COMPUTER CONTROL OF MACHINES AND PROCESSES

George A. Perdikaris, Ph.D.
University of Wisconsin-Parkside

Abstract

A method is presented for controlling machines and processes by a microcomputer. Examples of a motor drive plant (machine) and a heating plant (process) are presented. The computer controlled systems are designed and simulated using the language SIMULINK.[1] The motor control system is implemented in the laboratory. Results obtained from computer simulation are compared with laboratory findings.

I. Introduction

Industrial automation incorporating computers is becoming increasingly important in the production of goods and services. Automation systems are used to make paper, metal, wood, plastic, and other types of products; similar systems are also used to control the speed and fuel injection of an automobile or the temperature and humidity of an office building.

The basic premise of modern automation systems is efficient manufacturing. What this means is that manufacturers can bring new products to the market faster, keep inventories low, make products of consistently high quality, and have flexible facilities which allow for the manufacture of a variety of products. Furthermore, such systems help improve communication within the company as well as with its customers and suppliers.

It is a highly sophisticated job to design and build automation systems that operate and coordinate manufacturing processes. Quite often, such systems involve the application of computers to control the speed or position of the shaft of a motor, in real time. The “spinning” of motors intelligently is a high-tech marvel, fundamental to quality manufacturing. In the metal processing industry, for instance, the doors of refrigerators, washers, and dryers are formed out of steel, which is cut to size from large coils of metal and then pressed into shape by large presses, usually controlled precisely by computers. In the paper, plastics, or wood industries, as another example, consumer products ranging from toilet paper and baby diapers to office furniture and automobiles are also made by controlling machines and processes by computers.

This paper presents practical and systematic methods for modeling, simulation, and real-time control of two types of industrial systems. First, a method is presented for designing a computer controlled system that can be used to control the speed or position of a motor drive plant. Results obtained by computer simulation are verified experimentally. Next, a method is presented for designing a temperature (process) control system. The Ziegler-Nichols method for tuning the relevant proportional-plus-integral (PI) controller is illustrated and verified by computer simulation.

[1] SIMULINK is a registered trademark of MathWorks, Inc.

II. Computer Controlled Systems

Controlling a plant, machine or process, by a computer implies that the computer is an integral part of a feedback control system. In this configuration, the computer samples the response of the plant at certain instants of time and compares its findings with prescribed values stored internally. Any deviation from the desired behavior initiates corrective action as dictated by the control algorithm (controller), which is also stored in the computer.

When designing a digital control system, it is often practical and convenient to design the analog prototype control system first. This motivation comes from the fact that successful designs of analog controllers are well established. The analog control system is then converted into its digital counterpart by adding appropriate interfaces and adjusting the controller gains.

A closed-loop digital control system is shown in Fig. 1. As can be seen from the diagram, digital-to-analog converter (DAC) and incremental encoder or motor pulse generator (MPG) interfaces have been assumed. Specifically, the output of the digital controller (manipulation) is in digital form and must be converted to analog form before it can be communicated to the analog plant. A device that accomplishes this is the DAC or D/A. On the other hand, it is common to control the speed (velocity) and/or position (displacement) of the shaft of a motor by using a transducer that converts the shaft rotation into a signal that can be read by the computer. A transducer that does this is the MPG.

The shaft velocity is determined by counting pulses from a device register every T seconds, where T is the control sampling time. The signal from the pulse generator is proportional to the change in position from the previous sample. Thus, the pulse generator acts as a differentiator. For instance, if the input to the MPG represents position in radians (rad), the units of its gain are (pulses/rev=pulses/2πrad). Therefore, after differentiation, the velocity signal c(k) is (pulses/T). Specifically, a 2000 pulses/rev MPG has a gain given by

$$K_{dt} = \frac{\#\text{pulses}}{\text{rev}} = \frac{2000(pulses)}{2\pi(rad)}$$

A SIMULINK simulation diagram for the overall control system is shown in Fig. 2. Using the graphical models, the wordlength of interface devices such as the DAC and MPG can be varied interactively and relevant system responses can be observed graphically or stored for further evaluation. The designer can also tune (interactively) controllers such as the proportional-integral-derivative (PID), integral with proportional-derivative-feedback-plus-feedforward (PDFF) – shown in Fig. 1, or other types of controllers [1][2]. The computer simulation diagram of Fig. 2 includes blocks for modeling the DAC and MPG interface devices as well as a block for a trapezoidal profile generator.

A. Modeling an Industrial Motor Plant

It is assumed that the position of the motor shaft is monitored by an incremental encoder (MPG), which returns change of position (i.e., velocity) per sampling time T; also, the controller output is communicated to the analog plant via a DAC.

A motor drive plant is often modeled by a double-integrator transfer function

$$G_p(s) = \frac{\Theta(s)}{M(s)} = \frac{K_m}{s^2} \tag{1}$$

where the analog position plant gain K_m depends on motor drive parameters. If the DAC and MPG gains K_{da} and K_{dt} are taken into consideration, the digital position plant transfer function is found to be

$$\frac{\Theta(z)}{M(z)} = Z[\frac{1-e^{-sT}}{s}\frac{K_\theta}{s^2}] = \frac{(\frac{K_\theta T^2}{2})(z+1)}{(z-1)^2} \tag{2}$$

where the position plant gain K_θ is given by

$$K_\theta = (K_{da})(K_m)(K_{dt}) \tag{3}$$

K_θ can be calculated from given motor drive and interface parameter values or it can be determined experimentally.

The digital velocity plant transfer function can be determined by differentiating the digital position plant transfer function. That is,

$$\frac{\Omega(z)}{M(z)} = \frac{(\frac{K_\theta T^2}{2})(z+1)(z-1)}{z(z-1)^2} = \frac{(\frac{K_\omega T}{2})(z+1)}{z(z-1)} \tag{4}$$

where K_ω is the digital velocity plant gain defined by

$$K_\omega = TK_\theta \tag{5}$$

Like K_θ, K_ω can also be determined analytically from motor parameter values or, better yet, it can be determined experimentally. K_ω represents the slope of the unit-step response of the motor drive plant, which is normally a ramp function.

B. Velocity and Position Control

The motor drive plant is a PWM-powered direct current (DC) motor with current-loop feedback. The experimentally measured open-loop velocity gain is K_ω=5.35, which, assuming a sampling time of T=0.005 sec, corresponds to a position loop gain of K_θ=1070.

The plant is controlled according to the control strategy of Fig. 1. Note that the block marked "software" in Fig. 1 denotes software integration if the value of the parameter K_r=1. If K_r=0, on the other hand, the software integrator is essentially bypassed, which is required if velocity,

rather than position, is to be controlled. Using the PDFF algorithm, instead of the more conventional PID, the "software' integrator can also be changed into a first-order lag term by adjusting K_r in the range if $0< K_r <1$. This can prove useful if the position plant model is not a double integrator, but a first-order lag followed by a single integrator. Another advantage of the block diagram of Fig. 1 is that it can be used to control velocity or position using velocity command and feedback. It can be easily modified, however, for position command and/or feedback.

For velocity control, the method chosen for tuning the digital controller uses the Bessel filter form described in Reference [1]. Using this tuning criterion and assuming a system bandwidth of ω_n=50 rad/sec, with a sampling time of T = 0.005 sec, we obtain the controller gains

$$K_p = \frac{\sqrt{3}\omega_n}{K_\omega} = 16.187, \quad K_i = \frac{T\omega_n^2}{K_\omega} = 2.34$$

The other gains are K_v=0, K_d=0, and K_a = 0.

The motor control system was implemented in the laboratory using a Pentium microcomputer. The relevant signals were "captured" in real-time and downloaded into a data file for plotting. Simulation and experimental results are shown in Fig. 3.

Note that the 200 (pulse/T) for the step size of the trapezoidal profile shown in the figures corresponds to 300 RPM. That is, since the rated speed of this motor is 1000 RPM, and the sampling time is T=0.005 sec, the rated speed in (pulses/T) becomes

$$1000(\text{RPM})8000(\text{pulses/rev})\frac{0.005 \text{ sec}}{\text{T}} = 667 \text{ (pulses/T)}$$

Thus, for 300 RPM, instead of 1000 RPM, the speed becomes 200 (pulse/T).

For position control, the criterion chosen for tuning the digital controller is the ITAE filter form, also described in Reference [1]. Using this tuning criterion and assuming a position bandwidth of ω_n=30 rad/sec, with T = 0.005 sec, we obtain the controller gains

$$K_p = \frac{2.15\omega_n^2}{K_\theta} = 1.81, \; K_i = \frac{T\omega_n^3}{K_\theta} = 0.126, \text{ and}$$

$$K_d = \frac{1.75\omega_n}{TK_\theta} = 9.8; \text{ also, } K_v=0, \text{ and } K_a = 0.$$

Simulation and experimental results are shown in Fig. 4, left and right, respectively.

C. An Algorithm for Process Control

When designing an algorithm to control a process such as the temperature of a building, the main objective is to find a set of tuning values so that the control system meets transient and steady-state specifications for tracking reference inputs and/or for rejecting disturbances. Some applications require a relatively slow response without overshoot; in other cases, speed of response is essential and oscillations may not be a problem. The tuning problem may be approached systematically, as in the previous motor control examples, but every problem, new or old, brings with it new challenges for the designer, particularly in the digital control field.

A method suitable for tuning control algorithms such as the PID was proposed by Ziegler and Nichols in 1942 [3]. It is called the ultimate method because it requires finding the so-called ultimate gain (K_u) and the ultimate period (T_u) for the control loop. To find the ultimate gain K_u, we use the following approach. With only proportional control in operation, increase the proportional gain until the response oscillates, that is, until the system becomes marginally stable. The maximum value of the proportional gain for which the system oscillates is the ultimate gain K_u. The period of the sustained oscillations is the ultimate period T_u.

Ziegler and Nichols used the ultimate gain and ultimate period values to calculate controller settings for the following form of the PID controller, which is sometimes referred to as the ideal noninteracting PID controller,

$$m(t) = K_C[e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}] \tag{6}$$

A more conventional form of the PID controller, also known as the three-mode controller or the ideal parallel PID controller, is given by

$$m(t) = K_P e(t) + K_I \int e(t)dt + K_D\frac{de(t)}{dt} \tag{7}$$

Thus, the proportional gains are the same for both controllers and the integral and derivative gains are related by

$$K_I = \frac{K_C}{T_i}, \text{ and } K_D = K_C T_d \tag{8}$$

According to Ziegler and Nichols, the corresponding controller gains can be calculated from the relationships expressed by equations (9)-(12).

1. (P): $K_C = 0.5K_u$; $K_p = 0.5\ K_u$ (9)

2. (PI): $K_C = 0.45K_u$, and $T_i = \frac{T_u}{1.2}$; $K_p = 0.45\ K_u$, $K_i = \frac{1.2TK_p}{T_u}$ (10)

3. (PD): $K_C = 0.6K_u$, and $T_d = \frac{T_u}{8}$; $K_p = 0.6\ K_u$, $K_d = \frac{K_p T_u}{8T}$ (11)

4. (PID): $K_C = 0.6K_u$, $T_i = 0.5T_u$, $T_d = \frac{T_u}{8}$; $K_p = 0.6K_u$, $K_i = \frac{2TK_p}{T_u}$, $K_d = \frac{K_pT_u}{8T}$ (12)

Note that the right-hand, lower-case-subscript gain relationships for K_p, K_i, and K_d are for digital controllers, and that T is the control sampling time. In some cases, sustained oscillations are not allowed and the ultimate method cannot be used to tune the algorithm. The Ziegler-Nichols method is illustrated next by an example.

D. A Temperature Control System

A typical heating plant for a building is characterized by two time constants and a deadtime. For this example, assume time constants of 100 seconds and 10 seconds, and a deadtime of 40 seconds. A plant gain of 1 and a sampling time of 10 seconds are also assumed. It is required to design a digital PI controller using the Ziegler-Nichols method of tuning.

For the given parameter values, the plant transfer function can be written as

$$G_p(s) = \frac{e^{-40s}}{(10s+1)(100s+1)} = \frac{(0.001)e^{-40s}}{(s+0.1)(s+0.01)}$$

For simulation and/or analysis purposes, the pulse or digital transfer function of this plant is

$$G_hG_p(z) = Z[\frac{1-e^{-sT}}{s} G_p(s)] = \frac{(0.0355z+0.02465)z^{-3}}{z^2-1.2727z+0.332871}$$

The temperature control system was simulated using only proportional control action first. The proportional gain was increased until the response oscillated continuously. The ultimate gain observed is K_u=4.14 and the ultimate period is T_u=170 seconds. From Eq. (10), the PI controller gains are

$$K_C = 0.45K_u = 1.863, \text{ and } T_i = \frac{T_u}{1.2} = \frac{170}{1.2} = 141.67$$

For the conventional (analog) PI controller, then, the gains are

$$K_P = 1.863, \text{ and } K_I = \frac{K_C}{T_i} = \frac{1.863}{141.67} = 0.0132$$

Thus, the corresponding digital PI controller gains become

$$K_p = 1.863, \text{ and } K_i = TK_I = 10*0.0132 = 0.132$$

Simulation results showing the reference and feedback signals are shown plotted in Fig. 5. The results seem reasonable and, if desired, can be used as a starting point for fine-tuning the control algorithm.

III. Conclusions

A practical algorithm for controlling industrial machines and processes has been presented. The overall control systems can be simulated and implemented in real time using a microcomputer.

Using computer simulation, a designer can change the specifications of the interface devices, substitute the pulse generator by an ADC, or test the control algorithm when external disturbances enter the control loop. In the motor control case, the computer simulation results and the actual/experimental system performance characteristics are in close agreement.

The concepts presented/implied are general and can be applied to design other types of digital control systems.

Bibliography

[1] G. A. Perdikaris, Computer Controlled Systems: Theory and Applications, Kluwer Academic Publishers, 1991, reprinted in 1996.

[2] D. Y. Ohm, "A PDFF Controller for Tracking and Regulation in Motion Control," Proc of the PCIM Conference on Intelligent Motion, 1990.

[3] J. G. Ziegler and N. B. Nichols, "Optimum settings for Automatic Controllers," Trans ASME, Vol. 64, pp.759-768, Nov 1942.

GEORGE A. PERDIKARIS

George Perdikaris is currently a Professor of Engineering and Computer Science at the University of Wisconsin-Parkside. He received his B.S.E.E. from the University of Illinois in Champaign-Urbana and his M.S.E.E. and Ph.D. from the University of Missouri-Columbia. Dr. Perdikaris has been actively involved in joint research with industry in the area of digital/computer control of machines and processes. He teaches in Computer Engineering.

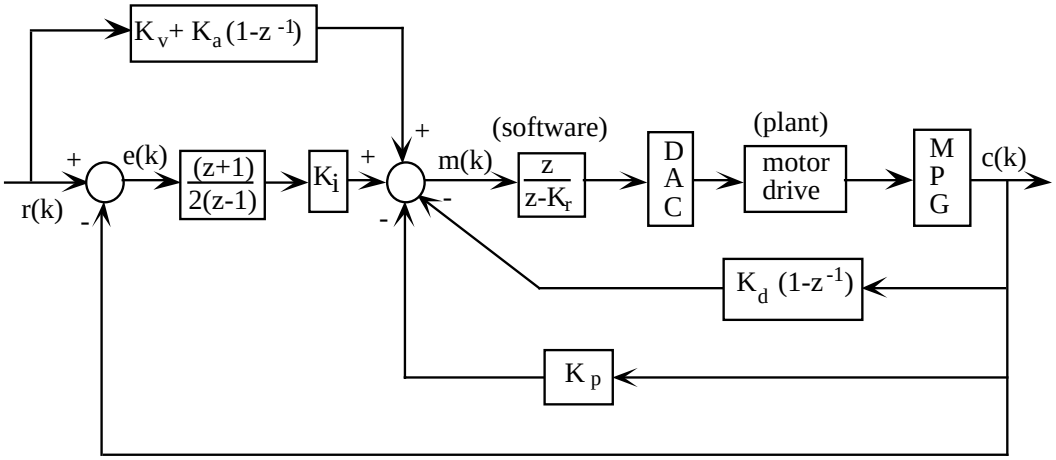

Fig. 1. Block diagram of the digital PDFF control system

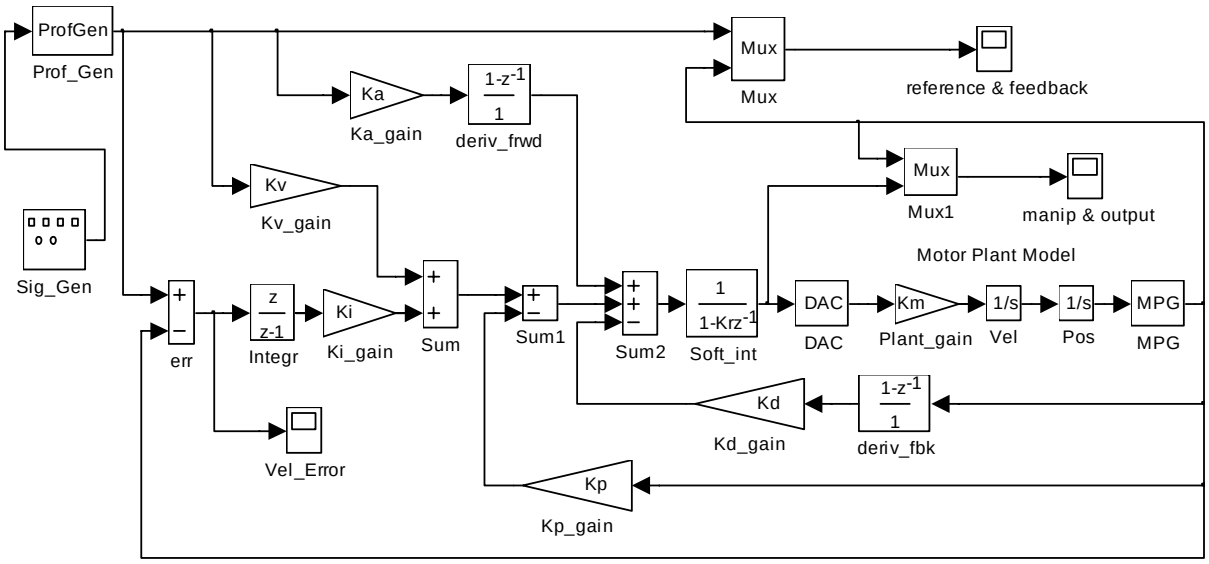

Fig. 2. Computer simulation diagram for the PDFF motor control system

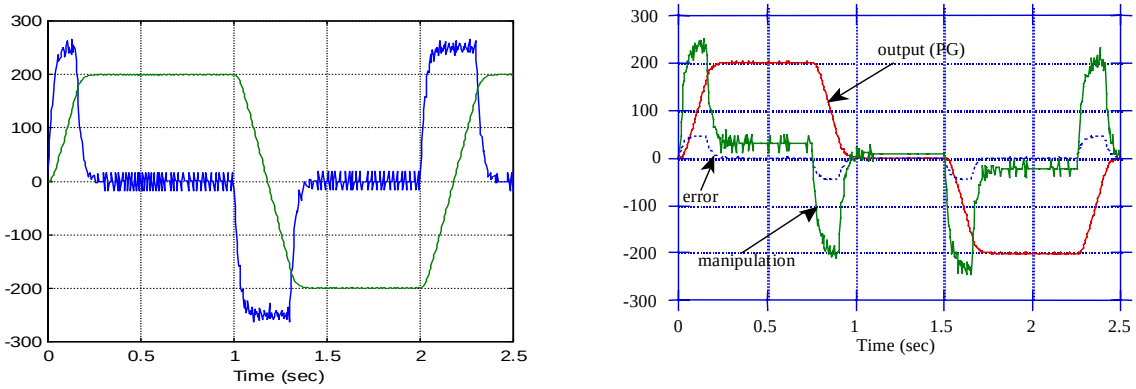

Fig. 3. Simulation (left) and experimental (right) results for motor velocity control system

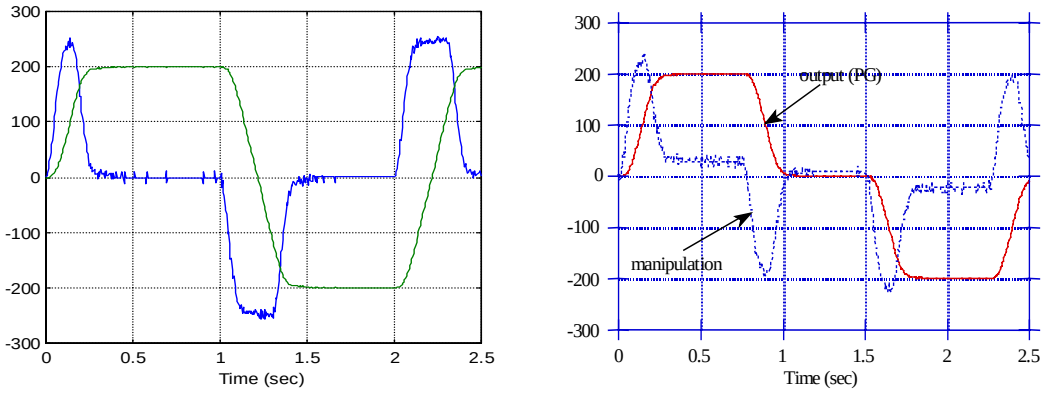

Fig. 4. Simulation (left) and experimental (right) results for motor position control system

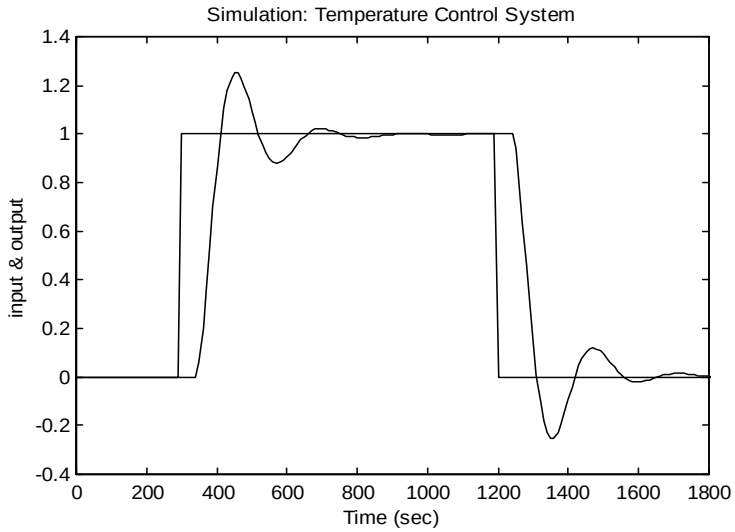

Fig. 5. Simulation results for temperature control system