THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

OPTIMIZATION OF THE RISE TIME FOR A SECOND ORDER PITCH RATE CONTROL SYSTEM

Group 2

Advisors: Benjamin Davis, Hamid Ahmad

CHE 488/ Fall 2011

I. Abstract

This paper describes the tuning of an optimal proportional-integral (PI) controller to minimize rise time for an airplane pitch rate control system. The pitch rate control system, along with the yaw and roll control systems, make up the backbone of an airplane flight control system. We present the transfer function of the system and impose stability constraints using Routh-Hurwitz tabulation. We must also impose constraints so that the system may be approximated as a standard form second order system, so that a closed-form rise time expression from Ogata [1] can be used. We rigorously prove that the objective function has a minimum over the feasible set. We design the optimal controller is designed using Matlab; the solution could be implemented as part of any airplane flight control system where the pitch dynamics can be modeled as first order.

II. Introduction

In aircraft flight systems, the primary concern is control over the principal axes: yaw, pitch, and roll to ensure a stable flight path. These axes are shown in Figure 1. Our focus in this paper is the pitch rate control system. A control system is a system which attempts to maintain a quantity (or quantities) in a specified operating range. In a pitch rate control system, the pilot sets a desired pitch rate, and the system responds by adjusting the actual aircraft pitch rate.

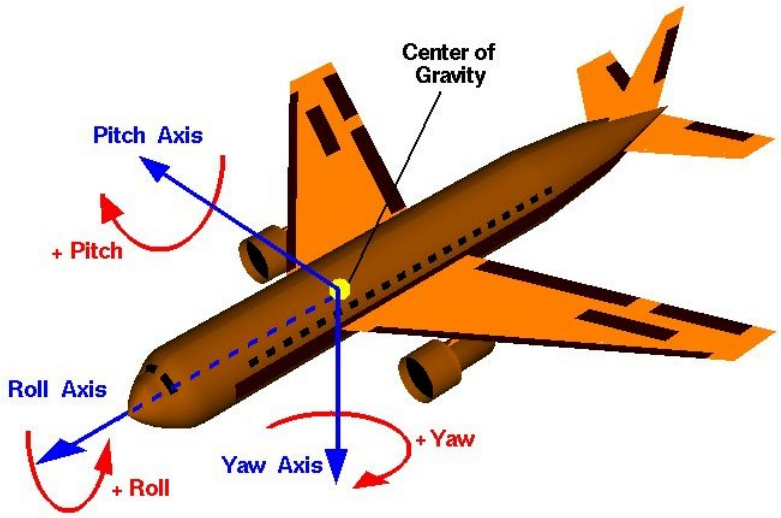

Fig. 1: Diagram which illustrates principal axes of aircraft motion, from [2]

As this problem is a standard one present in nearly every aircraft flight control system, a vast number of controlling methods have been explored, for example [3], [4]. Pitch rate controllers have been implemented by using fuzzy controls [5], neural networks [6], and sliding mode control methods [7]. This project uses a PI controller, as implemented in a pitch rate control system by [8].

PI controllers are used in a wide range of applications, but they are inherently linear and non-adaptive. As shown in [9], PI controller design based purely on system models gives generally "poor performance," unless they are retuned with flight test data. A more advanced and suitable PI controller is adaptive in nature [10]. Therefore, for proper application of our optimal solution using a linear non-adaptive PI controller, the controller will need to be adjusted and verified with flight test data. Simulations using a "crewed flight simulator" would also assist in adjusting the controller and verifying functionality [11].

As our system is linear and time invariant (LTI), LTI theory can be applied to develop a system model. Any LTI system can be completely characterized by its impulse response function. The impulse response $g(t)$ is given by $g(t) = c(t)|_{r(t)=\delta(t)}$, where $c(t)$ is the output of the system, $r(t)$ is the input to the system, and $\delta(t)$ denotes the dirac delta function. The impulse response can be used to determine the system response for *any* input $r(t)$:

$$c(t) = g(t) \star r(t) \tag{1}$$

where $\star$ denotes convolution.

Because convolution computations can be tedious, we apply the Laplace transform to Equation (1) to transform from the time domain to the Laplace domain. The convolution in the time domain maps to multiplication in the Laplace domain [1]:

$$\begin{aligned} \mathscr{L}[c(t) &= g(t) \star r(t)] \\ C(s) &= G(s)R(s) \end{aligned} \tag{2}$$

$G(s)$ in Equation (2) is defined as the transfer function of the system. Now, given any input in the time domain, we apply the Laplace transform and multiply it with our transfer function to find the response in the Laplace domain. By taking the inverse Laplace transform of this quantity, the response in the time domain can be obtained. This process is equivalent to convolving the input in the time domain with the impulse response of the system.

Our system with the PI controller is shown in Figure 2. $G(s)$ is the transfer function describing the pitch rate dynamics. Our desired pitch rate is shown as $R(s)$, while the actual pitch rate is shown as $C(s)$.

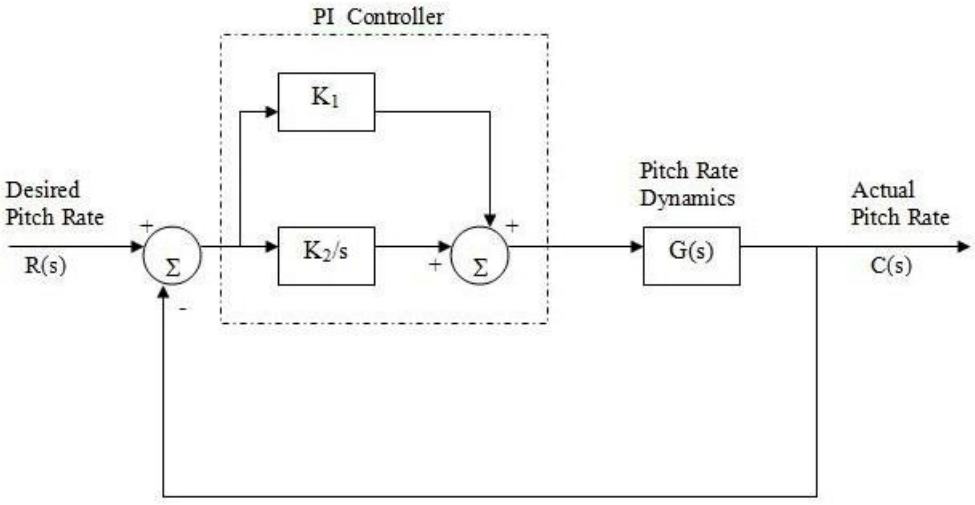

Fig. 2: Block diagram representing the pitch rate control system.

In this pilot-based model, the input can be modeled as a unit step function [12]. The unit step function is a function which is zero for $t < 0$ and is one for $t > 0$. Our pitch rate system dynamics $G(s)$ are drawn from Rynaski [13]. The dynamics apply to both the phugoid flight mode, characterized by low frequency oscillations, and to the short-term flight mode, characterized by high frequency oscillations. Rynaski shows that the effects of the phugoid oscillations are decoupled from the pitch rate expression. This is beneficial for us, as we can extend the operating range of our controller to both of these flight modes.

With these dynamics and this system setup, our goal is to minimize rise time. This performance criterion has been defined as the time it takes for the response $c(t)$ to go from 10% to 90% of the final value $r(t)$ [1]. This rise time t_r is illustrated in Figure 3.

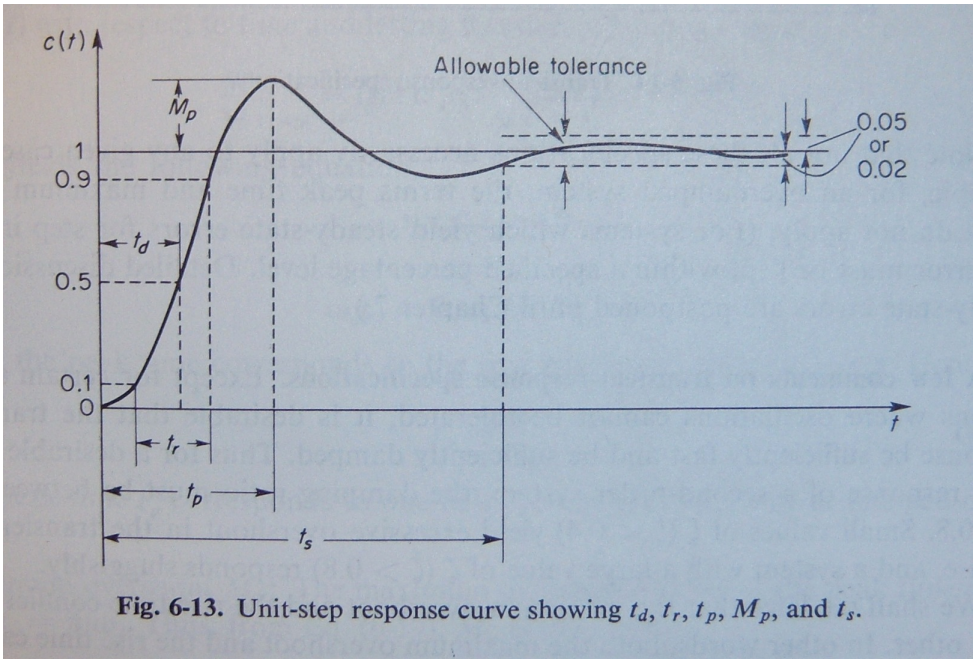

Fig. 6-13. Unit-step response curve showing t_d, t_r, t_p, M_p, and t_s.

Fig. 3: Diagram illustrating rise time t_r [1].

III. Problem Description

A zero, z, of a rational function $f : \mathbb{C} \to \mathbb{C}$ is a value such that $f(z) = 0$. If the Laurent series of the function $f(s)$ around a singular point p has a principal part that contains at least one nonzero term, then we call p a pole of $f(s)$ [14].

In general, we can express the transfer function of any second order system that contains no finite zeros as follows:

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3}$$

where ω_n is the undamped natural frequency and ζ is the damping ratio of the system [15]. These parameters determine the transient response characteristics, including rise time, of a second order system. As described in [1], the rise time of this system due to a unit step input has an analytical expression given by:

$$t_r = \frac{\pi - \cos^{-1}(\zeta)}{\omega_n\sqrt{1-\zeta^2}} \tag{4}$$

For this project, the transfer function $G(s)$ we used to model the pitch-rate command system is given by:

$$G(s) = \frac{20}{s + 4.4}, \tag{5}$$

the model used in the NASA aircraft report [13]. When $G(s)$ is placed in a unity feedback loop with a PI controller, as shown in Figure 2, the closed loop transfer function of the complete pitch-rate control system is given by $M(s)$. For our system, $M(s)$ is given by the following equation:

$$M(s) = \frac{(K_1 + K_2/s)G(s)}{1 + (K_1 + K_2/s)G(s)} \tag{6}$$

Substituting Equation (5) into Equation (6), $M(s)$ reduces to the following expression:

$$M(s) = \frac{s(20K_1) + 20K_2}{s^2 + s(20K_1 + 4.4) + 20K_2} \tag{7}$$

We see from Equation (7) that our pitch-rate control system is a secondorder system containing two poles and one zero. The transfer function cannot be expressed in the form given by Equation (3) because it contains a finite zero. Therefore, we cannot calculate the rise time of the system due to a unit step input directly from Equation (4). However, [16] outlines a procedure for suppressing the effects of the finite zero on the system's transient behavior, allowing us to approximate our system as a second order system with no finite zero. With this approximation, we can calculate the rise time of the system from Equation (4).

The first step of this approximation procedure is to rewrite the transfer function in the form:

$$M(s) = \frac{(s + \alpha)(\omega_n^2/\alpha)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{8}$$

where $s = -\alpha$ is the zero of the system. We therefore rewrite Equation (7) as follows:

$$M(s) = \frac{(s + K_2/K_1)20K_1}{s^2 + s(20K_1 + 4.4) + 20K_2} \tag{9}$$

where $-\alpha$= $-K_2/K_1$, $2\zeta\omega_n = (20K_1 + 4.4)$, and $\omega_n^2 = 20K_2$. Solving the latter two equations for ζ and ω_n in terms of K_1 and K_2 we have:

$$\zeta = \frac{20K_1 + 4.4}{2\sqrt{20K_2}} \tag{10}$$

$$\omega_n = \sqrt{20K_2} \tag{11}$$

In order for the finite zero to be considered insignificant in regards to the transient behavior, the magnitude of the zero must be more than 5 times the magnitude of the real part of the complex conjugate poles [16]:

$$\left|\frac{-K_2}{K_1}\right| \geq (5 + \varepsilon)\left|\Re(P_{complex})\right| \tag{12}$$

$P_{complex}$ is either of the two complex poles of the system, as each have the same real part. Note that we choose the quantity $\varepsilon > 0$ to equal 10^{-10} in order to have a non-strict inequality constraint.

The poles of the system given by Equation (9) are the roots of the denominator $D(s)$:

$$D(s) = s^2 + s(20K_1 + 4.4) + 20K_2. \tag{13}$$

From the quadratic formula, we see that $\frac{-b}{2a}$ corresponds to the real part of the roots when the roots are a complex conjugate pair. Therefore we can rewrite Equation (12):

$$\left|\frac{-K_2}{K_1}\right| \geq (5 + \varepsilon)\left|-10K_1 - 2.2\right| \tag{14}$$

As it is not conventional to use the absolute value function when expressing constraints, we rewrite the constraint into the following equivalent set:

$$\frac{K_2}{K_1} \leq -(5+\varepsilon)(10K_1+2.2) \tag{15}$$

$$\frac{K_2}{K_1} \geq (5+\varepsilon)(10K_1+2.2) \tag{16}$$

When determining the optimal values of K_1 and K_2, we must ensure that our system is stable for our choices of K_1 and K_2. A system is stable when all the real parts of its poles are less than zero [15]. We use the modified Routh-Hurwitz (R-H) stability criterion [1] to determine which constraints on K_1 and K_2 guarantee stability. Rather than allowing poles to exist near the imaginary axis, the modified method ensures that the real parts of the poles are less than or equal to -0.5, giving us a larger stability margin [17].

The denominator polynomial given by Equation (13) is used in the stability criterion with the substitution $s = z - 0.5$. After substitution, we obtain:

$$D(z) = z^2 + z(20K_1 + 3.4) - 10K_1 + 20K_2 - 1.95 \tag{17}$$

The Routh-Hurwitz array of this polynomial is shown below:

	(1)	(2)	(3)
z^2	1	$-10K_1 + 20K_2 - 1.95$	0
z^1	$20K_1 + 3.4$	0	0
z^0	$-10K_1 + 20K_2 - 1.95$	0	0

In order to satisfy the R-H criterion, all the entries in column (1) must be greater than or equal to zero. This gives us the following constraints on K_1 and K_2:

$$20K_1 + 3.4 \geq 0 \tag{18}$$

$$-10K_1 + 20K_2 - 1.95 \geq 0 \tag{19}$$

Our final constraint on K_1 and K_2 is a transient response constraint. Figure 3 shows the transient response of a second order system due to a unit step input. The rise time and maximum peak overshoot (M_p) are depicted in the figure. Selection of the damping ratio ζ involves a trade-off between rise time and M_p. A smaller damping ratio decreases the rise time but increases M_p, while increasing the ratio increases the rise time and decreases M_p. A suggested damping ratio range is given by [16]:

$$0.4 \leq \zeta \leq 0.7 \tag{20}$$

Substituting Equation (10) into Equation (20), we have:

$$0.4 \leq \frac{20K_1 + 4.4}{2\sqrt{20K_2}} \leq 0.7 \tag{21}$$

As can be seen in Equation (21), to avoid division by zero and imaginary quantities, we must impose an additional constraint on K_2:

$$K_2 \geq \varepsilon \tag{22}$$

Our objective function for the optimization problem is given by Equation (4). The constraints are given by equations (15), (16), (18), (19), and (21).

$$\underset{K_1,K_2\in\mathbb{R}}{\text{minimize}} \quad \frac{\pi - \cos^{-1}\left(\frac{20K_1+4.4}{2\sqrt{20K_2}}\right)}{\sqrt{4(5K_2 - 25K_1^2 - 11K_1 - 1.21)}} \tag{23a}$$

$$\text{subject to:} \quad \frac{K_2}{K_1} \leq -(5+\varepsilon)(10K_1 + 2.2) \tag{23b}$$

$$\frac{K_2}{K_1} \geq (5+\varepsilon)(10K_1 + 2.2) \tag{23c}$$

$$20K_1 + 3.4 \geq 0 \tag{23d}$$

$$-10K_1 + 20K_2 - 1.95 \geq 0 \tag{23e}$$

$$0.4 \leq \frac{20K_1 + 4.4}{2\sqrt{20K_2}} \leq 0.7 \tag{23f}$$

$$K_2 \geq \varepsilon \tag{23g}$$

This is a nonlinear problem (NLP) because the objective function is nonlinear, and there are nonlinear constraints. Figure 4 shows a plot of all the K_1 and K_2 values which satisfy the constraint set. The precision in the computational evaluation is 0.01, which accounts for the spaces between the feasible points. The lines seen are formed of many feasible points. The set is not convex because there exists a line segment connecting two points within the set, such that the line segment does not lie completely within the set. The feasible set is not convex, therefore this problem is not convex.

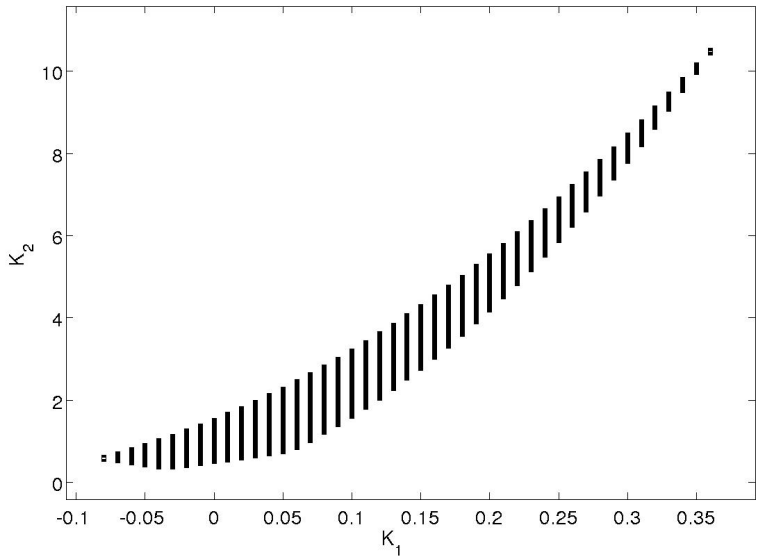

Fig. 4: Feasible set, determined by constraints.

IV. Existence of Minimum Proof

We can prove that our problem is feasible and that it has an optimal solution by using the Weierstrass theorem [18]. This theorem guarantees that a function that is continuous over a set achieves a minimum over the set if the set is closed and bounded. Therefore, we will begin by proving that the constraints on the decision variables K_1 and K_2 form a closed and bounded set of feasible points.

We see from Figure 4 that both K_1 and K_2 are bounded below and above. Because the constraints (23b) - (23g) are non-strict inequalities, the set of feasible points is closed.

To show that our objective function (23a) is continuous, we first prove that the expressions in the numerator and denominator of the objective function are continuous functions over the feasible set.

For the denominator of the function to be continuous, the terms inside the square root must be greater than or equal to zero to avoid producing imaginary quantities. However, in order for the objective function to be continuous, there must not be any division by zero. Therefore, the terms inside the square root must be strictly greater than zero:

$$4(5K_2 - 25K_1^2 - 11K_1 - 1.21) > 0 \tag{24}$$

Solving for K_2

$$K_2 > 5K_1^2 + 2.2K_1 + 0.242 \tag{25}$$

Figure 5 shows the boundary given by Equation (25), along with the feasible set shown previously. Note that all the feasible points of our optimization problem satisfy the requirement given by Equation (25). Therefore, the expression in the denominator is continuous for all feasible K_1 and K_2, and any discontinuity in the objective function due to division by zero is also avoided.

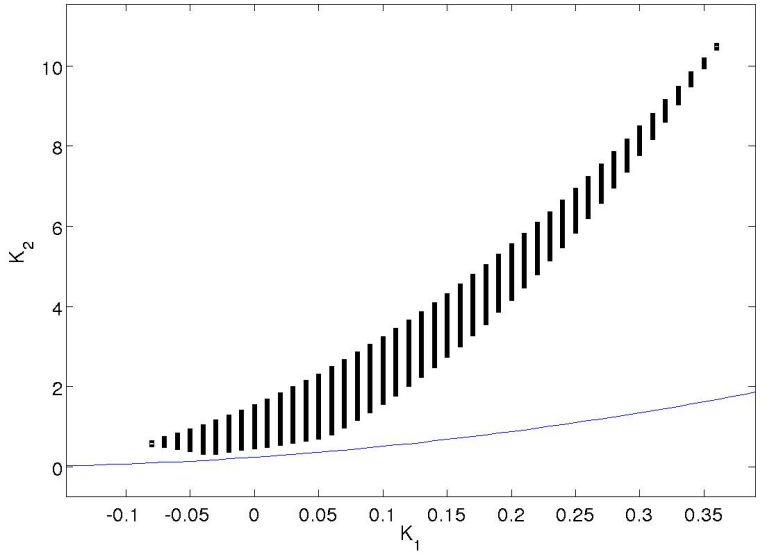

Fig. 5: Feasible set with continuity constraint as per Equation (25).

For the numerator to be continuous, the argument of $\cos^{-1}(\zeta)$ must be of magnitude less than or equal to 1. We show here that all feasible points fulfill this requirement. Recall Equation (10):

$$\zeta = \frac{20K_1 + 4.4}{2\sqrt{20K_2}}$$

As the magnitude of ζ must satisfy our continuity requirement:

$$\left\| \frac{20K_1 + 4.4}{2\sqrt{20K_2}} \right\| \leq 1 \tag{26}$$

Because K_2 is greater than zero:

$$\left| \frac{20K_1 + 4.4}{2\sqrt{20K_2}} \right| \leq 1 \tag{27}$$

By further simplifying, we see:

$$\frac{|10K_1 + 2.2|}{\sqrt{20K_2}} \leq 1 \tag{28}$$

From figure 4, K_1 is always greater than -0.22, thus the numerator is always positive. Rewriting with this realization:

$$\frac{10K_1 + 2.2}{\sqrt{20K_2}} \leq 1 \tag{29}$$

Solving for K_2, we find the requirement for continuity is the same as earlier, with equality replacing the strict inequality:

$$K_2 \geq 5K_1^2 + 2.2K_1 + 0.242 \tag{30}$$

Thus as in Figure 5, all feasible points satisfy this requirement.

The numerator and denominator are continuous functions over the feasible set, and because the ratio of continuous functions is continuous, our objective function is continuous.

Because our objective function is continuous over the feasible set, and the feasible set is non-empty, closed and bounded, then by the Weierstrass theorem a minimum exists.

V. Results

A. Solution Method

There are a number of solvers available that can solve this type of optimization problem. The LOQO software [19], developed by Robert Vanderbei and Hande Y. Benson uses an interior-point algorithm to solve this type of problem. The Matlab function *fmincon* is able to solve this problem using either the active-set algorithm or the interior-point algorithm. We solve the optimization problem using Matlab's optimization toolbox, specifically, the "fmincon" solver [20] with the "active-set" algorithm. There are two variables, with 5 nonlinear constraints in the problem; the total code is around 50 lines, including comments. The solution time is about 0.4s.

B. Solution Analysis

The optimal gains K_1 and K_2 are found to be:

$$K_1 = 0.3667$$

$$K_2 = 10.7556$$

with the transfer function for the optimal system, $M(s)_{opt}$, given by:

$$M(s)_{opt} = \frac{s(7.3340) + 215.1120}{s^2 + s(11.7340) + 215.1120}. \tag{31}$$

The optimal rise time t_r corresponding to this resulting system is:

$$t_r = 0.1475\text{s}$$

Matlab's "stepinfo" function [21] gives a rise time of $t_r = 0.0835\text{s}$ for the optimal system, a difference of $t_\Delta = 0.064\text{s}$, where our approximation overestimates rise time as compared to Matlab's numerically approximated rise time. This difference in rise time (a factor of 1.8) is primarily caused by our approximation using the dominant pole method. In this situation, when using rise time as a performance criterion, the dominant pole method creates the appearance of degraded system performance, when in fact the actual value is more agreeable.

The step response of $M(s)_{opt}$ system is shown in Figure 6. The damping ratio of this system is found to be $\zeta = 0.4$, a value which equals the Matlab numerical approximation. By examining the step response, we that the percent overshoot is within an acceptable operating range.

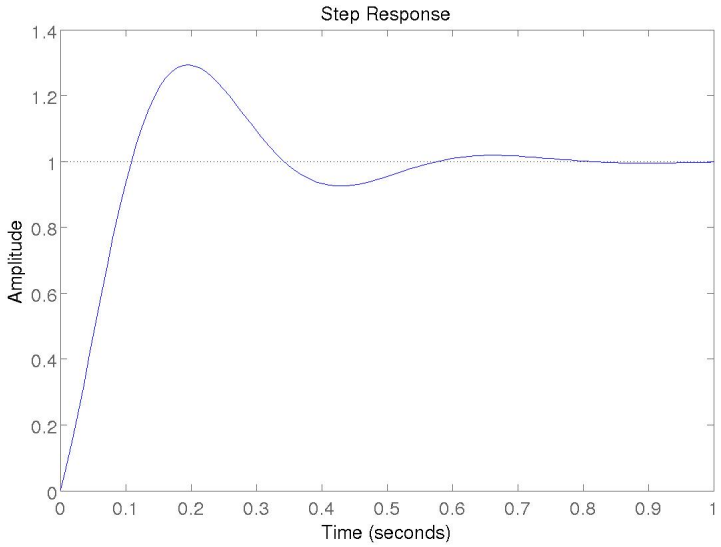

Fig. 6: Step response for optimal system.

C. Sensitivity Analysis

We examine the effects of various changes in the problem parameters on the solution:

1) Stability Margin: To begin, we change the stability margin from 0.5 to 0.1, thereby allowing for poles to move closer to the imaginary axis. This changes constraints given by Equation (23d) and Equation (23e). We rewrite the constraints to be:

$$20K_1 + 4.2 \geq 0$$

$$-2K_1 + 20K_2 - 0.43 \geq 0$$

We find:

$$K_1 = 0.3667$$

$$K_2 = 10.7556$$

With a corresponding rise time of:

$$t_r = 0.1475\text{s}$$

This is the same result as the original problem, and hence the reduced stability margin has no effect on the optimal value for the system. Figure 6 still represents the output of the optimal system with the changed stability margin.

2) Damping Ratio: We tighten the constraint on the damping ratio given by Equation (23f) to be:

$$0.5 \leq \zeta \leq 0.6$$

We find:

$$K_1 = 0.1467$$

$$K_2 = 2.6889$$

With a corresponding rise time of:

$$t_r = 0.3298\text{s}$$

and a damping ratio of:

$$\zeta = 0.5$$

As our original optimal solution corresponded to a ζ value of 0.4, this is no longer possible within our new constraints. As mentioned earlier, ζ determines the maximum percent overshoot. By increasing the lower bound on ζ, we increase our restriction on percent overshoot, therefore, we also increase the lower bound on minimum rise time. Figure 7 shows this.

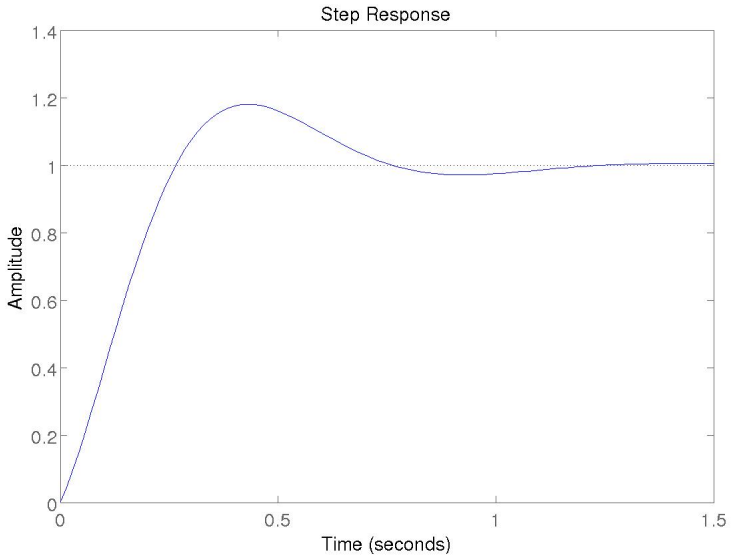

Fig. 7: Step response for system tightened ζ bounds.

By relaxing the constraint on the damping ratio given by Equation (23f) to be:

$$0.3 \leq \zeta \leq 0.8$$

We find:

$$K_1 = 15$$

$$K_2 = 1.2869 \times 10^4$$

With a corresponding rise time of:

$$t_r = 0.0039\text{s}$$

and a damping ratio of:

$$\zeta = 0.3$$

As expected, the rise time decreased, as we allowed for ζ to approach a lower value. The K_1 and K_2 values are found to be outside the original feasible set. This is acceptable, as the feasible set must be reevaluated with the new constraints. The step response can be seen in Figure 8.

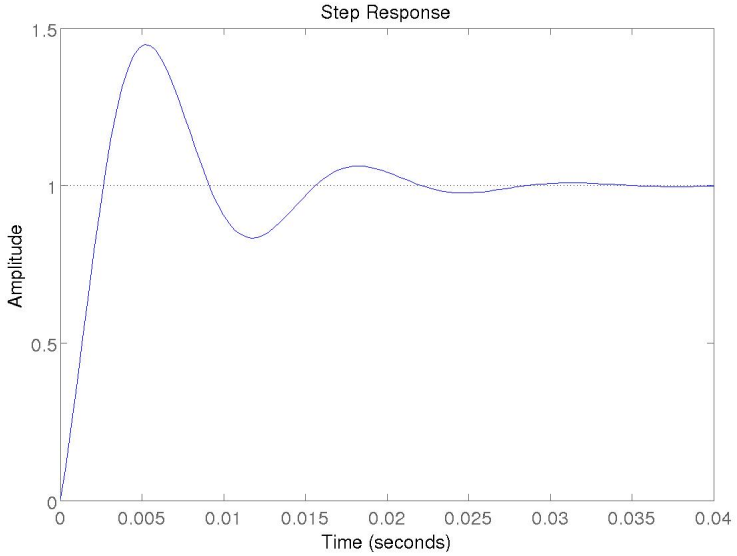

Fig. 8: Step response for system slacked ζ bounds.

3) Dominant Poles: We increase the dominant poles requirement to now require the magnitude of the zero in the transfer function to be 10 times the magnitude the real part of the dominant pole. This changes constraints given by Equation (23b) and Equation (23c) to be:

$$\frac{K_2}{K_1} \leq -(10)(10K_1 + 2.2)$$

$$\frac{K_2}{K_1} \geq (10)(10K_1 + 2.2)$$

We find:

$$K_1 = 0.1000$$

$$K_2 = 3.2000$$

With a corresponding rise time of:

$$t_r = 0.2704\text{s}$$

This should not be compared with the previous results, as here we are changing the problem in a fundamental way. We are changing the permissible range of the zero, therefore we impact a parameter which allows for our approximation to even hold. Hence, we are changing the very nature of our approximation. To check the effect of this in comparison to the initial problem, we view the difference between our calculated rise time, and the rise time given by Matlab's "stepinfo." Matlab's function gives a rise time of $t_r = 0.1746\text{s}$. This is a difference of $t_\Delta = 0.0958\text{s}$, with a factor of 1.6 difference between the two values. Therefore, by having the approximation restriction more stringent, we do increase the accuracy of the approximated function, as expected.

VI. Conclusions and Future Work

The optimal K_1 and K_2 values which satisfy the constraints and give us the optimal rise time were found to be $K_1 = 0.3667$ and $K_2 = 10.7556$, with an optimal rise time of $t_r = 0.1475$s. We showed that the solution is insensitive to changes in constraints given by Equation (23d) and Equation (23e). We showed that the solution is sensitive to changes in constraints given by Equation (23b), Equation (23c), and Equation (23f). This sensitivity was justified through control theory.

By using the dominant poles method to give a closed form solution of the rise time expression, we reduce the accuracy of the performance criterion. However, this is necessary, as the only other options either produce a lengthy (many page) expression for rise time, or give numerical approximations for rise time. As it is required that we rigorously prove that a minimum exists for the rise time objective function, such alternates are ruled out.

Initially, we attempted to formulate a general proof for showing the existence of a minimum. This method would not call for a closed form expression of rise time, and would use real analysis to show that the rise time function is guaranteed to be continuous [22]. From there, we would be able to apply Weierstrass' theorem to show that a minimum value exists. This would be especially useful for higher order systems, as a closed form expression for rise time fails to exist past order two, unless the dominant pole method or other methods are applied to model the system as a second order.

The method involves showing that continuity is preserved through a series of steps. The first step would be to show that the inverse Laplace transform of the step response, $c(t)$, is continuous. Next, we show that the inverse function of the response, $t(c)$, is continuous. Then, we show that the "first crossing function" (the time when the inverse function reaches a specified $c(t)$) of the inverse function at $c(t) = 0.1$ and $c(t) = 0.9$ is continuous. By subtracting these "first crossing functions", we would have an expression for rise time. If the operations were shown to preserve continuity, then the rise time function is shown to be continuous. This method is powerful in that we do not necessarily need to evaluate all the expressions, but can still show that continuity is preserved. However, the mathematics proved way beyond our abilities, and such a method was abandoned.

This sidelined any attempt to move to a higher order representation of the system dynamics. We attempted to use fourth order system dynamics (fifth order closed loop transfer function) [8] but this proved too challenging. Use of even a second order representation of pitch rate dynamics makes it very difficult to keep our performance expressions in a closed form. Even by applying the dominant pole method to second order pitch rate dynamics (third order system) [12], we found the percent overshoot is necessarily very large (over 80%). Therefore, in our paper we only explore the problem with first order pitch rate dynamics.

A similar issue was encountered when expanding our work to include a derivative controller. This control element added an additional zero into our closed loop transfer function, which made our previous rise time approximation infeasible. The only option in this case was the long and calculation expensive method discussed earlier, which proved beyond our mathematical ability.

The necessity in proving rigorously that a minimum exists also impedes efforts to raise the order of the pitch rate dynamics. In fact, it is almost definite that higher order systems cannot be optimized with the requirement that existence of a minimum be proved.

With the current restrictions on the project, future work in optimizing rise time for pitch rate control systems is very difficult. Even if the controls scheme is changed, the same issues would be present, and the rise time expression would be very difficult to find. Future research into a

performance criterion that can be easily be expressed in a closed form would prove very useful for controls engineers.

REFERENCES

[1] K. Ogata, *Modern Control Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[2] T. Benson. (2008) Aircraft rotations. [Online]. Available: http://www.grc.nasa.gov/WWW/k-12/airplane/rotations.html

[3] D. McLean, "Globally stable nonlinear flight control system," *IEEE Proceedings Pt. D Control Theory and Applications*, vol. 130, no. 3, p. 93, May 1983.

[4] D. Saussie et al., "Aircraft pitch rate control design with guardian maps," in *18th Mediterranean Conf. Control Automation*, June 2010, pp. 1473 –1478.

[5] S. Kamalasadan and A. Ghandakly, "Nonlinear fighter aircraft pitch-rate tracking using a multiple fuzzy reference model adaptive controller," in *IEEE Int. Conf. Computational Intelligence for Measurement Systems and Applications*, July 2005, pp. 44 – 49.

[6] S. Kamalasadan and A. Ghandakly, "A neural network parallel adaptive controller for fighter aircraft pitch-rate tracking," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 1, pp. 258 –267, Jan. 2011.

[7] C. Hao et al., "Sliding mode controller design for an aircraft pitch rate track system," in *IEEE Int. Conf. on Networking, Sensing and Control*, April 2008, pp. 1004 –1007.

[8] S. M. Shinners, *Advanced Modern Control System Theory and Design*. New York: Wiley, 1998.

[9] F. Demourant et. al, "Falsification of an aircraft autopilot," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 1, 2002, pp. 803 – 808 vol.1.

[10] L. Jinli and D. Manfeng, "Fuzzy adaptive pi control for near space aircraft electric propulsion system," in *Int. Conf. Computer, Mechatronics, Control and Electronic Engineering*, vol. 4, Aug. 2010, pp. 472 –475.

[11] G. J. Balas et al., "Control of the f-14 aircraft lateral-direction axis during powered approach," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 6, pp. 258 –267, Nov. 1998.

[12] C. Barbu et al., "Anti-windup design for manual flight control," in *Proc. American Control Conf.*, vol. 5, 1999, pp. 3186 –3190.

[13] E. G. Rynaski, "The interpretation of flying qualities requirements for flight control system design," unpublished.

[14] J. W. Brown and R. V. Churchill, *Complex Variables and Applications*. New York: McGraw-Hill, 2009.

[15] S. M. Shinners, *Modern Control System Theory and Design*. New York: Wiley, 1998.

[16] M. Gopal, *Control Systems Principles and Design*. New Delhi, India: Tata McGraw Hill, 2002.

[17] H. K. Ahmad, private communication, 2011.

[18] E. K. Chong and S. H. Zak, *An Introduction to Optimization*. Hoboken, NJ: Wiley, 2008.

[19] *LOQO Users Manual*, R. J. Vanderbei, 2006.

[20] Product documentation: fmincon. [Online]. Available: http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html

[21] Product documentation: stepinfo. [Online]. Available: http://www.mathworks.com/help/toolbox/control/ref/stepinfo.html

[22] S. M. Mintchev, private communication, 2011.

VII. MATLAB CODE

The following script runs the solution algorithm. We use the active set algorithm with the fmincon solver to find the optimal solution.

```
% This script runs the solution algorithm. We use the active set algorithm
% with the fmincon solver to find the optimal solution.

clear;close all;clc;
k0 = [0.1 2.1]; % starting guess

options = optimset('Algorithm','active-set');
[k,fval] = fmincon(@rise_time,k0,[],[],[],[],[-1],[15],@constraints,options);

% Print optimal k values and rise time
k
rt = rise_time(k)

% Plot step response
den_a = 1;
den_b = (20.*k(1) + 4.4);
den_c = 20.*k(2);
step(tf([20*k(1) 20*k(2)],[den_a den_b den_c]))
```

The following function gives rise time for an input K vector. This serves as our objective function.

```
% Function that gives rise time for an input K vector. This serves as our
% objective function.

function f = rise_time(k)
zeta = (20.*k(1) + 4.4) ./ (2.*(20.*k(2)).^(1/2));
w_n = (20*k(2))^(1/2);
f = (pi - acos(zeta)) / ((w_n)*(1 - zeta^2)^(1/2));
```

The following function serves to call our constraints for the fmincon solver.

```
% This function serves to call our constraints for the fmincon solver.

function [c,ceq] = constraints(k)

% defining denominator of transfer function (coefficients of polynomial)
den_a = ones(size(k(1)));
den_b = (20.*k(1) + 4.4);
den_c = 20.*k(2);

% stability constraints: by ensuring all poles are to the left of sigma =
% -0.5 in the s-plane. these constraints were determined by application of
% the modified routh-hurwitz criterion.

c1 = -3.4/20 - k(1);
c2 = 10.*k(1) - 20.*k(2) + 1.95;

% zero suppression constraint: to suppress the effects of the zero in the numerator of the ↩
    transfer
% function. doing so will allow us to use the standard second order
% expressions for response characteristics.

c3 = -abs(-k(2)./k(1)) + (5+eps).*abs(-10.*k(1) - 2.2);

% constraint on the damping ratio (zeta)

zeta = (20.*k(1) + 4.4) ./ (4.*(5.*k(2)).^(1/2));
c4 = (0.4 - zeta);
c5 = (zeta - 0.7);
```

```
28
29 % constraint vector in form c <= 0
30
31 c = [c1; c2; c3; c4; c5];
32
33 ceq = [];
```

The following script plots the feasible set to produce Figure 4. Note the commented code on the bottom of the script, which creates Figure 5.

```
1  % Code that plots feasible set for rise time optimization problem for second order system.
2
3  clear;clc;close all;
4
5  % test points
6  [k1,k2] = meshgrid(-1:.01:15);
7
8  % pre-allocating memory
9  c1 = zeros(size(k1));
10 c2 = zeros(size(k1));
11 c3 = zeros(size(k1));
12 c4 = zeros(size(k1));
13 c4_a = zeros(size(k1));
14 c4_b = zeros(size(k1));
15 den_a = zeros(size(k1));
16 den_b = zeros(size(k1));
17 den_c = zeros(size(k1));
18 zeta = zeros(size(k1));
19
20
21
22 % defining denominator of transfer function (coefficients of polynomial)
23 den_a = ones(size(k1));
24 den_b = (20.*k1 + 4.4);
25 den_c = 20.*k2;
26
27
28
29 % stability constraints: ensuring all poles are to the left of sigma =
30 % -0.5 in the s-plane. these constraints were determined by application of
31 % the modified routh-hurwitz criterion.
32
33 c1 = (k1 >= -3.4/20);
34 c2 = (-10.*k1 + 20.*k2 -1.95 >= 0);
35
36
37 % zero suppression constraint: to suppress the effects of the zero in the numerator of the ↩
       transfer
38 % function. doing so will allow us to use the standard second order
39 % expressions for response characteristics.
40
41 c3 = (abs(-k2./k1) >= (5+eps).*abs(-10.*k1 - 2.2));
42
43 % constraint on the damping ratio (zeta)
44
45 zeta = (20.*k1 + 4.4) ./ (4.*(5.*k2).^(1/2));
46 c4_a = (0.4 <= zeta);
47 c4_b = (zeta <= 0.7);
48 c4 = c4_a.*c4_b;
49
50
51 % vector that sums all successful constraints
52 z = c1+c2+c3+c4;
53
54 % number of feasible points: displayed on screen
55 size(find(z == 4))
56
57
58  %% Plot
```

```
59
60  val = find(z == 4);
61  plot(k1(val),k2(val),'ks');
62  hold on;
63
64 % below is the plot of the curve used in the continuity proof
65 %  x = [-1:.01:10];
66 %  y = (1/5).*(25.*x.^2 + 11.*x + 1.21);
67 %  plot(x,y);
```