A Novel Self-Tuning Control Design Approach for Industrial Applications

Richard J. Eucker and Zhiqiang Gao
Department of Electrical and Computer Engineering
Cleveland State University, Cleveland, Ohio 44145

Abstract: **A real time self-tuning control algorithm was developed to accommodate slow and sudden changes in the plant dynamics. Using a matrix interpolation technique, the controller coefficients are updated based on the changes in the frequency response of the plant so that a predetermined desired loop gain is maintained throughout the operation. The computer algorithms developed here can also be used as a computer aided design (CAD) tool for off-line control design. It replaces a tedious design process with a simple, easy to use, computer algorithm. The new algorithms were tested successfully in both software and hardware in the loop simulation.**

I. INTRODUCTION

Most control systems in industry today are fixed simple PID control loops. These control loops do not take into account the dynamic variations that occur in many systems. Therefore, over time, as the dynamics of the process inevitably varies, they fall out of tune, which causes the performance and stability to degrade.

The design techniques available to control engineers include PID (Proportional-Integral-Derivative), root-locus, state-space, and the frequency response based loop shaping technique, see [1,2,5] for more details. All these techniques require the engineer to know the actual transfer function of the plant. The identification of the transfer function of the plant can be quite a tedious procedure. Even when the transfer function of the plant is known, the controller design can still be a time consuming.

Among the available classical control techniques, the loop shaping technique [5] is arguably the most powerful one because it addresses all design issues such as command following, disturbance rejection, uncertainty in the process, noises, and stability margins, etc. The main idea is to determine a controller based on constraints of the loop gain transfer function, which are determined based on closed loop specifications. Unfortunately, the design process is an iterative one where one uses Bode and Nyquist plots to come up with an appropriate compensator to match the loop gain constraints. Such design is often tedious requiring skills and experiences; furthermore, once the controller is determined, it can not be easily tuned, like a PID controller, to accommodate changes in the process dynamics.

An algorithmic approach to loop shaping design was proposed in [3], which is based on the matrix interpolation method [7]. The objective for the current investigation is to address the implementation issues of the method in [3] and to develop a computer aided design (CAD) software package to allow an engineer to quickly and easily design compensators for general purpose control systems. To automatically determine the transfer function of the controller, the new design package only requires a set of dynamic I/O data taken from the plant and the closed loop system specifications. The transfer function model of the plant is not required, which makes the package much more practical and easy to use.

Note that similar software algorithms based on [7] have already been developed for single input and single output (SISO) as well as multi-input and multi-output system identification [8,9]. The identification and the loop shaping control design problems are similar in that both are trying to obtain a transfer function from a given frequency response.

Another objective of the current research is investigating the implementation of a real time self-tuning algorithm based on the interpolation method described above. Since the method only requires the I/O data and closed loop specifications, the design can be carried out in real time, where controller frequently adjusts itself to accommodate subtle changes in the process. Although this concept has been tested successfully in software simulation in a Web Tension Process [4], the actually implementation in hardware proves to be much more challenging [13].

II. A MATRIX INTERPOLATION APPROACH TO CONTROL DESIGN

To begin to understand the theory behind the self-tuning algorithm based on matrix interpolation, first consider the linear and time invariant SISO control system in Figure 2.1. In the block diagram, $G_p(s)$ represents the transfer function of the plant, $G_c(s)$ represents the transfer function of the compensator to be determined. The variables **r**, **u**, **y** represent the reference commend (setpoint), plant input, and plant output, respectively. The loop gain transfer function is defined as $L(s)=G_p(s)G_c(s)$. The design specifications are first translated into the loop gain frequency response constraints [5], $L(j\omega)=G_p(j\omega)G_c(j\omega)$, and then $G_c(s)$ is designed to meet these constraints.

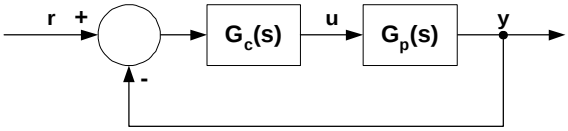

Figure 2.1 A Feedback Control System

Currently, $G_c(s)$ is determined iteratively so that the loop gain frequency response, $L(j\omega)$, satisfies all constraints. With all the constraints the designer must meet, the process of finding the appropriate $G_c(s)$ requires human intuition and experience. A great deal of trial and error is required and compromises have to be made. Such compromises include

the transient response specifications, the compensator complexity, the actuation limit, and the stability robustness, etc. It is shown in the following that this process can be automated by using the matrix interpolation theory, which is briefly illustrated below. More details on the interpolation theory can be found in [3,7].

A. Matrix Interpolation Problem Formulation

The design specifications for L(jω) are expressed as interpolation constraints of the form:

$$L(jw_i) = a_i, \qquad i = 1, l \tag{2.1}$$

The variables α_i are defined as complex numbers. The variable l represents the number of interpolation points. Most design specifications such as, crossover frequency, gain and phase margins, stability robustness against high frequency unmodeled dynamics, and disturbance rejection against low frequency output disturbance, can be interpreted as constraints on the magnitude and phase of L(jω) at a set of frequencies, $\{\omega_i\}$. With the desired L(jω) set, the designer must now solve the following equation to find a compensator $G_c(s)$ such that

$$L(jw_i) = G_p(jw_i)G_c(jw_i), \quad i = 1, l \tag{2.2}$$

where $G_p(j\omega_i)$ is the given frequency response of the plant evaluated at ω_i. Following (2.1) and let

$$G_p(jw_i) = b_i, \; i = 1, l \tag{2.3}$$

the numerator and denominator coefficients of $G_c(s)$ can be obtained by solving the following set of equations

$$G_c(jw_i)b_i = a_i, \; i = 1, l \tag{2.4}$$

Knowing the desired degree of $G_c(s)$, α_i, and β_i, solving $G_c(s)$ from (2.4) can be seen as the rational function interpolation problem. This problem is also known as the transfer function curve fitting problem [6].

B. Solving the Interpolation Problem

Control system constraints and properties can be expressed as interpolation constraints of transfer functions or transfer function matrices. Recent development in matrix interpolation theory [7] offers new theoretical framework in which various algebraic aspects of the matrix interpolation problems are explored. Computer algorithms [8,9] are developed to solve practical problems. The following section will show that the design problems formulated in equations (2.1) to (2.4) can be effectively solved using the matrix interpolation theory and algorithms.

First, let the compensator transfer function, $G_c(s)$, be defined as the ratio of a numerator and denominator polynomial, i.e., $G_c(s) = n(s)/d(s)$. The equation (2.4) is equivalent to solving the polynomial matrix [n(s), -d(s)], that satisfies

$$\left[n(jw_i), -d(jw_i)\right]\begin{bmatrix} b_i \\ a_i \end{bmatrix} = 0, \; i = 1, l \tag{2.5}$$

where l is the number of constraints.

Given d_1 and d_2 as column degrees of n(s) and d(s), and l constraints $\{j\omega_i, \alpha_i, \beta_i\}$, the matrix [n(s), -d(s)] can be uniquely determined from

$$\left[N, -D\right]\left[S_l, C\right] = \left[0, E\right] \tag{2.6}$$

where

$$\left[n(s), -d(s)\right] = \left[N, -D\right] S(s) \tag{2.7}$$

with

$$S(s) = \text{blk diag}\left\{\left[1, s, ..., s^{d_i}\right]^t\right\} \; i = 1,2 \tag{2.8}$$

$$S_l(s) := \left[S(jw_1)c_1, \ldots, S(jw_l)c_l\right] \tag{2.9}$$

The row vectors N and D from (2.6) and (2.7) contain the coefficients of n(s) and d(s), respectively, with $c_i = [\beta_i, \alpha_i]'$. In (2.6), the equation [N, D]C = E represents k additional constraints on the coefficients. The number of constraints, k, is defined as the number of columns of C or E. It is further defined as

$$k = \sum_{i=1}^{2} d_i + (p + m) - l \tag{2.10}$$

In addition, C is selected so that $[S_l, C]$ has full rank, assuming S_l has full rank [7]. Satisfying this equation guarantees a unique solution exists for any E. These additional conditions can be used, for example, to ensure the properness of $G_c(s)$, or to make d(s) a monic polynomial [7].

Note that (2.6) can be solved as a weighted least square problem [10]. Frequency weighting is easily implemented to reflect the degree of importance of each loop gain constraints. For example, the desired crossover frequency, the mechanical resonant frequencies, the frequencies of disturbance, etc. are more important than others. Therefore the designer can weight those frequencies higher in importance.

The solutions can be obtained in one step by solving (2.6), which is a set of linear algebraic equations. Solving (2.6) involves splitting $S_l(s)$ into two matrices, r_l and $S1_l$. The matrix r_l is the last row of the matrix $S_l(s)$ and the matrix $S1_l$ is the remaining rows in the $S_l(s)$ matrix. The compensator parameter vector, [N –D], is determined from:

$$\left[N\ -D\right] = \left[\text{real}(r_l)\ \text{imag}(r_l)\right] \times \left[\text{real}(S1_l)\ \text{imag}(S1_l)\right]^{-1} \tag{2.11}$$

Note that, assuming that the plant is stable and the controller determined above is also stable, the closed-loop system is guaranteed to be stable if the desired the loop gain is chosen to not encircle (-1,0) point on the complex plane.

III. SELF-TUNING CONTROL AND REAL TIME IMPLEMENTATION

The significance of the solution to loop shaping problem, described in (2.1) to (2.11) above is two-fold:

1) It provides a computational tool that will help the control engineer to quickly find a solution; it avoids the tedious iterative design process associated with the design;
2) It enables the control design and tuning based on loop shaping approach to be implemented in real time.

Note that in this paper we assume that:

1) The design specifications are already translated to loop gain constraints in the form of (2.1);
2) There is enough excitation in the time domain input and output data so that the approximate frequency response of the plant can be obtained.

Under these assumptions, the design problem becomes one of obtaining the plant frequency response in (2.4) and solving the linear algebraic equation in (2.11), which can be done automatically in software. The software platform chosen in this research is Matlab/Simulink.

A. Software Implementation

A set of computer programs has been written, to implement the design process described in (2.1) to (2.11). After the user inputs the plant I/O data and the desired loop gain frequency response, the computer program will automatically compute the controller parameters. This set of programs forms a basis of a control design CAD package. The graphical user interface is to be in the next phase of research. In this paper, we concentrate on the problem of implementing this algorithm for real time self-tuning applications. The critical issue is to obtain reliable plant frequency response.

The design strategy is that, once the frequency response is established, it is compared to the previous frequency response data. If the variation exceeds a predetermined limit, the controller redesign is automatically carried out; otherwise, no action will be taken. The system configuration is shown in Figure 3.1.

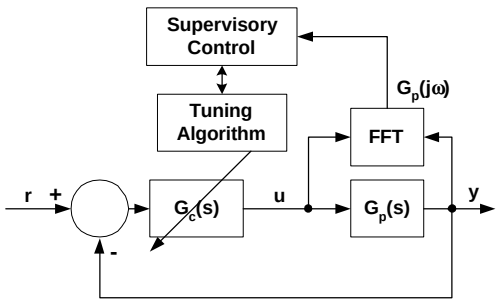

Figure 3.1 Diagram of a Self-Tuning Control System

Implementing self-tuning requires the input and output (I/O) data to be continuously monitored. The frequency response of the plant, $G_p(j\omega_i)$, is obtained from the I/O data by performing spectral analysis. Using the new $G_p(j\omega_i)$ and the constraints of the desired loop gain, $L(j\omega_i)$, a new compensator is computed using equations (2.1) to (2.12). A flow chart of this process is shown in Figure 3.2.

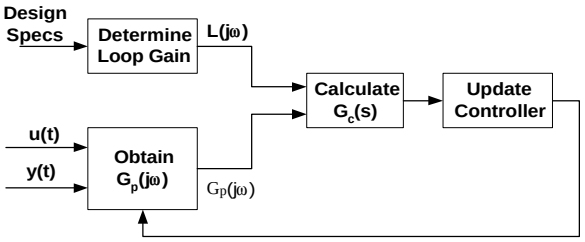

Figure 3.2 Flow Chart of the Self-Tuning Process

B. Determining the Frequency Response of the Plant

The most critical component in developing a self-tuning control system is accurately computing the frequency response of the plant. The data conversion from time domain to frequency domain must occur in real time to allow for a self-tuning system to work automatically. The calculation of the plant frequency response must also be accurate and not too sensitive to system noise and computational inaccuracy. Lastly, the frequency response calculation must allow for frequency weighting.

The frequency response of the plant, theoretically, can be obtained by taking the ratio of the Fourier Transform of the input and the output data. But such approach is known to suffer poor numerical properties and is sensitive to noises. A more reliable method to determine the plant frequency response from the input and output data is to use a ratio of the cross spectra and auto spectra of the input and output time history, G_N [11], as:

$$\hat{G}_N\left(e^{jw_0}\right) = \frac{\hat{\Phi}^N_{yu}(w_0)}{\hat{\Phi}^N_u(w_0)} \tag{3.1}$$

where $\hat{\Phi}^N_{yu}(w_0)$ is the cross spectra of the input and output signal and $\hat{\Phi}^N_u(w_0)$ is the auto spectra of the input signal.

C. Software Simulation

A Simulink model was developed to implement the self-tuning algorithm in real time. The model consists of a matrix interpolation block and a compensator block. The model is shown in Figure 3.3.

Matlab Simulink S-Functions, written in both 'C' and Matlab's command language, were utilized in developing a working model of a self-tuning control system.

The simulation started at t = 0 with the initial plant set to $G_p(s) = 5/(s+5)(s+1)$. The simulation was run long enough to verify that initial $G_c(s)$ allowed the output to track the square wave setpoint change according to the desired loop gain. At t = 350 seconds, the plant transfer function was changed to $G_p(s) = 5/(s+15)(s+1)$. The effect of this makes the output

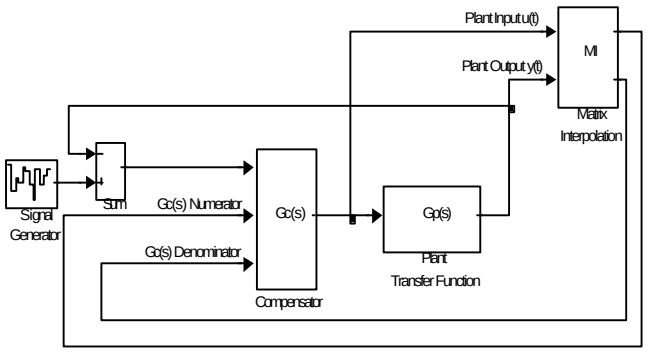

Figure 3.3 SIMULINK Diagram of Self-Tuning Model

sluggish in tracking the setpoint. As shown in Figure 3.4, by t = 550 seconds, $G_c(s)$ had been automatically tuned to allow the output to track the reference signal as it did before the plant was changed.

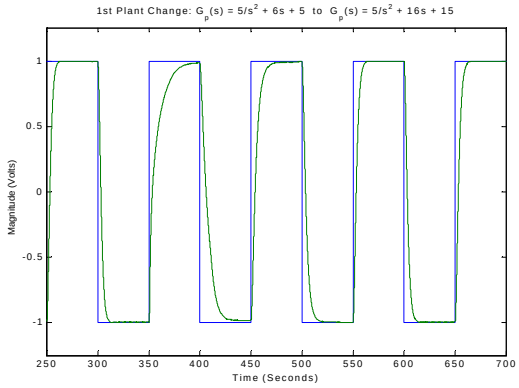

Figure 3.4 Plot After First Plant Change in Simulation

Finally, at t = 750 seconds, the plant was changed back to the original transfer function, $G_p(s) = 5/(s+5)(s+1)$. The output began to overshoot the reference signal, but by t = 950 seconds, the output was tracking the reference signal as before as shown in Figure 3.5.

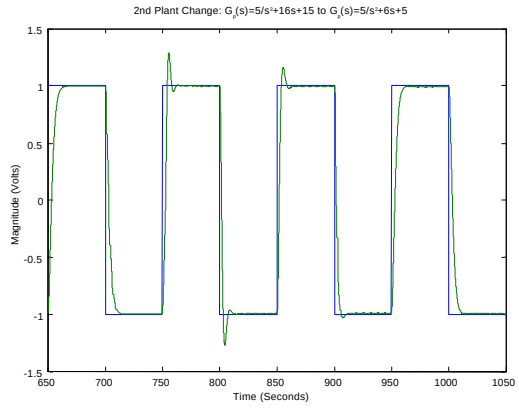

Figure 3.5 Plot After Second Plant Change in Simulation

The simulation demonstrated that the self-tuning algorithm is able to continuously monitor the frequency response of the plant and adjust the controller parameter to offset the dynamics variations of the plant.

IV. HARDWARE IN THE LOOP SIMULATION

Once the algorithms have been developed and tested in software, the next step is to bridge the gap between software simulation and real world applications. Here, the method of the so-called "hardware in the loop" simulation is used. It has the flexibility of software simulation since the controller is implemented in a PC. Instead of using a mathematical model, the PC controls a physical plant in the real world via a data acquisition board. This way, the engineer can fine-tune the process in the lab before working on actual production equipment.

In the hardware in the loop simulation, the self-tuning controller is implemented on a PC running Simulink with a Keithley Metrabyte Analog I/O board as a communication link to the plant. The plant is implemented on a separate DSP board made by Integrated Motions, Inc. (IMI) [12], see Figure 4.1. The plant is also programmable using Simulink. The switch in Figure 4.1 can be switched from the PC side via a separate channel to simulate a sudden dynamic change.

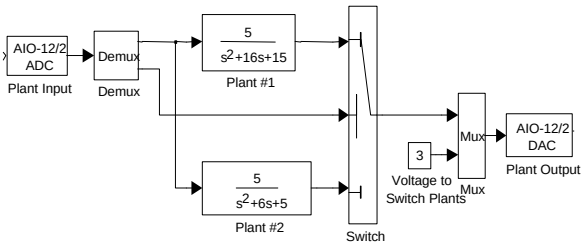

Figure 4.1 Hardware in the Loop Plant Setup

The simulation model shown in Figure 3.3 was modified by replacing the simulated plant transfer function with the Keithley Metrabyte device drivers as shown in Figure 4.2.

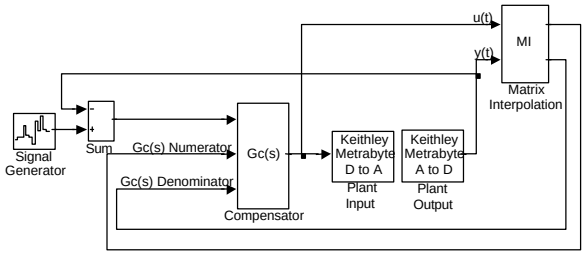

Figure 4.2 SIMULINK Diagram of Self-Tuning Model

The hardware simulation started at t=0 with the switch pointed to plant #1, $G_p(s) = 5/(s+15)(s+1)$. By t = 150

seconds, the compensator is well tuned and the output tracks the square-wave reference signal quite nicely. At t = 250 seconds, the switch was thrown to enable plant #2, $G_p(s) = 5/(s+5)(s+1)$. As shown in Figure 4.3, at t = 450 seconds, the plant was tracking the reference signal as well as before as a result of the continuous tuning of the controller.

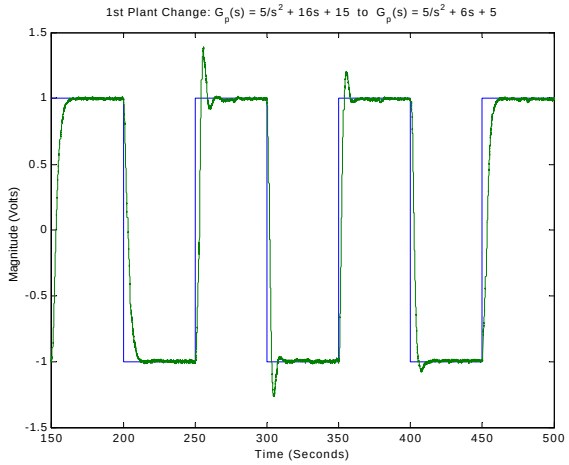

Figure 4.3 Plot After First Plant Change in Hardware Simulation

Finally, at t = 550 seconds, the switch was thrown back to plant #1, $G_p(s) = 5/(s+15)(s+1)$. As can be seen in Figure 4.4, by t = 750 seconds, the performance is again recovered.

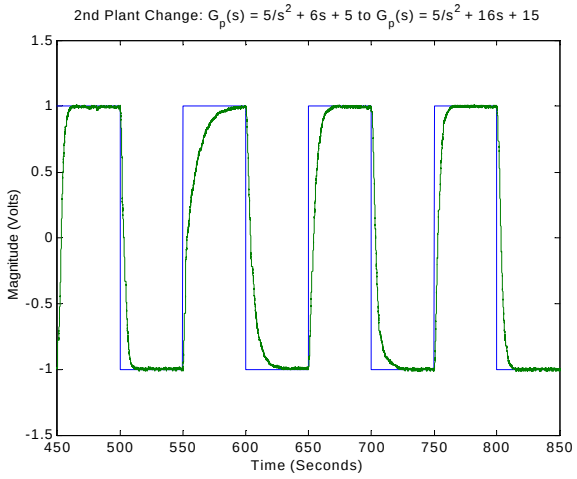

Figure 4.4 Plot After Second Plant Change in Hardware Simulation

V. CONCLUDING REMARKS

A self-tuning control system based on matrix interpolation techniques was developed and verified in hardware simulation. A CAD package has been developed to allow the engineer to design an accurate self-tuning control system based on the constraints of loop gain frequency response. Interested readers are referred to [13] for more technical details.

REFERENCES

[1] J.M. Maciejowski, Multivariable Feedback Design, Addison Wesley, New York, 1989.

[2] J.C. Doyle, B.A. Francis, and A.R. Tannenbaum, Feedback Control Theory, Macmillan, New York, 1992.

[3] Z.Gao, "An Algorithmic Approach to Loop Shaping Design" Journal of Franklin Institute, Vol. 332B, No.6, pp. 643-656, 1995.

[4] Brian Boulter, "Matrix Interpolation Based Self-Tuning Web Tension Regulation", M.S. Thesis, Dept. of Electrical Engineering, Cleveland State University, Oct. 1995

[5] Charles E. Rohrs, James L. Melsa and Donald G. Schultz, Linear Control Systems, McGraw-Hill, Inc., New York, 1993.

[6] E.C. Levy, "Complex Curve Fitting", IEEE Transactions on Automatic Control, AC-4, 1959.

[7] P.J. Antsaklis and Z. Gao, "Polynomial and Rational Matrix Interpolation: Theory and Control Applications", Journal of Control International, Vol.58, No.2, pp.349-404, July 1993.

[8] R. V. Savescu, "System Identification from Frequency Response Using Matrix Interpolation Theory," M.S. Thesis, Dept. of Electrical Engineering, Cleveland State University, Oct. 1992.

[9] B. Tabachnik, "A Computer Algorithm for Multi-input Multi-output System Identification Using Chebychev Polynomials", M.S. Thesis, Dept. of Electrical Engineering, Cleveland State University, May 1994.

[10] C.L. Lawson and R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, Inc, Englewood Cliffs, 1974.

[11] Lennart Ljung, System Identification: Theory for the User, Prentice Hall PTR, New Jersey, 1987

[12] IMI MX31User's Manual, Integrated Motion Inc., Natick, MA, 1996

[13] Richard J. Eucker, "Self-Tuning Control Systems Using Matrix Interpolation", M.S. Thesis, Dept. of Electrical Engineering, Cleveland State University, Dec. 1998