RC23940 (W0604-118) April 24, 2006
Computer Science

IBM Research Report

Teaching Control Theory to Computing Practitioners

Yixin Diao, Joseph L. Hellerstein, Sujay Parekh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. Ithas been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distributionoutside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g, payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

TEACHING CONTROL THEORY TO COMPUTING PRACTITIONERS

Yixin Diao, Joseph L. Hellerstein, Sujay Parekh

IBM Thomas J. Watson Research Center
Hawthorne, New York 10532
email: {diao, hellers, sujay}@us.ibm.com

Abstract: Although feedback control is an essential part of computing systems, computing practitioners rarely have knowledge of control theory. Further, computing practitioners interested in learning control theory are faced with the problem that introductory texts contain unfamiliar examples and unfamiliar mathematics. To remedy this situation, we have developed a tutorial, book, and spreadsheet based laboratory simulations to teach control theory to computing practitioners. Key considerations in the pedagogy are to use examples drawn from computing systems, provide a short, self-contained introduction to discrete time modeling and dynamic response, and incorporate case studies of how control theory has been used in commercial computing systems.

Keywords: computing practitioner, spreadsheet simulation, control of computers

1. INTRODUCTION

Feedback control is an essential part of computing systems. For example, program execution rates can affect execution priorities, which in turn impact future execution rates (Bach, 1996). Another example is congestion control in networks where round trip delays are used to adjust data transfer rates, which in turn impact future round trip delays (Tannenbam, 2002). Given the importance of feedback design in computing systems, it is therefore surprising that computing practitioners know very little about control theory (probably due to the historical emphasis of computer science on algorithm design). This paper describes efforts we have undertaken to teach control theory to computing practitioners.

Existing pedagogy for introducing control theory is well evidenced by the structure of text books such as (Ogata, 1997) and (Franklin *et al.*, 1994). These introductory control theory texts provide examples drawn from electrical and mechanical engineering such as dash pots, electrical circuits, and cruise control. The texts begin with a brief review of continuous time modeling and dynamic response (e.g., dynamic models represented by differential equations, Laplace transform for transforming the above into an algebraic form, and response versus pole locations). The pedagogy continues with feedback controller properties, root-locus analysis, frequency domain analysis, and state space design. Sometimes, there is a chapter on digital control at the end of the book.

There are several problems with using this pedagogy to teach control theory to computing practitioners. First, most computing practitioners have little background in mechanical or electrical engineering, and so the examples provide little intuition. Second, while there have been computer science research papers that use continuous time models (e.g., (Hollot *et al.*, 2001)), computing practitioners are much more comfortable with discrete time models. However, discrete time constitutes a small (or non-existent) part of the cur-

rent pedagogy for an introductory course. Further, computing practitioners rarely have background in dynamic models and dynamic response, and so the existing pedagogy is not approachable without a pre-requisite class. Next, there needs to be an interactive environment in which computing practitioners can use familiar tools to explore the effects of plant structure and control design on control objectives, especially stability, settling time, and bias. Last, getting computing practitioners to adopt methodologies based on control theory requires examples of how these methodologies have been used in for solving representative problems in widely used commercial products.

From the foregoing, we identify several requirements for the pedagogy of an introductory course on control theory for computing practitioners:

(1) The examples used should be drawn from computing systems.
(2) The dynamic models should be in discrete time.
(3) Any mathematical background should be self-contained. In particular, there must be a self-contained, intuitive introduction to dynamic response.
(4) There should be an interactive environment for exploring the effects of plant structure and control design. This environment should be familiar to computing practitioners and be widely available.
(5) There must be material that demonstrates that the techniques are useful in commercial computing systems. Otherwise, practitioners will have little interest.

Over the last two years, we have developed a pedagogy for teaching control theory to computing practitioners. On two occasions, we taught a one semester class on control theory in the Computer Science Department at Columbia University. In conjunction with this, we have written a book on control theory for computing scientists (Hellerstein *et al.*, 2004). And, more recently, we have refined the material into a three hour tutorial that has been presented at two leading computer science conferences ACM SIGMETRICS and IEEE/IFIP Integrated Management.

The classes typically started with 15 to 20 students and then dwindled to a half or a third of the original size by the fourth lecture. Attendance at the tutorials was larger—typically thirty students, all of whom stayed for the entire session and were fairly interactive. The comments we have received indicate that most were interested in knowing what control theory is and its relevance to computing systems. Only a few have subsequently applied control theory to computing systems.

The remainder of this paper is organized as follows. Section 2 discusses control challenges in computing systems. Section 3 provides details of the tutorial we have used, especially how we addressed the requirements listed above. Section 4 describes a spreadsheet based environment for computing practitioners to explore control problems. Our conclusions are contained in Section 5.

2. CONTROL CHALLENGES IN COMPUTING SYSTEMS

Recently, there has been an explosive growth in the use of computing systems to deliver information services, such as providing access to a bookstore on the Internet. Increasingly, **service level objectives (SLOs)** are used to specify the desired end-user performance characteristics of the information services delivered. For example, an online bookstore might have an SLO requiring that requests for the status of an order be satisfied in less than 1 second. Here, "time in which an order is satisfied" is a service level metric. Thus, computing elements or resources are allocated with considerations for SLOs and the cost of the resources. These costs include hardware, software, and people to operate them.

Control problems in computing systems arise from two trends: (1) the rising cost of operating computing systems to achieve SLOs, and (2) the widespread use of the Internet for connecting businesses with customers. While the cost of hardware and software has declined dramatically over the last four decades, the cost of operating computing systems continues to increase. Industry analysts estimate that the cost of operations accounts for 60% to 90% of the total cost of ownership of computing systems (Humphreys and Melenovsky, 2003), largely because of the human involvement required for operations. One part of the operations cost is configuring software systems (often consisting of multiple products), a task that frequently involves adjusting configuration parameters such as limits on the number of connected users, buffer pool sizes, and the number of concurrent threads in response to changes in workloads (the volume of requests made by end users and the resource demands of these requests). In essence, configuration parameters are the actuators by which the system is controlled. It is complicated to manipulate these actuators to achieve SLOs. As a result, human experts are often needed, which increases costs.

Businesses often choose to outsource their computing systems to reduce labor costs. Outsourcing means that an organization such as EDS or IBM (the service provider) provides the computing systems and skilled professionals for operations. In outsourcing, the customer specifies SLOs

that quantify the service expected from the service provider. The service provider must enforce SLOs so that customers receive the correct service levels. In one way, enforcing SLOs is a regulation problem in that resource allocations are adjusted to achieve desired values of service level metrics. However, enforcing SLOs is also an optimization problem in that service providers want to minimize the cost of the resources needed to meet SLOs.

A second trend facing computing systems is the role that businesses have on the Internet in supplying information about the company's goods and services and in selling these goods and services. The broad reach of the Internet, which has great appeal for both of these goals, exposes computing systems to large variations in incoming requests called flash-crowds that can vary by a factor of twenty (LeFebvre, 2001). Handling flash crowds is a disturbance rejection problem.

The foregoing characteristics of computing systems result in several challenges in applying control technologies. First, computing systems consist of many resources that are shared among multiple end-to-end services (Aman *et al.*, 1997). Thus, enforcing an SLO for a service often means regulating the loads applied to the resources used to deliver the service. For example, service levels are often based on end-to-end metrics such as end-to-end response times. However, actuators typically control an individual resource such as CPU execution rates, the size of memory pools, and the number of threads allocated. Thus, the most common way to achieve an SLO is to establish response time targets for each resource, and have controllers for each resource operate independently to achieve these targets. An example of this is an electronic storefront that consists of an application server and a database server in which there is one response time target for the application server and another response time target for the database server. This decomposition of an end-to-end SLO into response time targets for individual resources results in a kind of supervisory control.

Second, as noted earlier, computing systems exhibit tremendous variability. One source of variability is having Internet-facing systems that experience rapid convergence of interest for millions of users, such as for weather, news, and merchandise. Another source of variability is the ease with which software based systems can be changed. Since change is easy, changes to software systems are frequent, such as applying software patches. These changes can result in dramatic variations in performance characteristics.

A third challenge arises from the fact that computing practitioners approach control design in an ad hoc way. As a result, it is common for computing systems to have unnecessarily complex dynamics that are difficult to control.

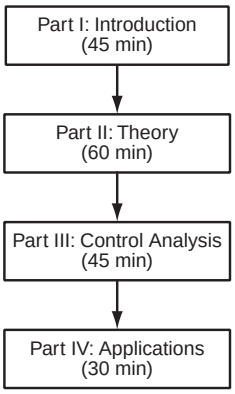


Fig. 1. Structure of the tutorial for teaching control theory to computer scientists.

More details on control challenges in computing systems can be found in (Abdelzaher *et al.*, 2003).

3. PEDAGOGICAL APPROACH

This section describes our pedagogy for teaching control theory to computing practitioners. The discussion is structured along the lines of a three hour tutorial that we developed to provide computing practitioners with a quick introduction to control theory.

Figure 1 depicts the structure of the tutorial. There are four parts: introduction, theory, control analysis, and application to commercial systems. Our experience is that the tutorial provides practitioners with sufficient background to do control analysis for many design problems in computing systems. The tutorial assumes a very modest mathematical background, just knowledge of the geometric series.

Part I, the introduction, has two purposes. The first is to provide intuition about control theory and its value. The second is to introduce key concepts, especially the basics of block diagrams (e.g., plant, controller, reference input, control input, and measured output), and control objectives (especially, stability, short settling times, accuracy, and minimal overshoot). Each of these concepts is new to the computing audience. Hence, we use simple examples to show how a computing problem is represented in a block diagram. An example using IBM Lotus Domino Server (Hellerstein *et al.*, 2004) is depicted in Figure 2(a). The introduction of control objectives faces a similar challenge: in computer science, the commonly discussed objectives relate to system complexity, performance, fault tolerance, compatibility and so on. It is therefore important to discuss (through examples) how these system-level objectives relate to control

objectives. We also distinguish between feedforward control (which is often used in computing systems, e.g., based on queueing theory models) and feedback control.

Part II, theory, provides an intuitive and self-contained introduction to discrete time modeling and dynamic response. This is the longest part of the tutorial, taking about an hour to present. There are three subparts: signals, transfer functions, and composition techniques. The signals subpart introduces the z-Transform in an intuitive way, without discussion of continuous time. This is accomplished by presenting the z-Transforms as a way to represent signals as a sum instead of a list. For example, consider a finite length signal that has value 0.5 at time 0, 0.25 at time 1, and 0.125 at time 2. In the time domain, we have a list representation $\{0.5, 0.25, 0.125\}$. In the z domain, we use the summation $0.5z^0 + 0.25z^{-1} + 0.125z^{-2}$. Next, we show that representing signals as sums in the z domain allows us to add signals, multiple them by a constant, do time shifts, and express time delays. Communicating these concepts usually takes less than ten minutes. After students have a clear understanding of finite length signals and operations on them, we introduce infinite length signals expressed using the geometric series.

The second subpart of Part II addresses transfer functions. We define transfer functions in terms of operations on signals. A very effective progression is to first show a transfer function with a constant gain, then introduce a time delay, and then the addition of several time delay terms with different gains. As with signals, we first introduce transfer functions with a finite number of terms, and then we show how to incorporate an infinite number of terms. We connect this with the geometric series, which in turn leads to a discussion of poles. The most important insight here is the relationship between poles and stability, settling times, and oscillations. To simplify matters, we only consider first order systems so that no review of complex numbers is required.

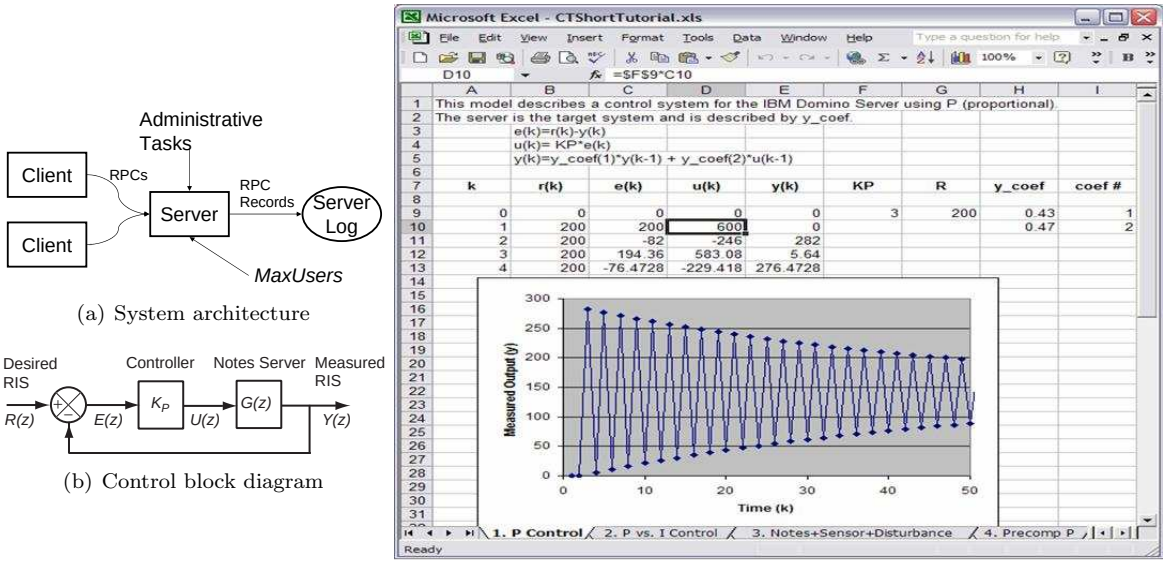
The last subpart of Part II is about composition of systems. This is fairly easy material to communicate since much of computer science is about composing components. We do not address convolution directly. Rather, we use simple algebra to show that the z-Transform of two plants in series is the product of their z-Transforms. This in turn leads to a discussion of reducing block diagrams, and a derivation of several transfer functions for feedback control systems.

Part III addresses control analysis. This is done by introducing a series of design problems based on the IBM Lotus Domino Server (Hellerstein *et al.*, 2004). We begin by introducing proportional and integral control. The control performance of P and I controllers is compared using a spreadsheet simulation described in Section 4. For pedagogical purposes, we assume that since the P controller has shorter settling times, we should figure out how to make it more accurate and reduce the overshoot. To address the former, precompensation is introduced, and the results developed in Part II are used to determine the gain of the precompensator. To reduce overshoot, we introduce filters, and discuss several issues in their design and placement in the control system. Last, we show that the characteristics of P and I controllers can be combined using a PI controller, although the design is more complicated because we must select a gain for the P controller and another gain for the I controller.

Part IV provides applications of control theory to commercial computing systems. We discuss two examples in which IBM's database management product has incorporated control theory based designs. The first example is utilities throttling, a capability for regulating high overhead administrative programs that are essential to the operation of a database management system (e.g., backup, restore) (Parekh *et al.*, 2004). The second example is self-tuning memory management, a facility for optimizing the allocation of memory in database management systems (Diao *et al.*, 2004). Of particular interest in the latter example is the use of regulatory control to solve an optimization problem.

The tutorial is largely based on a book we have written to teach control theory to computing practitioners (Hellerstein *et al.*, 2004). We have used the book to teach a one semester class on control theory for Computer Science graduate students at Columbia University. As with the tutorial, the book uses discrete time models and examples drawn from computing systems. The book augments the tutorial in the following ways:

- The introductory material includes more examples of control of computing systems, such as streaming media and caching with differentiated service.
- There is an extensive development of modeling techniques, a critical consideration for computing practitioners to apply control theory. There is a chapter devoted to model construction using system identification, and the book includes a discussion of first principle models (especially based on concepts from queueing theory).
- A wider range of discrete time linear models is included. In addition to a discussion of first order systems, there is a separate discussion of transient behavior of higher order systems (including a brief review of complex



(a) System architecture

(b) Control block diagram

(c) Spreadsheet simulation

Fig. 2. A control system that regulates RPCs in System (RIS) for the IBM Lotus Domino Server.

variables), and there is a chapter devoted to state models. Computing system examples are used throughout.

- Control design is discussed at length, especially pole placement, root locus, and empirical methods.
- There is a chapter on advanced control techniques such as adaptive control, stochastic control, and fuzzy control.

4. INTERACTIVE LAB

This section describes an interactive environment that provides computing practitioners with a way to explore the implications on control performance of changes in the plant and control design. Foremost, we intended that this environment be easy to use and enable practitioners to gain a deeper understand of control systems. MATLAB is an excellent tool for accomplishing these goals, and there are excellent interactive programs for providing an introduction to control theory (e.g., http://www.engin.umich.edu/group/ctm/). However, few computing practitioners have access to MATLAB, and it is an expensive product (although there are significant educational discounts).

Our approach has been to use a spreadsheet to simulate and visualize linear system dynamics. Spreadsheets have appeal since they are widely available, and the user interface is well understood. Further, linear system dynamics are easily expressed in spreadsheets by having time move left to right and top to bottom.

We illustrate our interactive lab using the IBM Lotus Domino Server. The Domino Server provides email capabilities to user machines running a client application. The user interacts with his client application to write, send, read, and organize their email. These interactions are translated into Remote Procedure Calls (RPCs) that are sent to the Domino Server. If many users are doing email at the same time, then the server may be processing many RPCs concurrently. An important metric here is the number of RPCs in the Domino Server, which we denote by **RIS**. Early versions of the Domino Server had poor performance when the RIS was too high, and so administrators sought to reduce RIS. Unfortunately, there is no way to regulate RIS directly. However, administrators could control the number of *connected* users by adjusting the tuning parameter MAXUSERS. Connected users is approximately equal to RIS during busy times such as mid-morning. But connected users is much larger than RIS during lunch and other light periods. Thus, controlling RIS by adjusting MAXUSERS required on-going intervention on the part of the administrators, which was undesirable.

Figure 2(b) displays a feedback loop to automatically regulate RIS. Administrators specify the desired RIS, and the controller adjusts MAXUSERS so that measured RIS equals desired RIS. Figure 2(c) displays a spreadsheet simulation of Figure 2(b) for proportional control. The first five rows give text description on model information. The simulation area starts from Row 9, and the column description is given in Row 7. The leftmost column in Figure 2(c) (i.e., Column A) is time k. The next four columns (i.e., B-E) are signals: reference input $r(k)$, control error $e(k)$, control input $u(k)$, and measured output $y(k)$. The remaining columns (i.e., F-I) are parameters used in spreadsheet calculations: R is the reference input, K_P is the control gain, and the vector

y_coeff describes a first order model of the plant $y(k) = \text{y\_coeff}(1)y(k-1) + \text{y\_coeff}(2)u(k-1)$. The spreadsheet simulation is constructed as follows.

(1) The initial signal values are specified. As we see in cells B9 through E9, these values are 0.
(2) The time domain difference equation used to compute each signal are entered. This is done on Row 10. For example, Cell D10 uses the control error to calculate a normalized value of MAXUSERS using the formula F9*C10, where F9 is the absolute address of the control gain and C10 is the relative address of the control error.
(3) Iterations of the simulation are constructed. This is achieved by copying Row 10 to subsequent rows.
(4) The plot of the simulation results is constructed. For example, the plot in Figure 2(c) displays the time-varying value of the measured output $y(k)$ in column E.

We use the spreadsheet to explore how control gain K_P affects control performance, especially stability, accuracy, settling times, and overshoot. This is done by changing the value in Cell F9. First, we change the value in F9 to -1. This makes the control loop unstable. When teaching the tutorial, we ask students to explain what happened in qualitative terms. The kind of answer we look for is: "A positive control error should result in an increase in MAXUSERS so that measured RIS increases to desired RIS. However, with a negative control gain, a positive control error results in a *decrease* in MAXUSERS that increases control error causing MAXUSERS to decrease further."

Next, we explore the relationship between K_P and settling time, accuracy, oscillations and overshoot. When F9 value is 0.1, there is no overshoot or oscillation. However, the settling time is long. As F9 value is increased, settling time becomes shorter, but there is overshoot and eventually oscillation. Much above $K_P = 3$, the system is unstable. After changing the value of F9, all of the above effects can be immediately visualized through the plot as well as the corresponding columns. The spreadsheet can also be used to study the effects of changes in the plant via cells H9 and H10. The tabs at the bottom of Figure 2(c) are other spreadsheets that correspond to the systems studied in Part III of the tutorial.

5. CONCLUSIONS

Feedback control is an essential part of computing systems. Unfortunately, computing practitioners rarely have knowledge of control theory, and existing texts are poorly suited for this audience. To address this, we have developed a tutorial, book, and spreadsheet based laboratory simulations to teach control theory to computing practitioners. Key considerations in the pedagogy are to use examples drawn from computing systems, provide a short, self-contained introduction to discrete time modeling and dynamic response, and incorporate case studies of how control theory has been used in commercial computing systems.

REFERENCES

Abdelzaher, Tarek F., John A. Stankovic, Chenyang Lu, Ronghua Zhang and Ying Lu (2003). Feedback performance control in software services. *IEEE Control Systems Magazine* **23**(3), 74–90.

Aman, Jeffrey, Catherine K. Eilert, David Emmes, Peter Yocom and Donna Dillenberger (1997). Adaptive algorithms for managing a distributed data processing workload. *IBM Systems Journal* **36**(2), 242–283.

Bach, Maurice J. (1996). *The Design of the UNIX Operating System*. Prentice-Hall Software Series.

Diao, Yixin, Joseph L. Hellerstein, Adam Storm, Maheswaran Surendra, Sam Lightstone, Sujay Parekh and Christian Garcia-Arellano (2004). Incorporating cost of control into the design of a load balancing controller. In: *Real-Time and Embedded Technology and Application Systems Symposium*.

Franklin, G. F., J. D. Powell and A. Emami-Naeini (1994). *Feedback Control of Dynamic Systems*. third ed.. Addison-Wesley. Reading, Massachusetts.

Hellerstein, Joseph L., Yixin Diao, Sujay Parekh and Dawn M. Tilbury (2004). *Feedback Control of Computing Systems*. John Wiley & Sons.

Hollot, C. V., V. Misra, D. Towsley and W. B. Gong (2001). A control theoretic analysis of RED. In: *Proceedings of IEEE INFOCOM '01*. Anchorage, Alaska. pp. 1510–1519.

Humphreys, John and M. Melenovsky (2003). Enabling business agility through server blade technology. Technical Report P508.872.8200. IDC.

LeFebvre, William (2001). CNN.com: Facing a world crisis. In: *Proceedings of the 15th Conference on Systems Administration*.

Ogata, Katsuhiko (1997). *Modern Control Engineering*. 3rd ed.. Prentice Hall.

Parekh, Sujay, Kevin Rose, Yixin Diao, Victor Chang, Joseph L. Hellerstein, Sam Lightstone and Matthew Huras (2004). Throttling utilities in the ibm db2 universal database server. In: *American Control Conference*.

Tannenbam, A. (2002). *Computer Networks*. 4th ed.. Prentice Hall.