Submitted to the IEEE Control Systems Magazine

Revised: January 28, 2006.

A Crash Course in Feedback Control

A MATLAB®-based introduction with one prerequisite: high school algebra

By Jorge Cortés and William B. Dunbar

Introduction

Motivation for outreach has been generated within the control community in the last decade. In particular, there is a need for new and improved course materials for both traditional (engineering undergraduate and graduate students) and non-traditional (high school, or non-engineering students) audiences. In 1998, an article resulting from an NSF/CSS Workshop on New Directions in Control Engineering Education [1] made the following recommendations regarding needed reform in undergraduate control education:

- *"to provide practical experience in control systems engineering to first year college*

students to stimulate future interest and introduce fundamental notions like feedback and the systems approach to engineering," and

- *"to encourage the development of new courses and course materials that would significantly broaden the standard first introductory control systems course at the undergraduate level."*

In 2003, a Panel on Future Directions in Control, Dynamics, and Systems provided a renewed vision of future challenges and opportunities, along with recommendations to agencies and universities to ensure continued progress in areas of importance to the industrial and defense base [2]. One of the five primary recommendations is that the community and funding agencies invest in *"new approaches to education and outreach for the dissemination of control concepts and tools to nontraditional audiences. As a first step toward implementing this recommendation, new courses and textbooks should be developed for both experts and nonexperts."* The Panel also recommended the integration of software tools such as MATLAB® into these new courses. In 2002, the undergraduate introduction to control course at Caltech (CDS 110) was revamped to incorporate high-level presentation lectures for both non-engineering and engineering students in addition to the more detailed lectures for engineering students [3]. As part of the effort, a new book is also under development [4].

Motivation for outreach has also been provided from outside our field. Ten years ago, feedback control was identified by the National Science Education Standards as being fundamental to understanding systems, and systems in turn was identified as a unifying concept for K-12 science education [5]. In the bigger picture, in 2005, the need for curricular material that motivates middle-school and high school students to pursue advanced work in science and mathematics in the United States was underscored by the Committee on Prospering in the Global Economy of the 21st Century, a subcommittee of the National Academy of

Sciences, National Academy of Engineering, and the Institute of Medicine [6].

This paper presents an overview of a recently designed and implemented introductory course on feedback control amenable to traditional and nontraditional audiences. The course material is based on fundamental concepts in dynamical systems, modeling, stability analysis, robustness to uncertainty, feedback as it occurs naturally, and the design of feedback control laws to engineer desirable static and dynamic response. The material also includes an introduction to MATLAB®, provides many MATLAB® exercises to reinforce concepts, and concludes with a control design and simulation-based analysis to achieve wall tracking with a kinematic robot. The only prerequisite for the course material is high school algebra. In July of 2005, a four-week course based on the material was taken by a group of 17 talented high school students, with a range of 9 to 11 in grade level. By the end of the course ($\sim$30 hours of lecture time), each student had successfully implemented a wall tracking controller on a robot (called `Robobrain`) designed specifically for the course. All course material is freely available online, see [7].

The target audience for our course material includes not only high school students, but undergraduates, graduate and postdoctoral researchers, who may have taken few math courses beyond what is required in high schools in the United States, and who want a primer on dynamics and control. The material could be used to initiate an interdisciplinary collaboration between control theoreticians and biologists, for example, by helping the biologists begin to learn the language and principles behind feedback control. The course might also be integrated as part of a first year general engineering course. By design, the material provides explicit motivation for learning more advanced mathematics, a component often missing in introductory engineering curriculums. In the fall of 2006, a first year one-quarter undergraduate course will be developed in the Baskin School of Engineering at UC Santa Cruz

based on this material. In the remainder of the paper, an overview of the course material is provided, followed by a description of the venue for the course given in the summer of 2005, and feedback from high school students that took the course.

Overview of Lecture Material

The material is broken down into **nine lectures**, each taking 1-3 hours to cover, depending on the complexity of the specific lecture topics and the ability of the audience. The first lecture is a set of presentation slides. All other lectures include two parts: first, a 15-20 minute slide-based introduction, giving an overview of the topics covered in that lecture and reiterating the big picture; second, 45-160 minutes of white board lecture and MATLAB® exercises, guided by our lecture notes. Lectures are structured in an interactive way, alternating between short explanations of important concepts given to the whole class, and time allocated for the students to solve simple exercises individually. As stated, all slides, lecture notes and relevant MATLAB® files are freely available online at [7]. The material assumes each participant has access to a computer with MATLAB® installed.

Lectures 1 and 2: Introduction to the Course and MATLAB®

Lecture 1 is a presentation providing a high-level introduction to the concept of feedback control, as it occurs naturally and as it is designed for use in engineering (Figure 1 shows one of the slides from this lecture). The introduction also gives an overview of the mathematics and analysis tools utilized in the course, and a preview of the robotic platform `Robobrain` encountered at the end of the course. Lecture 2 provides an introduction to MATLAB®, including variable assignment, vectors, plotting, making m-file functions, and loop-functions for plotting data. There are many MATLAB® primers available, such as in [11], that could be used to supplement the minimal number of topics that we cover. Still, for the purpose

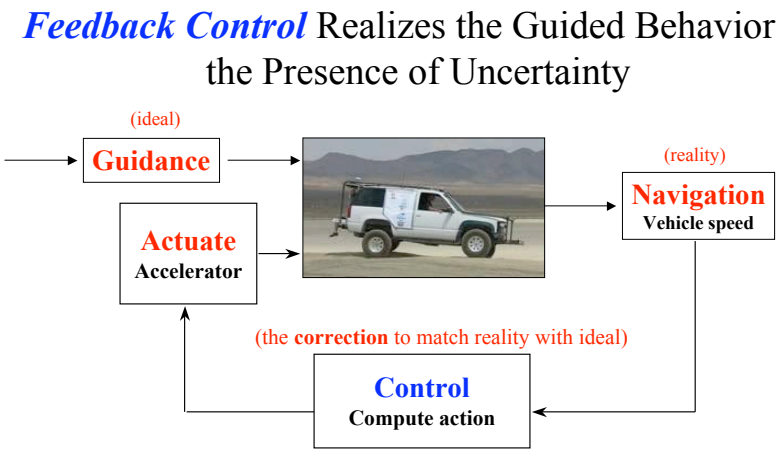

Figure 1. Introductory slide presenting feedback control in the context of guidance, navigation and control of an autonomous off-road vehicle [8]. The slide presents control as a means of correcting the actual system response to match a desired (guidance) response. The example also helps introduce the concepts of stability, performance and robustness using the familiar objective of cruise control of a vehicle.

of moving to the next topics in our course, we found that our coverage was sufficient in practice (see the section later entitled "Implementation of the Course Material" below for a discussion on our experience with teaching the course).

Lectures 3 and 4: Introduction to Discrete-Time Dynamics

Without a background in differential equations, an introduction to dynamics and modeling is facilitated by considering systems in discrete time. In order not to rely on a background in linear algebra, the models considered must also be low dimensional. Therefore, one and two-dimensional linear and nonlinear discrete-time models are considered in Lectures 3 and 4. In the context of these models, basic high school algebra is sufficient to make a beginning.

Dynamics are introduced with the following one-dimensional, discrete-time and time-invariant real-valued map:

$$x_{k+1} = f(x_k). \tag{1}$$

To understand causality, the *orbit* $\{x_0, x_1, x_2, ..., x_N\}$ is defined as the unique sequence of numbers, or trajectory, that arises from a given initial point x_0 and by evaluating the function (1) repeatedly, up to some desired number of times N. We then define what it means for f to be linear ($f(x) = ax$, given $a \in \mathbb{R}$), and not linear, or nonlinear ($f(x) = \cos(x)$, for example). The concept of *fixed point* (equilibrium) is then defined for these maps, and the quantitative (in the case of linear f) and qualitative behavior of (1) near such points is explored. In turn, we are able to qualitatively define *stability* and *attractivity* of a fixed point of (1), by examining orbits for a set of initial conditions near the fixed point.

As a preview into the complexity that is possible with such a simple equation (1), the one-dimensional logistic map [9] is examined, where $f(x) = rx(1-x)$, given $r \in [0, 4]$. As a

first real test of their MATLAB® skills, students are asked to generate a plot that shows the limiting behavior of orbits as the parameter r is varied. Here are the instructions given in the Lecture 4 notes.

Name: Orbit diagram algorithm

Goal: Plot orbit diagram of logistic equation

The idea is to have a figure where many orbits of the logistic map are plotted (one for each value of r between 3.4 and 4). So the x-axis of the figure will correspond to values of r, and the y-axis will correspond to values of the orbits.

1: Set $i = 0$.

2: Set $r = 3.4 + i$.

3: Iterate the logistic map for 200 cycles (after 200 cycles, the system should settle down to its eventual behavior — the settling portion of the response is called the *transient*) starting from $x_0 = .6$.

4: Once the transients have decayed, plot many points, say $x_{201}, \ldots, x_{400}$, versus the current value of r in the figure.

5: Set $i = i + 0.005$. If $i = 0.6$, exit the algorithm. Otherwise, return to step 2.

For those unfamiliar with the logistic map, the resulting figure is an illustration of chaos. Not only does the exercise promote applied learning of writing for-loops and functions in MATLAB®, but the images they generate promote enthusiasm for more programming opportunities and more advanced dynamics. Most student like the idea that they are capable of creating chaos*!*

Lecture 5: Introduction to Modeling

Mathematical models of systems are presented as a tool for *prediction*, so one can characterize

how a system behaves under a variety of conditions. Based on the previous material, students can think of this as identifying a function f that generates an orbit closely matching the actual evolving behavior of a given system. At the outset, the concepts of *uncertainty* and *robustness* are impressed upon the students. Models are never perfect; the hope is that they are good enough for a close match, and eventually, for control design and analysis. An excerpt from Lecture 5 given here shows the language with which we attempt to introduce these fundamental concepts.

Excerpt from Lecture 5

Of course, models are not perfect. They try to describe very complex natural and engineered phenomena. Therefore, we should have in mind that a model is always an approximation of the actual behavior of the real system. This ***uncertainty*** comes from various sources: we usually do not know with total exactness, for example, the values of the parameters of the system (the mass, the friction coefficient, etc.). Another reason is that many dynamic processes are just too difficult to model exactly, and we are often forced to make approximate models.

When approximations are required in modeling, and this is always the case, a principle that has the power to save us is called ***robustness***. Robustness is the ability of a system to be relatively *insensitive to measurement, parameter and environmental variations or uncertainties*. Let us consider examples of such variations, in the case of cruise control.

Example 1 (Sources of uncertainty in cruise control)**.** Recall the cruise control model (presented in Lecture 4)

$$v_{k+1} = v_k + \frac{\Delta}{m}[-bv_k + u_{\text{eng},k} + u_{\text{hill},k}]. \tag{2}$$

Here, v_k is car speed, $u_{\text{hill},k}$ is road incline, $u_{\text{eng},k}$ is accelerator control input, all at time step k. The sample period is Δ and the mass is m. An example of measurement uncertainty is if your measurements of v_k are not exact. For example, your speed sensor is actually giving you $v_k + \sigma_k$, where σ_k is a variable that changes randomly, contaminating the speed measurement. An example of parameter uncertainty is if we do not know the mass m exactly. This is the case as fuel is being burned causing a decrease in mass and we are not taking this into account since it is assumed that m is constant for all time. Lastly, an example of environmental uncertainty is $u_{\text{hill},k}$. At best, we might be able to say how large this term gets over a certain time period, but we do not know in general the exact value of the road incline, now or in the future. ■

Robustness is one of the most useful properties of control. Think again about the cruise controller of your parents' car. You want to keep the car going at constant speed, right? The cruise controller automatically adapts the accelerator setting so that the system is insensitive to climbing uphill or going downhill, and it does so *with no exact knowledge of the true incline of the road traveled!* In short, the cruise controller makes the actual behavior of the car robust to changes in the road incline conditions.

The lecture continues with a two-dimensional predator-prey model

$$H_{k+1} = H_k + \frac{b_r}{D}H_k - \frac{a}{D}L_kH_k,$$
$$L_{k+1} = L_k - \frac{d_f}{D}L_k + \frac{a}{D}L_kH_k,$$

used to predict how hare populations H_k and lynx populations L_k influence one another over time, as a function of the model parameters. The example is the first leap into modeling in more than one dimension, extending the students MATLAB® function writing skills to generate orbit plots for the two populations. This example is followed by the three-dimensional kinematic model of the `Robobrain` robot used as the experimental platform at the end of the course. The model corresponds to that of a unicycle, and is given by

$$\begin{aligned} x_{k+1} &= x_k + \Delta u_k \cos(\theta_k), \\ y_{k+1} &= y_k + \Delta u_k \sin(\theta_k), \\ \theta_{k+1} &= \theta_k + \Delta v_k. \end{aligned} \tag{3}$$

Definitions for *state, control inputs, disturbances, parameters* and *outputs* are given, and examples of each are identified for both models. For example, (v_k, u_k) are the rotational and translational velocity of the robot, respectively, and correspond to the two control inputs. Since the robot model has control inputs, the concept of fixed point is restated, now with the equilibrium state dependent upon the choice of controls. The next lecture introduces feedback control abstractly (not just for the robot), and leverages the learned MATLAB® programming skills for model-based prediction, as well as the previous topics of fixed points and stability.

Lecture 6: Introduction to Feedback Control

The utility of models for prediction is emphasized in the previous Lecture 5 material, and Lecture 6 introduces model-based feedback control design and analysis. To engineer autonomy in systems, e.g., in robots or cruise-controlled cars, models are also used to design and analyze feedback control policies. The advantage of doing control design and analysis with a model, as opposed to trying different controllers on the real system from the beginning, is impressed upon the students. Some of the recent advertisements for MATLAB® and

SIMULINK® by The MathWorks Company in the IEEE Spectrum Magazine are useful in making this point. In particular, the recent ad in [10] identifies the savings and success of prediction-based (simulation-based) control design in the case of the Mars rovers, for which zero test flights to Mars are possible.

Based on the mathematics we have presented so far, feedback control is introduced in this lecture as a means of *shaping the dynamics*, so that (i) one can reassign the fixed point(s) of a model as desired, and (ii) these desired fixed points are stable and attractive. Generically, we rewrite the model in (1) to include a control input u_k as

$$x_{k+1} = f(x_k, u_k). \tag{4}$$

Next, by example, the *unforced* or *open-loop* dynamic model of the cruise-controlled car is reexamined, by setting $u_{\text{eng},k} = 0$ for all k in (2). Students are asked to calculate and analyze the fixed point speed v_{eq} in the absence of any road incline, yielding convergence to $v_{\text{eq}} = 0$ (due to the friction term) from any initial speed. For this one-dimensional example, the stability can be analyzed explicitly be seeing that v_k always tends to slow down. In high-dimensional examples later, simulations (orbit calculations) are the sole means of determining stability, since students are not expected to know any matrix algebra. Continuing with the cruise-control example, students are next asked to recalculate the fixed point with the control in (2) defined as $u_{\text{eng},k} = K(v_{\text{des}} - v_k)$, where v_{des} is the desired speed of the car when the cruise-controller is working. The result (assuming a constant, possibly non-zero incline disturbance) is

$$v_{\text{eq}} = \frac{K}{b+K} v_{\text{des}} + \frac{1}{b+K} u_{\text{hill}}.$$

Thus, one can *engineer* a fixed point, by choice of control. In this case, the larger the value

of K, the closer the steady-state speed approaches its desired value, and the less influence the road incline has on the steady-state speed (robustness). By calculating orbits, students also get to examine the stability and attractivity of the new fixed point.

The remaining lectures prior to the experimental implementation are about control design and analysis, first for a two-dimensional model (the inverted pendulum), and next for the three-dimensional robot model in (3).

Lecture 7: Feedback Control of an Inverted Pendulum

This lecture provides a case study in control design and simulation-based analysis. In Lecture 7, a normalized discrete-time model of a pendulum with angle θ_k and torque control u_k is given as

$$\frac{1}{\Delta^2}(\theta_{k+1} - 2\theta_k + \theta_{k-1}) = -\sin\theta_k + u_k,$$

with the objective of designing u_k to make the pendulum upright position $\theta_{\text{eq}} = \pi$ (a fixed point) stable and attractive. If students have been exposed to Newton's second law in a physics course (about half of our COSMOS students had taken high school physics), this model is easy enough to derive, making using of the finite difference approximation of the second time derivative of the variable θ. The first trick is to convert this model into the form of (4), since all analysis to this point has relied on the difference equations being in first-order form. Although students are often mystified when first introduced to a *change of variables*, the technique is mandatory in many control theory and application results. Therefore, this example serves the additional purpose of providing a simple example of applying a change of variables. Specifically, in (4), the state x_k and function f become two-dimensional, defined

as $x_k = (z_k, y_k) = (\theta_k, \theta_{k-1})$ and

$$f(x_k, u_k) = \begin{bmatrix} 2z_k - y_k - \Delta^2 \sin z_k + u_k \\ z_k \end{bmatrix}.$$

Students are then asked to find the constant torque u_{eq} such that the fixed point state is the upright position, i.e., $x_{\text{eq}} = (\pi, \pi)$. Next, defining the overall control as $u_k = u_{\text{eq}} + \tilde{u}_k$, the desired fixed point is to be stabilized by choosing the parameters K_1 and K_2 in the feedback control

$$\tilde{u}_k = K_1(\theta_{\text{eq}} - z_k) + K_2(\theta_{\text{eq}} - y_k).$$

By using the canned MATLAB® function `pendulum.m`, students examine the response of the pendulum to a variety of control parameter choices, and identify those parameters that result in stability and attractivity. In the later portion of Lecture 7, an integrator (summation term for discrete-time control) is also explored in the cruise-control model (2) to remove the steady-state error in the steady-state speed v_{eq} shown above. Although the students are by now capable of making a function like `pendulum.m`, the infrastructure allowed us to maintain the desired pace of the course.

Lecture 8: Open-loop analysis of the `Robobrain` robot

In Lecture 8, the robot model in (3) is tackled. As with the previous models, the open-loop analysis is performed. The first step is to identify the equilibrium states of the dynamics, with the corresponding equilibrium control inputs. Students are then asked to decide if these equilibrium states are stable and attractive. Once they decide they are not, they are then faced with the need to design a feedback control to achieve our ultimate control objective: autonomous wall tracking, so that, starting from an arbitrary initial configuration,

the `Robobrain` robot turns to track any wall at a desired separation distance, and at a constant velocity.

In preparation for this task, students are guided through a series of exercises involving the robot model, targeted at reinforcing the concepts of open-loop versus closed-loop control, uncertainty and robustness to disturbances. Students are asked to create two MATLAB® programs, `robobrain.m` and `robobrainDist.m`. The first one plots the orbit of the robot given some specific control signals from a generic initial configuration.

Students realize that the controls are specified ahead of time, without ever looking at the actual evolution of the `Robobrain` robot during its motion. The other program, `robobrainDist.m`, is meant to show the risks associated with this open-loop strategy. Students are asked to simulate the same model, with the exact same values for the parameters and the initial conditions as before, but with a slight disturbance in the equations. In our experience, this exercise convinces students of the need for feedback control when solving the wall tracking problem, which is the subject of the next lecture.

Lectures 9: Feedback Control of the `Robobrain` robot

This lecture is structured into three parts. First, students are reintroduced to the discrete-time model (3) of the unicycle, and the correspondence between the control inputs u_k, v_k in the equations and the angular velocities of the physical robot (which are the actual quantities directly controlled) is reviewed.

In the second part, students are guided through the task of defining a strategy for the robot to find the wall and then make it turn to track the wall at a specified separation distance d_{sep}. The measured distances to the wall from each infrared (IR) sensor (see Figure 2) are defined by $d_{l,f}$ (distance from the left front sensor), $d_{l,m}$ (distance from the left middle sensor), and

$d_{l,b}$ (distance from the left back sensor). Students are asked to figure out how to convert the three left scalar distance measurements into the state values x and θ of the robot, and write a MATLAB® program to compute them, assuming a straight long hallway.

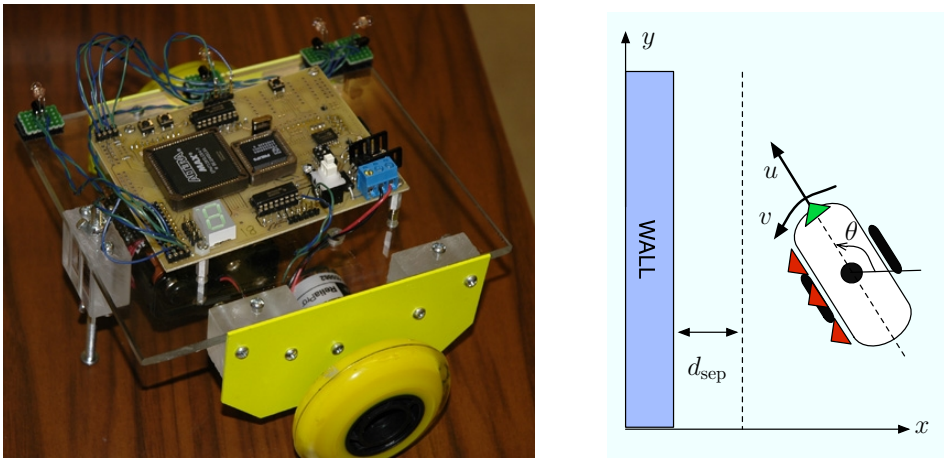

Figure 2. Left, `Robobrain` robot developed at UCSC; right, schematic drawing of robot. The three (red) triangle shapes on the left side of the robot represent IR sensors used to determine distance and heading relative to the wall, and the single (green) triangle shape on the front represents an IR sensor used to determine the distance to the wall in the forward direction. The objective is to have the robot track the wall at a constant separation distance d_{sep}.

The logic used to find the wall is simple. The robot moves forward at a constant speed with $u = v_{\text{nom}}$ (the nominal wall tracking velocity is some percentage of the maximum velocity of each wheel) and $v = 0$, until the front sensor reads $d_f \approx 2d_{\text{sep}}$. Once this occurs, the robot turns clockwise slowly in place with control $u = 0$ and $v = -0.05v_{\text{nom}}$. The robot keeps turning until $d_{l,b} = d_{l,m} = d_{l,f}$. After this, the robot uses the sensors to keep track of distance and heading relative to the wall.

The third part of the lecture consists of designing and analyzing the actual controller running on the robot to track the wall. Students are exposed to a discrete-time proportional-

derivative wall tracking control given by

$$u_k = v_{\text{nom}}, \qquad v_k = k_p\,(x_k - d_{\text{sep}}) + k_d \frac{x_k - x_{k-1}}{\Delta}. \tag{5}$$

Students are asked to modify the MATLAB® program `robobrain.m`, created in Lecture 8, to include the feedback controller (5). The new function asks for the initial configuration x_0, y_0, θ_0 of the robot, the time step Δ, the number of iterations N, the desired separation distance d_{sep} and the control gains k_p, k_d, and outputs a vector with components x, y, θ and the elapsed time. An example simulation where this controller is employed is shown in Figure 3.

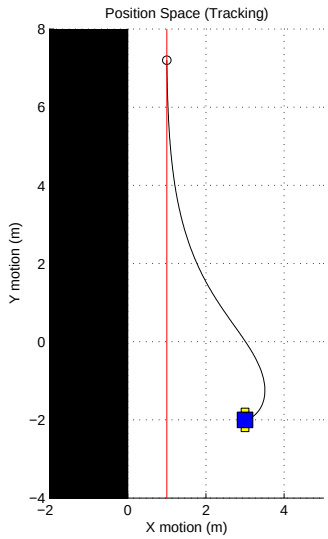

Figure 3. Simulation of feedback control for keeping a desired separation from the wall at a constant speed. Initial condition is $(x_0, y_0, \theta_0) = (3, -2, 0)$, nominal wall tracking velocity is $v_{\text{nom}} = 1$ and desired separation is $d_{\text{sep}} = 1$. Control gains are $k_p = 1/3$ and $k_d = 1$. Sample period is $\Delta = 0.05$.

To conclude the lecture, students are asked to tune the values of the control parameters k_p and k_d. The knowledge of linear algebraic tools would allow them to do this analytically. Since these are not assumed, they are encouraged to select some values a priori, use the

newly defined program to simulate the system, and modify their selection accordingly until the desired behavior is obtained. In our experience, it was at this point that students were somewhat disappointed with not being able to do something more rigorous to ensure that the control yields the desired wall tracking behavior. Guessing parameters and simulating the model was, in other words, lame. When told that more advanced math, specifically calculus and college-level linear algebra, would enable them to determine the control parameters analytically to get the desired result, the students were enthusiastic to learn these subjects.

Once the control parameter tuning task is completed, students are instructed to transcribe their parameter choices into the actual robot program, which is run in C. Our course finale consisted of each robot trying to track a curved indoor path. The students competed to see which robot could track the wall for the largest distance along the wall. Figure 4 shows experimental results for one of the successful cases.

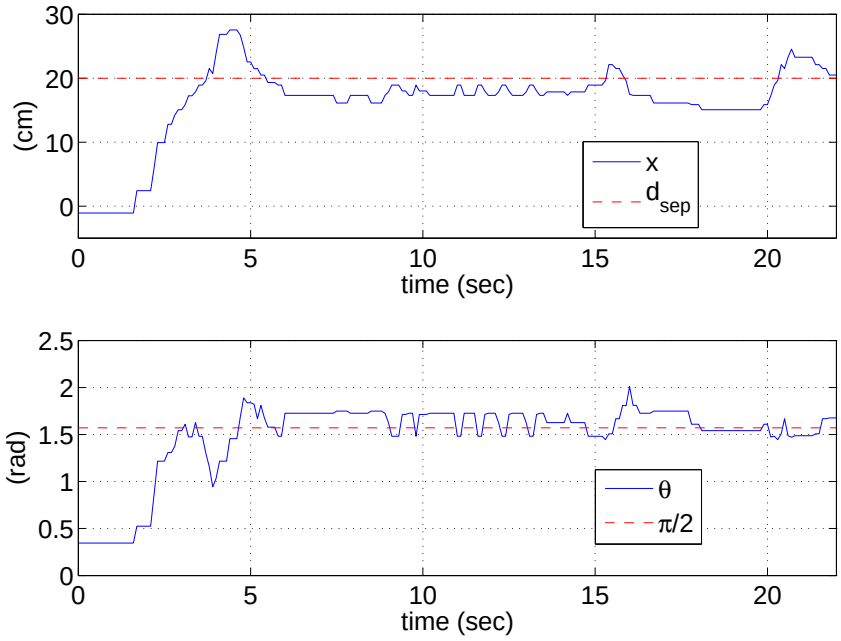

Figure 4. Experimental results of feedback control for keeping the robot at a desired separation ($x = d_{\mathbf{sep}}$) from the wall at a constant speed ($u_k = v_{\mathbf{nom}}$) and constant heading ($\theta = \pi/2$). The feedback control law (5) was turned on at time $t = 2$ seconds.

Implementation of the Course Material: COSMOS

In the summer of 2005, a four-week course based on the material was taken by a group of talented high school students, with a range of 9 to 11 in grade level. The course was part of the COSMOS program held at the UC Santa Cruz campus.

COSMOS (the California State Summer School for Mathematics and Science) is a selective four-week summer residential program for young scholars with demonstrated interest and achievement in math and science [12]. The program is located on four University of California campuses: Davis, Irvine, San Diego, and Santa Cruz. The student population of COSMOS 2005 at Santa Cruz was composed of 76 female and 75 male students, with 37% of the total being underrepresented. Each COSMOS student enrolls in one of several topical "clusters" for the duration of the program. Each cluster is comprised of two courses, each designed and instructed by UCSC faculty, lecturers, researchers and/or graduate students. Cluster 2 (out of 7 total clusters) was comprised of the courses "Nanotechnology" and "Making Robots and Making Robots Intelligent," the latter being our crash course in feedback control. Our course met for typically 90–120 minutes each day for 16 lecture days out of the four weeks, and minimal assignments outside of the classroom were required.

The overall experience was very positive and the lecture format worked extremely well. Students responded well to the introduction to MATLAB®, and our lecture model consisting of short theoretical explanations combined with simple computer exercises to reinforce the theory. The general introduction to the course at the very first day of class served them well to broadly understand the main ideas of feedback control. We often referred to this lecture during the course, placing the specific contents of the class into the big picture. We believe that students appreciated and were aided by the general introduction on day one and the

brief high-level introductions at the beginning of each subsequent lecture.

In future offerings of the course, we plan to integrate the `Robobrain` robots into the lectures sooner, instead of waiting until the final lecture. By our experience, we anticipate that presenting the students with hardware that they would eventually use would motivate them to learn the theoretical content of the course. Students would be eager to understand the math that will allow them to make the robot operate autonomously.

This experience will be turned into an introductory undergraduate course in the engineering curriculum at UC Santa Cruz. In the fall of 2006, the course "Computer Engineering 8 – Robot Automation: Intelligence through Feedback Control" will be offered, and introduce first-year undergraduate students to MATLAB®, programming, dynamics, feedback control and robotics using our lecture material. We believe that this course will serve as a great recruiting tool for bright students. Additionally, the experience will hopefully motivate them to learn the more advanced mathematics, programming and hardware courses available within the computer engineering, electrical engineering and applied mathematics and statistics curricula at UC Santa Cruz.

Feedback from Students

At the end of the course, we conducted an online survey about the satisfaction of students with the course. According to the students' evaluations and the parents' comments at the end of the course, the experience was as challenging and rewarding for the students as it was for us. Various parents expressed the feeling that the COSMOS experience will shape their kids' future careers.

We asked the students to express their opinions in 6 multiple-choice questions and 8 qualitative questions. Out of 17 enrolled students, we received 14 evaluations. Table 1 summarizes

the outcome to the quantitative questions.

	Poor	Fair	Satisfactory	Very Good	Excellent
Clarity and understandability	-	-	-	10	4
Preparation and organization	-	-	1	7	6
Course as learning experience	-	-	-	5	9
Will recommend course to buddies	-	-	1	1	12

Table 1. Quantification of student satisfaction with the course.

Below are some of the students responses to the qualitative questions:

- What did you like most about the course?
  - "It was a challenge so I never got bored. I also enjoyed discovering all of the practical applications of the material we were learning. It was also an experience I never could have had in high school."
  - "I liked most how first we learned the idea, then the math, then the application of it all. I really liked how it all came together eventually, it was really clear at the end."
  - "I liked working on the programming of the `Robobrain` and I guess that includes the work leading up to the `Robobrain`. The last day of class was amazing, but that was only because we had to work up to it. If the `Robobrain` was just handed to us than I don't think it would have been as good an experience."
- How much and in what ways did the handouts help your learn the material in this course?
  - "The handouts were key in the course — they explained and helped to reinforce the reasoning and purpose of all that was taught in the class. The handouts gave meanings and definitions to the terms, assigned tasks to help the student have a better understanding of the material, MATLAB®, and making robots intelligent."

  - "The handouts were very helpful. I liked how they were written and how they introduced the ideas, explained, and then had us do a task to learn them. Referring back to the handouts was helpful as well."

- Please tell us the concepts covered in the course that you understood the most and the least.

  - "Most of the concepts I had a good grasp on, although there were a few I was a little hazy about. Equilibrium, steady-state, fixed points, stability, etc., I understood really well. The only idea I really had any trouble on was towards the end with finding derivatives, because I'm not all that familiar with it."
  - "I understood all of the algebra and altering the equations very well. I also understood why we change the equations like we do. I got the graphing very well too, both writing the programs to produce the graphs and reading the graphs to tell what they meant. I understood the calculus the least, because i haven't had the class yet. Most of the derivative stuff I just hitched a ride on."
  - "The concepts covered in the course that I understood the most included the mathematical concepts and the entire necessity of feedback control for robotics. The concepts I understood least probably included much of the later math for `Robobrain`, which to me was a little rushed. I'm sure if we went over the math a few more days, I would be able to understand."

In the summer of 2006, we will be offering the same course within the COSMOS program.

Conclusions

We have summarized the contents of an introductory course to feedback control developed at

UC Santa Cruz. The course material, available at [7], is based upon fundamental concepts in dynamical systems, modeling, stability, robustness, and the design of feedback control laws. We have also reported our very positive experience in using the material in a summer course for motivated high school students during the summer of 2005. We believe that researchers from systems and control theory will find the material useful for a variety of activities, ranging from education and outreach to high school and undergraduate students, to interdisciplinary collaborations with scientists from other disciplines.

An interesting study performed in [13] examines how novices learn feedback control concepts, an in particular how they learn a perspective on control that includes abstraction, so that the concepts are not tied to a specific problem description. The study involved the use of a GUI-based software that allows the students to connect and simulate a set of standard functional objects (e.g., sensor, actuator, set point unit) that exist in many control applications. As part of an introductory engineering course, the software was made available to a group of high school students for $\sim$45 minutes per day for 5 days. A key finding of the study is that "the idea of a signal is a core concept in an experts conception of feedback systems, and is intricately tied to the definition of the functional components that make for a feedback control system." It is also reported that by the end of the course, students still had trouble with the concept of signals.

In our summer course, we used MATLAB® and described vectors (which most had seen in a math or physics course) as a sequence of measurements taken at snapshots of time. It did take most students a few days and several exercises to connect the "list of numbers" (vector) in MATLAB® with the evolution of a model of a real physical system. Once they grasped this idea, we were able to describe the use of models for prediction and control design. We never formally introduced the word signal, and did not attempt to generalize the vector description

to more general scenarios (continuous time, for example), as this seemed unnecessary. As a future direction, our material, based largely on algebra and MATLAB®-based examples, and a GUI-based approach to control could be combined to help students comprehend the concepts.

Acknowledgments

We sincerely thank Mr. Bill Thompson, the high school instructor assigned to our cluster, for his help and enthusiasm. During the preparation of the materials for the course, Bill was our constant source of guidance in the difficult task of approaching the minds of 17 high school students. His advice on how to best shape our materials to reach them were always very helpful.

We thank Noah Wilson and Daniel Garalde, who assembled the `Robobrain` robots. Noah was also responsible for the C programs that students used to implement the feedback controller designed in class.

We would also like to thank all of the students who took the course with us and provided invaluable feedback. Their enthusiastic response to the course materials far exceeded our best expectations. Teaching feedback control to them was truly a joyful experience.

References

[1] P. Antsaklis, T. Basar, R. DeCarlo, N. H. McClamroch, M. Spong, and S. Yurkovich, editors. *NSF/CSS Workshop on New Directions in Control Engineering Education*.

National Science Foundation and IEEE Control Systems Society, 1998. Available at http://robot0.ge.uiuc.edu/∼spong/workshop.

[2] R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE Control Systems Magazine*, vol. 23, no. 2, pp. 20–33, 2003.

[3] Caltech course Control and Dynamical Systems 101/110, Analysis and Design of Feedback Systems, http://www.cds.caltech.edu/∼murray/cds101/.

[4] K. J. Astrom and R. M. Murray. *Analysis and design of feedback systems.* Preprint, 2003.

[5] National Research Council. National Science Education Standards. Washington, DC: National Research Council, 1996.

[6] Committee on Prospering in the Global Economy of the 21st Century: An Agenda for American Science and Technology. *Rising Above The Gathering Storm: Energizing and Employing America for a Brighter Economic Future.* National Academy of Sciences, National Academy of Engineering and Institute of Medicine, The National Academies Press, 2005. Available at http://www.nap.edu/catalog/11463.html.

[7] All course material is available at http://www.soe.ucsc.edu/˜jcortes/controlcrashcourse/.

[8] Autonomous vehicle photo of Caltech's 2004 DARPA Grand Challenge entry, courtesy of Richard Murray.

[9] S. H. Strogatz. *Nonlinear Dynamics and Chaos.* Westview Press, Cambridge, 1994.

[10] Advertisement for MATLAB® and SIMULINK® by The MathWorks Co. (back cover page), *IEEE Spectrum*, August, 2005. More information available at http://mathworks.com/mbd.

[11] K. Sigmon and T.A. Davis, MATLAB® *Primer, Sixth Ed.*, CRC Press, 2002.

[12] The California State Summer School for Mathematics and Science (COSMOS), http://www.ucop.edu/cosmos. At UC Santa Cruz, http://epc.ucsc.edu/cosmos/.

[13] J. Ma, "A case study of student reasoning about feedback control in a computer-based learning environment," In *Proc. 29th ASEE/IEEE Frontiers in Education Conf.*, San Juan, Puerto Rico, 1999, pp. 12d4-7–12d4-12.

Jorge Cortés received the Licenciatura degree in mathematics from the Universidad de Zaragoza, Spain, in 1997 and his Ph.D. degree in engineering mathematics from the Universidad Carlos III de Madrid, Spain, in 2001. He is an assistant professor in the Department of Applied Mathematics and Statistics at UC Santa Cruz. Prior to coming to UCSC in 2004, he held postdoctoral positions at the Systems, Signals and Control Department of the University of Twente, and at the Coordinated Science Laboratory of the University of Illinois at Urbana-Champaign. His research interests focus on mathematical control theory, distributed motion coordination for groups of autonomous agents, and geometric mechanics and geometric integration, and he is the author of the book "Geometric, control and numerical aspects of nonholonomic systems" (Springer Verlag, 2002).

William B. Dunbar (dunbar@soe.ucsc.edu) earned a B.S. degree in engineering science and mechanics from Virginia Tech in 1997, an M.S. degree in applied mechanics and engineering science from UC San Diego in 1999, and a Ph.D. in control and dynamical systems from Caltech in 2004. He is currently an assistant professor in the Department of Computer Engineering at UC Santa Cruz. His research areas include distributed model predictive control, consistent approximations of optimization algorithms, and supply chain management problems. All correspondence regarding this article can be addressed to: William B. Dunbar,

University of California, 1156 High Street, Mail Stop: SOE3, Santa Cruz, CA 95064 USA. Fax: 831-459-4829.