate and an introduction to the Nyquist Sampling Theorem.



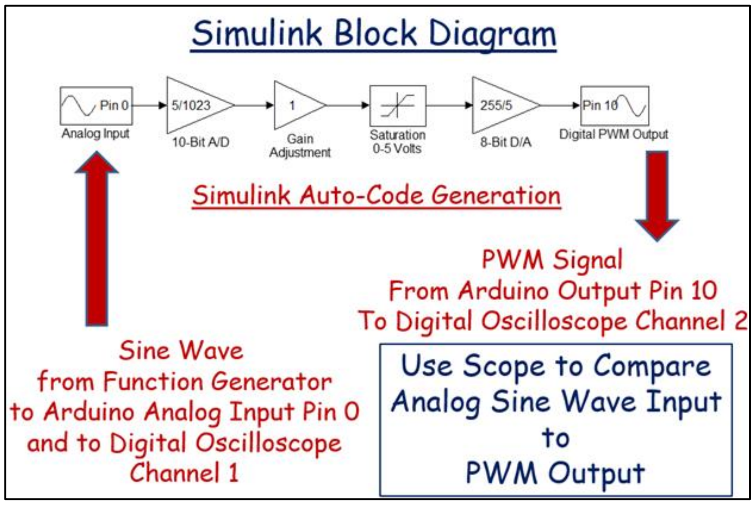

Simulink Block Diagram
Pin 0
Analog Input
5/1023
10-Bit A/D
1
Gain Adjustment
Saturation 0-5 Volts
255/5
8-Bit D/A
Pin 10
Digital PWM Output
Simulink Auto-Code Generation
PWM Signal From Arduino Output Pin 10 To Digital Oscilloscope Channel 2
Sine Wave from Function Generator to Arduino Analog Input Pin 0 and to Digital Oscilloscope Channel 1
Use Scope to Compare Analog Sine Wave Input to PWM Output

Figure 7. Simulink-Arduino application 1, hardware (top) and programming scheme (bottom).

One of the most powerful mathematical results of the digital era is the Nyquist Sampling Theorem. A sampled signal can be converted back to its original analog signal without any error if the sampling rate is more than twice as large as the highest frequency of the signal. The Nyquist Frequency = ½ f_s and it is a discrete-time system property. Restated in mathematical form, if T_s is the spacing between samples and $f_s = 1/T_s$ is the sampling frequency, then theoretically we can convert the samples of an analog signal back into the original signal if $f_s > 2f_{highest}$, where $f_{highest}$ is the highest frequency contained in the analog signal. The value $2f_{highest}$ is called the Nyquist rate. $f_{highest}$ is the highest frequency contained in a signal. The Nyquist rate is the lower bound of the sampling frequency that satisfies the Nyquist sampling criterion. It is a continuous-time signal property.

Mathematicians and engineers added to Nyquist's original sampling result by discovering precisely how to recreate the original signal from only its samples. They showed that if an analog signal is sampled at a rate greater than two times the bandwidth, then it can be exactly reconstructed from its samples. In fact, there is a mathematical formula for reconstructing the signal and it can be implemented in a very practical way. The reconstruction of the original signal from samples of the signal is done with a digital-to-analog (D/A) converter.

The implications of the Nyquist sampling theorem are nothing short of remarkable! The process of sampling a signal, manipulating a sequence of numbers that results from sampling to remove noise or emphasize certain features, for example, and producing an analog output signal after the manipulations is called Digital Signal Processing (DSP). Band-limited signals (signals whose highest frequency falls below some finite value) can be reconstructed perfectly from their samples, as long as the sampling rate is greater than twice the bandwidth of the signal that was sampled.

What happens if we unfortunately sample a signal too slowly and fail to meet the requirement of the Nyquist sampling theorem? Aliasing refers to one signal "pretending to be" another signal when their samples are the same. It is an undesired effect due to undersampling whereby one signal can masquerade as another. Aliasing is an inevitable, irreversible process which shifts frequencies. It cannot be completely eliminated, only reduced.

3.4 e - Brushed DC motor / Arduino closed-loop control exercise

Another exercise in which the student can experience hands-on application of the programing he or she is learning is coding the speed control of a small DC motor. The student uses proper coding of a micro-controller, already learned, to achieve desired performance of the electro-mechanical system. The mechanical response to changes in the coding world keeps the student engaged in mundane task of learning rules of syntax. Figure 8 illustrates a system level summary of the components utilized in the exercise.

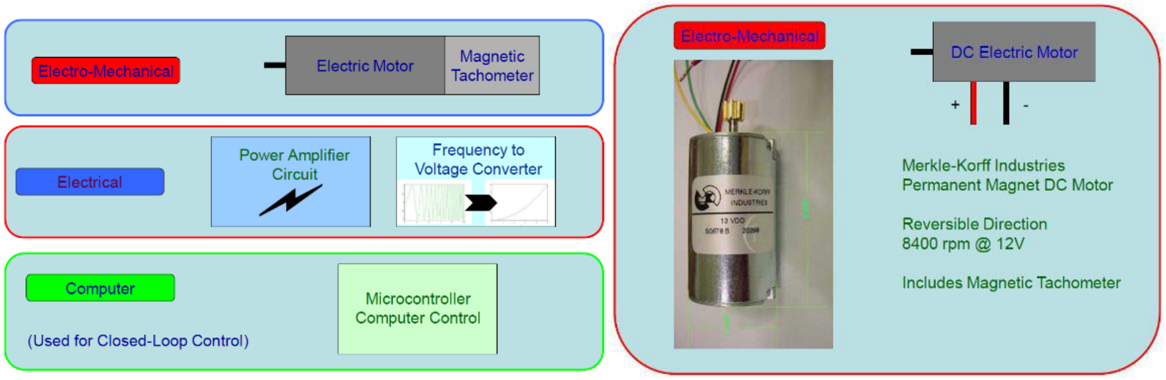


Figure 8. Simulink-Arduino application 2, DC motor system hardware overview.

An *electric machine* is a device that can convert either mechanical energy to electrical energy or electrical energy to mechanical energy. An example of mechanical to electrical is a generator. Conversely, an example of electrical to mechanical is a motor. All practical motors and generators convert energy from one form to another through the action of a magnetic field. Magnetic field acts as the medium for transferring and converting energy. Motors, Generators, and Transformers are ubiquitous in modern daily life. Why?

Electric power is clean, efficient, and easy to control and transmit over long distances. Four basic principles describe how magnetic fields are used in these devices. A current-carrying wire produces a magnetic field in the area around it. A time-changing magnetic field induces a voltage in a coil of wire if it passes through that coil (basis of transformer action). A current-carrying wire in the presence of a magnetic field has a force induced on it (basis of motor action). A moving wire in the presence of a magnetic field has a voltage induced in it (basis of generator action).

3.5 f - Writing assignment example

Drop a magnet down a copper pipe such as is shown in Figure 9. The magnet "floats" down the copper pipe defying gravity. There are five fundamental laws of nature demonstrated here. What are they? See Maxwell's Equations and Newton's Laws of Motion and Gravity. Communication of complex technical concepts in simple terms is the hallmark of an accomplished scientist and engineer. Explain what you observe to a lay person in a 250-word document. Organization, grammar, and style are just as important as accurate content.

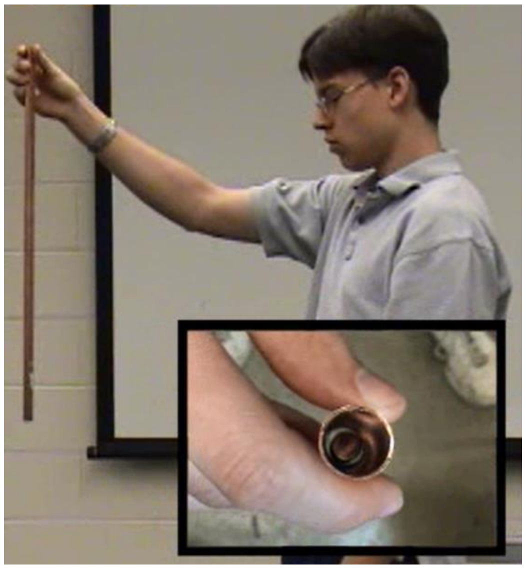


Figure 9. Magnet dropping through a copper tube, an example of a motivating experiment for a writing assignment.

4 Rubric and Assessment

Students evaluations for the course, as well as evaluations of how the course impacts courses in the sophomore year (e.g., Dynamics) and in the junior year (e.g., Modeling, Analysis, and Control of Dynamic Systems) where problem-solving skills and engineering tools (MATLAB, Simulink, SimMechanics, LabVIEW) are widely applied, are being conducted and will be reported in future work.

5 Conclusion and Future Work

A pilot, 3-credit, freshman course, called Engineering Problem Solving, is currently being evaluated to potentially replace the traditional Computer Programming course. The course focuses on real-world problems and emphasizes two processes: the Engineering Problem Solving Process and the Engineering System Investigation Process. MATLAB scripts utilizing iterations and conditional statements are learned, as well as Simulink graphical programming. The Arduino microcontroller is used with the MATLAB Simulink real-time code generation to understand aliasing and pulse-width modulation and perform real-time speed control of a brushed DC motor. Symbolic mathematics using MATLAB MuPAD is introduced, as is measurement using LabVIEW with the myDAQ National Instruments device. The pilot course is now in its second offering to 2 sections and will be expanded to possibly include all freshman engineers next year. Already students who have taken the course have expressed approval as they are using what they have learned in their sophomore Statics, Dynamics, and Strength of Materials courses. Formal evaluations will be conducted as the pilot course becomes a regular course for freshman engineers.

Bibliography

1. *A High-School Level Course in Feedback Control*, Jorge Cortes and William Dunbar, IEEE Control Systems Magazine, June 2007, pages 79-89.
2. *Design of an Introductory MATLAB Course for Freshman Engineering Students*, Darryl Morrell, ASEE 2007, pages 12.458.1-12.458.7.
3. *An Introduction to Technical Problem Solving with MATLAB v.7*, Jon Sticklen and M. Taner Eskil, Oxford University Press, April 2006.