# An investigation of the performance of a self-tuning control system on a nonlinear aircraft model using control allocation

**Carl Banks**
**ECE 6414**

## I. Introduction

TRADITIONALLY, airplanes employ three controls: ailerons, elevator, and rudder, for control about their three axes (the roll, pitch, and yaw axes, respectively.) Modern, high-performance fighter airplanes often come with many more than the traditional three controls; however, the requirement for control about three axes remains the same. The control systems of such aircraft often include a control allocation system, also called a control mixer, operating inside the feedback loop.

The damage that these same modern, high-performance fighters are apt sustain as part of their mission can cause significant changes to their dynamics. This can be a serious problem, especially for airplanes that are inherently unstable, as modern fighters often are. Consequently, these aircraft employ adaptive flight control systems

The major problem with adaptive flight control is what happens when the airplane is not maneuvering. Airplanes spend much time cruising, with all states and controls more or less constant. Such long periods have adverse effects on the parameter identification.

This report will investigate a simple nonlinear self-tuning control system. The airplane dynamics are modeled by a global nonlinear polynomial model of an F-16 [1]. I shall investigate longitudinal motion only (that is, motion in the plane of symmetry about the pitch axis). The controlled variable is the angular velocity about the pitch axis, $q$. We use only one control, the elevator deflection, $\delta_e$ (although it seems, and probably is, silly to use control allocation for just one control). The parameter identification scheme is sequential least squares with a forgetting factor.

## II. Description of Simulation

### A. System Dynamics

Although the simulation models the entire airplane dynamics, the adaptive controller is only concerned with the pitching moment equation. The pitching moment equation for a symmetric airplane in longitudinal motion only is given, quite simply, by Equation 1.

$$\dot{q} = M/I_{yy} \tag{1}$$

where $I_y y$ is the moment of inertia about the pitch axis.

The pitching moment $M$ is more easily modeled when nondimensionalized, so write it in terms of the pitching moment coefficient, $C_M$:

$$M = \frac{1}{2}\rho V^2 S\bar{c} C_M \tag{2}$$

Here, $\rho$ is the outside air density, $V$ is the true airspeed, $S$ is the wing area, and $\bar{c}$ is the wing chord. These are all known values.

The pitching moment coefficient is a function of the state and control variables, and is modeled by the higher-order polynomial given by Equation 3 [1].

$$\begin{aligned} C_M &= m_0 + m_1\alpha + m_2\delta_e + m_3\alpha\delta_e + m_4\delta_e^2 \\ &\quad + m_5\alpha^2\delta_e + m_6\delta_e^3 + m_7\alpha\delta_e^2 + C_{M_q}(\alpha)\tilde{q} \end{aligned} \tag{3}$$

The polynomial coefficients $m_0$, $m_1$, etc., are, of course, the parameters to be identified. $\alpha$ is the angle of attack, is a function of the states, and is considered known. The term $C_{M_q}(\alpha)\tilde{q}$ is of no relevance to this report.

### B. Feedback Controller

For this control system, the output variable is the pitch rate, $q$. The reference signal $q_{\mathrm{cmd}}$ comes directly from the pilot's control stick. The control is $\delta_e$, the elevator deflection.

We now invent a new abstract control, namely $\dot{q}_{\mathrm{cmd}}$. This is the commanded angular acceleration, and the feedback controller will treat it as the control variable. Once the controller has determined $\dot{q}_{\mathrm{cmd}}$, the control mixer will choose $\delta_e$ to yield the desired $\dot{q}$, using Equations 1–3.

The controller design was modeled after the pitch-rate control augmentation system from Reference [2], except that the $\alpha$ feedback loop is not present. Figure 1 presents the block diagram for the control system.

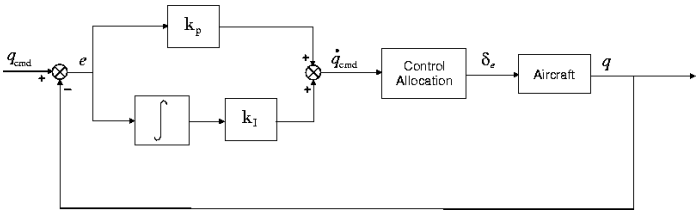


Fig. 1. Block diagram of control system for this report

The values of the gains are $K_p = -25$ and $K_I = -1$, which I admit I found mostly by trial and error. These values give good performance. Figure 2 shows the performance of the control system using the known aircraft model (i.e., not the estimated parameters). It plots $q$ and $q_{\mathrm{cmd}}$ for a pull-up maneuver (note that $q$ is negative for a pull-up), and we see that $q$ follows $q_{\mathrm{cmd}}$ quite well, with only about a 0.1 second lag.

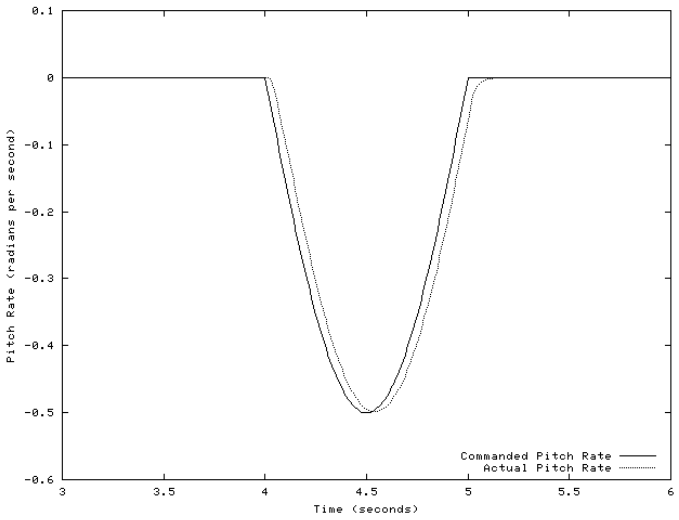


Fig. 2. Commanded and actual pitch rates for a pull-up maneuver (not using parameter identification).

### C. Mixer

The control mixer must choose $\delta_e$ to make $\dot{q} = \dot{q}_{\mathrm{cmd}}$. Given $\dot{q}_{\mathrm{cmd}}$, we can determine $M_{\mathrm{cmd}}$ via Equation 1, and in turn $C_{M\mathrm{cmd}}$ via Equation 2.

In this simplified case, we could solve Equation 3, which is cubic in $\delta_e$, directly. However, more realistic problems will not only have more complex models, but will also be overdetermined (since there are more controls than controlled variables). Therefore, we will determine $\delta_e$ by linearizing Equation 3 about the current condition and solving that.

For the linearization, we could determine the current moment coefficient using Equation 3. But we already have an observed value of $C_M$ (which we call $C_{M_0}$), because the parameter identification requires it. In real life, $C_{M_0}$ is the output of some sort of Kalman filter; this is certain to be more accurate than the $C_M$ yielded by Equation 3, so we use it instead. We subtract this from $C_{M\mathrm{cmd}}$ to yield $\Delta C_{M\mathrm{cmd}}$.

We want $\Delta C_M$ to equal $\Delta C_{M\mathrm{cmd}}$. Expanding $\Delta C_M$ as a first-order Taylor series, we get:

$$\Delta C_M = C_M - C_{M_0} = \frac{\partial C_M}{\partial \delta_e}\Delta\delta_e + \cdots \tag{4}$$

We ignore all other terms and solve for $\Delta\delta_e$:

$$\Delta\delta_e = \frac{\Delta C_M}{\partial C_M/\partial\delta_e} \tag{5}$$

From Equation 3, we see that $\partial C_M/\partial\delta_e$ is given by:

$$\begin{aligned}\frac{\partial C_M}{\partial \delta_e} &= m_2 + m_3\alpha + m_5\alpha^2 \\ &\quad +2(m_4 + m_7\alpha)\delta_e + 3m_6\delta_e^2\end{aligned} \tag{6}$$

The mixer uses Equations 5 and 6, with $\Delta C_{M\mathrm{cmd}}$, to determine $\Delta\delta_e$.

There is a certain advantage to this approach to a flight control system. The feedback controller does not need to concern itself with the complicated relationship between $q_{\mathrm{cmd}}$ and $\delta_e$; now it is only concerned with the straightforward relationship between $q_{\mathrm{cmd}}$ and $\dot{q}_{\mathrm{cmd}}$. It spares the feedback gains from becoming intertwined with the effectiveness of $\delta_e$ at a particular flight condition.

Operating within the feedback loop, the mixer handles the dirty work of determining $\delta_e$. The mixer provides, in effect, a gain scheduling for $\delta_e$. $1/(\partial C_M/\partial\delta_e)$ acts like a gain; and varies as the effectiveness of $\delta_e$ varies.

### D. Parameter Identification

For on-line parameter identification, these simulations use a sequential least squares algorithm [3] (sometimes called recursive least squares, although it's not recursive) with a forgetting factor. This particular algorithm is known to have conditioning problems when the inputs vary slowly [4].

Equations 7 and 8 give the parameter vector and input vector, respectively, for the least squares algorithm.

$$\theta = \begin{bmatrix} m_0 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & \dots \end{bmatrix}^T \tag{7}$$

$$\mathbf{w} = \begin{bmatrix} 1 & \alpha & \delta_e & \alpha\delta_e & \delta_e^2 & \alpha^2\delta_e & \delta_e^3 & \alpha\delta_e^2 & \dots \end{bmatrix}^T \tag{8}$$

The output is $C_{M\mathrm{est}} = \mathbf{w}^T\theta$, and the output error is $C_{M\mathrm{est}} - C_{M_0}$.

The update formula used is given by Equations 9 and 10.

$$\mathbf{P}_{k+1} = \lambda^{-1}\mathbf{P}_k - \frac{\lambda^{-2}\mathbf{P}_k\mathbf{w}_k\mathbf{w}_k^T\mathbf{P}_k}{1 + \lambda^{-1}\mathbf{w}_k^T\mathbf{P}_k\mathbf{w}_k} \tag{9}$$

$$\theta_{k+1} = \theta_k + \mathbf{P}_{k+1}\mathbf{w}_k(C_{M\mathrm{est}} - C_{M_0}) \tag{10}$$

where $\mathbf{P}_k$ is the covariance matrix at step $k$, and $\lambda$ is the forgetting factor. $\mathbf{P}_0$ is initialized to $\epsilon I$, where $\epsilon$ is a small positive.

### E. Implementation Notes

The simulation makes a few simplifications which the following paragraphs explain.

First, the sensors and controls are assumed to have no lag. Moreover, the simulation does not model observer dynamics; variables such as $C_M$ are simply used directly. However, Gaussian noise is added to the sensed and observed variables before use by the controller and parameter estimator.

The simulation does not check control rate limits, but it does check position limits.

The simulation models a continuous control system. It uses the same time step, 0.01 seconds, as the airframe. The parameter identification also operates 100 times per second; however, it shall be seen that intentionally slowing it can help conditioning problems.

## III. Simulation Results

We have already said that sequential least squares has conditioning problems for cruising flight, and in the simulations, it did not disappoint.

Each simulation starts out from a trimmed condition: with no noise, an input signal of zero, and using the known model, the airplane would fly straight and level forever. Every simulation initialized the parameters to their known values. Also, every

simulation added Gaussian noise to the variables used by the controller and estimator.

The simulations were run with different forgetting factors and least squares update rates. Some of the simulations modeled elevator damage occurring at $t = 200$ s. The control's effectiveness was halved by adjusting the appropriate parameters in the known model. Some simulations modeled a pull-up maneuver at $t = 800$ s, followed by an opposite push-down maneuver 10 seconds later.

The first few simulations, however, were straight and level throughout. Figure 3 shows the simulation with $\lambda = 0.99$ and a least squares update rate of 100 times per second. It only took about 11 seconds for the parameters to blow up. Knowing that

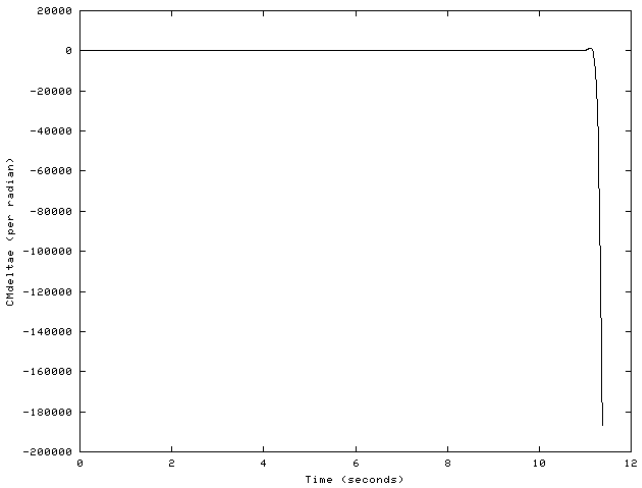


Fig. 3. Time history of the parameter $m_2$, for $\lambda = 0.99$.

increasing $\lambda$ can improve conditioning, at the expense of transient performance, I tried $\lambda = 0.999$. This time the parameters lasted almost 100 seconds before blowing up, as seen in Figure 4.

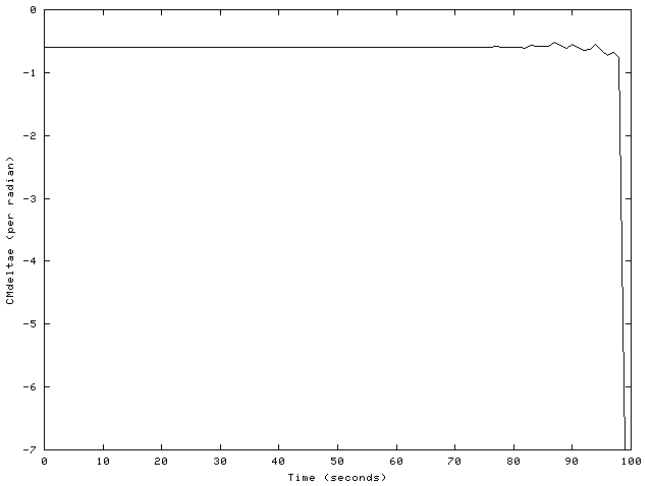


Fig. 4. Time history of the parameter $m_2$, for $\lambda = 0.999$.

It seems logical to just eliminate the forgetting factor altogether. Figure 5 shows the percent error for six of the parameters, for a simulation run with $\lambda = 1$. The change in values of all parameters remained well under one percent for the 20 minutes simulated.

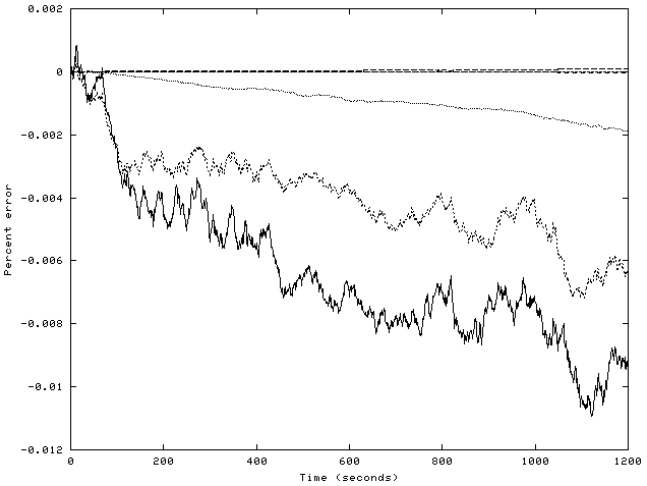


Fig. 5. Time history of all the $C_M$ parameters, for $\lambda = 1.0$ (no forgetting factor).

Having found a configuration where the parameters do not blow up, let us look at its response to some damage sustained by the control. Figure 6 shows the actual and estimated values of one parameter, $m_2$. At $t = 200$ s, the damage occurs, which halves the parameter's actual value. The estimate stays near the actual value until the damage occurs. But, curiously, after the damage, it moves *away* from the new actual value.

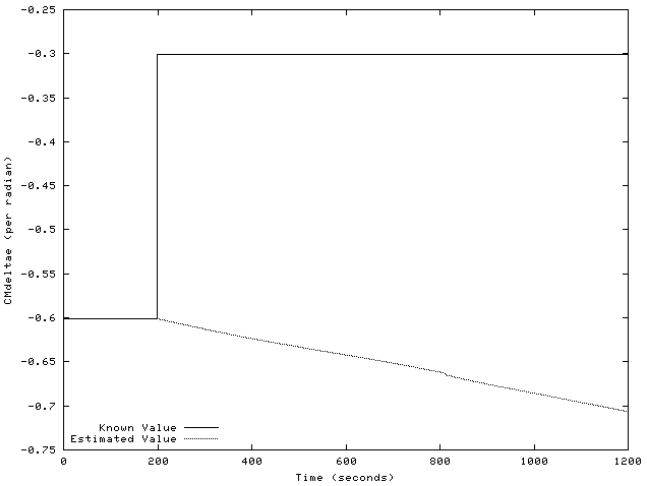


Fig. 6. Time history of the parameter $m_2$, along with its known value, for $\lambda = 1.0$. At $t = 200$ s, the elevator effectiveness dropped by half.

How does this affect performance? Figure 7 shows the pitch-rate of a pull-up maneuver performed ten minutes after the damage occurred, along with the commanded rate. The performance is very good, even though the estimated parameters are significantly different from the actual values. This is no doubt due to a robust feedback control; it tells us that a good feedback control can compensate for a poor system model quite well in some cases. For comparison, Figure 8 shows the same pull-up maneuver performed with no damage to the elevator. There is hardly any apparent difference in their performances.

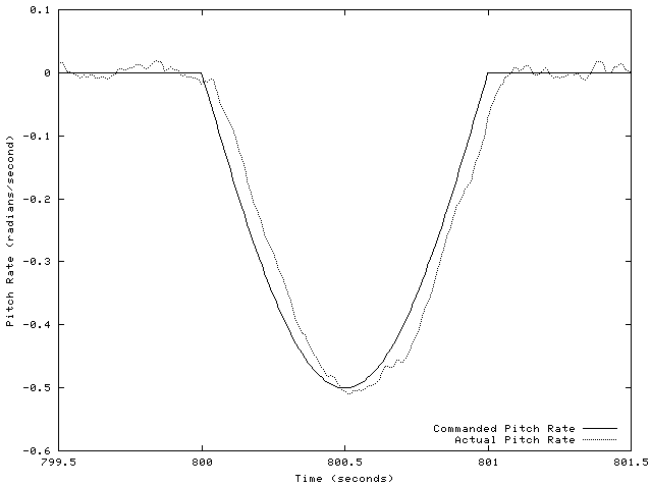


Fig. 7. Time history of the command and actual pitch rates, around $t = 800$ s, when a pull-up maneuver occurred. The conditions here the same as for Figure 6.

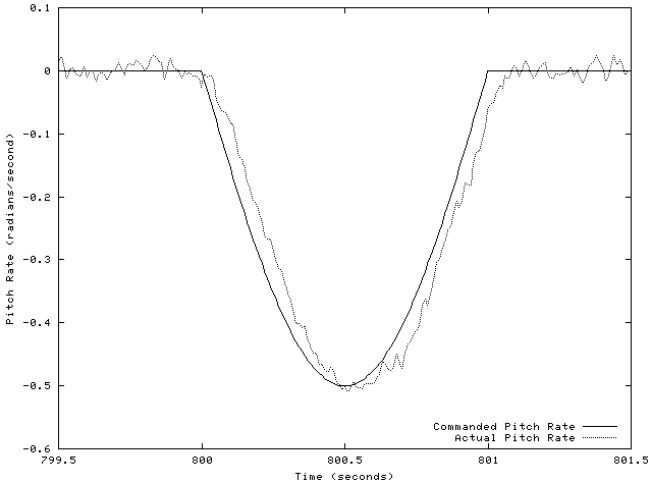


Fig. 8. Time history of the command and actual pitch rates, around $t = 800$ s, when a pull-up maneuver occurred. In this case, there was no damage to the elevator. $\lambda = 1.0$.

So far, when faced with the problem of poor conditioning in the estimator, we have attempted to remedy it by increasing (and ultimately eliminating) the forgetting factor. Although the airplane performed its maneuver well, the parameter estimates were unacceptable. At some point, having accurate parameters will be crucial, and a forgetting factor will be necessary for that.

Therefore, we try another remedy for the conditioning problem: decreasing the least squares update rate. The rationale for this is that increasing the time between updates gives the airplane more time to change its state. Too many samples with the nearly the same input is what causes the conditioning problem, so increasing the sample time could reduce the number of similar inputs.

In following simulations, the forgetting factor was set to 0.99, while the estimator was only updated once every ten time steps (ten times per second). Indeed, as Figure 9 shows, the parameter $m_2$ (which is the coefficient of $\delta_e$) does not blow up. In fact, after damage occurs, it migrates to the new value. (That is, the estimator does what it's supposed to do.)

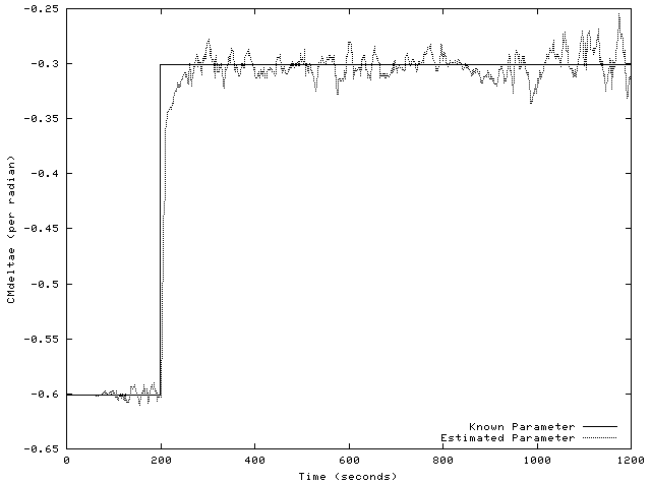


Fig. 9. Time history of the parameter $m_2$, along with its known value, for $\lambda = 0.99$, but at a lower sampling rate. At $t = 200$ s, the elevator effectiveness dropped by half.

However, the result is not so good when looking at higher order parameters. Figure 10 shows $m_3$, the coefficient of $\delta_e\alpha$. Here, although the parameter doesn't blow up, it varies from the known value by orders of magnitude more than the parameter's value itself. The other parameters governing a term with $\alpha$ behave even worse. The reason for this poor behavior is obvious:

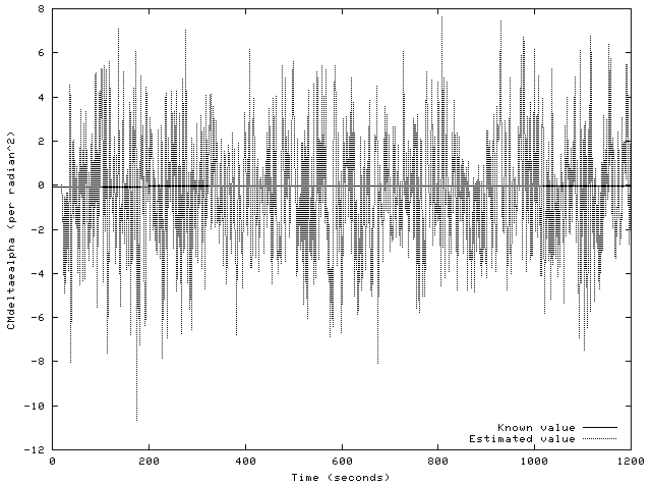


Fig. 10. Time history of the parameter $m_3$, along with its known value. The conditons are the same as for Figure 9.

while the airplane is cruising, the angle of attack changes little. The estimator cannot collect data for a wide range of $\alpha$, and so paramters depending on it are poorly identified. This is especially true for the coefficients of terms involving higher powers of $\alpha$, which are dominated by the lower order terms during cruise.

Figure 11 looks briefly at the control system performance under these conditions. The performance is mostly quite good. However, near the end of the maneuver, the airplane departs from its commanded motion before recovering. The temporary departure is likely because a pull-up maneuver involves a sud-

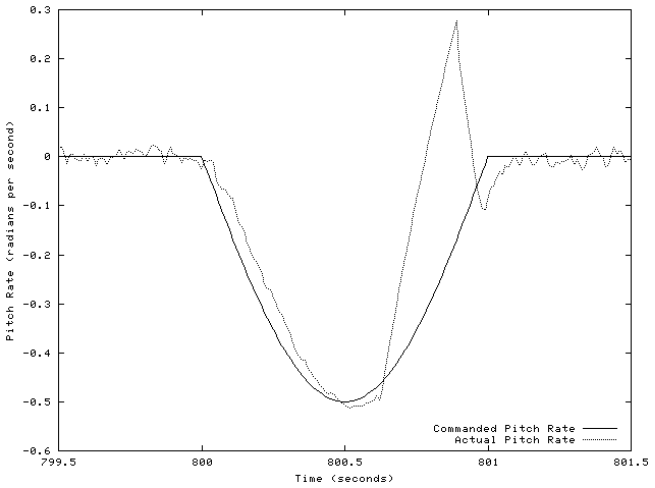


Fig. 11. Time history of the command and actual pitch rates, around $t = 800$ s, when a pull-up maneuver occurred. The conditions are the same as for Figure 9.

den and temporary increase in $\alpha$. At high angles of attack, the poorly identified parmeters such as $m_3$ and $m_5$ come to dominate, resulting in the temporary departure. Here, the aircraft recovered, but more serious cases could result in loss of control.

One significant fact about the previous simulation is that its success depended on a certain amount of noise present. When the simulation was run with less noise, the parameters once again blew up.

## IV. Conclusions

We have observed one of the serious problems with online parameter identification used for flight control: the ill-conditioning problem that can result in the estimated parameters blowing up when the airplane is cruising. The sequential least squares algorithm with a forgetting factor is highly susceptible to this.

We saw two remedies for this problem. The first was to increase or eliminate the forgetting factor. This resulted in acceptable controller performance for the maneuver attempted. However, the lack of a forgetting factor was very detrimental to the accurate estimation of the parameters.

The second remedy was to decrease the least squares update rate. This resulted in an estimator for which the parameters did not blow up, and one of the paramters was successfully estimated. However, higher order paramters were not successfully estimated. Also, the robustness is to be doubted, for it seems to depend on a certain amount of noise for its stability.

Finally, we saw another serious problem with parameter identification: the poor estimation of certain paramters that govern higher order terms during cruise. This appeared to cause a temporary departure in one of the simulations.

## References

[1] Eugene A. Morelli, "Global nonlinear parametric modelling with application to F-16 aerodynamics," in *Proceedings of the American Control Conference*, Philadelphia, June 1998, pp. 997–1001.

[2] Brian L. Stevens and Frank L. Lewis, *Aircraft Control and Simulation*, Wiley, New York, 1992.

[3] Jerry M. Mendel, *Discrete Techniques of Parameter Estimation: The Equation Error Formulation*, Control Theory: A Series of Monographs and Textbooks. Marcel Derker, Inc., New York, 1973.

[4] Marc Bodson, "An adaptive algorithm with information-dependent data forgetting," in *Proceedings of the American Control Conference*, Seattle, WA, June 1995, pp. 3485–3489.