Table 1 Computational times for jitter analysis

Algorithm	0.1-s window, s	1-s window, s	10-s window, s
Direct (script language)	152.8	936.1	8665.3
Vectorized (script language)	9.5	9.9	10.4
Direct (C-based compiler)	9.8	87.6	1066.2
Vectorized (C-based compiler)	3.9	4.1	6.6

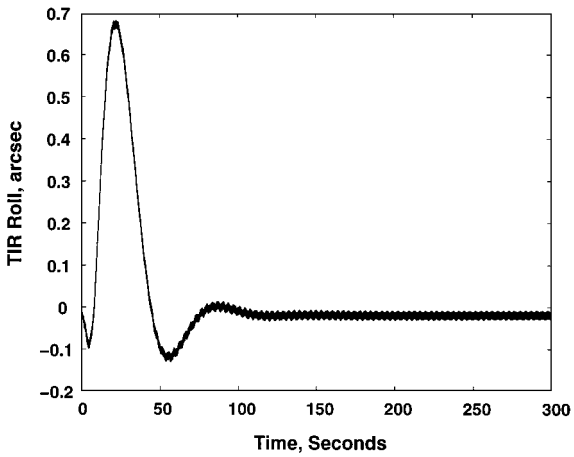

Fig. 1 Time response for Measurements of Pollution in the Tropespheric (MOPITT) instrument cryocooler disturbance sequence.

with the vectors $\boldsymbol{y}_{\max}$, $\boldsymbol{y}_{\min}$, $\boldsymbol{l}_{\max}$, and $\boldsymbol{l}_{\min}$ computed for the first jitter window (previous window). The same procedure is followed for the subsequent jitter windows, starting each time with the information on the extrema computed for the previous jitter window, until all jitter values are computed.

Numerical Example

To demonstrate the feasibility of the proposed jitter analysis algorithm, it has been applied in the computation of jitter values for the thermal infared Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER/TIR), a science instrument on the Earth observing system (EOS) AM-1 spacecraft,[5] a NASA Earth observation and remote sensing mission. A closed-loop simulation of the spacecraft was performed with an instrument cryocooler disturbances as the excitation source. A 300-s time history of the roll response of ASTER/TIR is presented in Fig. 1. With the sampling period at 0.001 s, the roll response resulted in a 300,001-element output vector (corresponding to 300,000 time steps). Jitter values were computed for three windows at 0.1, 1, and 10 s, using the proposed vectorized approach and the direct approach. Both techniques provided jitter values of 0.02, 0.09, and 0.63 arcsec, corresponding to the 0.1-, 1-, and 10-s, windows, respectively. The formulations were implemented in the MATLAB[4] computational environment. Both MATLAB script language implementation, which is amenable to vector operations, and the C-based compiler optimized implementation, which is optimal for looping, vector, and scalar operations, were used. The timing results are presented in Table 1. It is noted that the times reported for each jitter window are based on a jitter analysis for a single time window from scratch, i.e., the longer windows do not leverage off of the shorter windows.

The feasibility and efficiency of the proposed vectorized algorithm are clearly observed from Table 1. The superiority of the vectorized jitter algorithm becomes drastically profound as the size of the jitter window increases, approaching orders of magnitude reduction in the computational time.

Concluding Remarks

A vectorized algorithm for efficient computation of spacecraft jitter values has been presented. The algorithm identifies the extreme points in the time history, which are the points that may dominate the jitter values depending on the location of the jitter window along the time history. The span of influence of each extremum is then computed by the algorithm and used in an efficient and vectorized fashion to obtain the jitter values. The algorithm is sequential, i.e., it starts with the smallest jitter window and works its way to the largest window requested. The algorithm has been successfully applied to the computation of jitter values for a science instrument on the EOS AM-1 spacecraft. The results indicate that the proposed algorithm reduces the required computational time by several orders of magnitude, thereby demonstrating the feasibility of the approach.

References

[1] "EOS-A Pointing and Orbit Requirements (AFM T-9)," GE Aerospace, NAS5-32500, Princeton, NJ, Aug. 1991.

[2] "EOS-A Pointing and Orbit Study Update (AFM T-9)," GE Aerospace, NAS5-32500, Princeton, NJ, May 1992.

[3] Giesy, D. P., "Efficient Calculation of a Jitter/Stability Metric," *Journal of Spacecraft and Rockets*, Vol. 34, No. 4, 1997, pp. 549–557.

[4] "MATLAB Reference Guide—High-Performance Numeric Computation and Visualization Software," The MathWorks, Inc., Natick, MA, July 1993.

[5] Asrar, G., and Dokken, D. J. (eds.), "1993 Earth Observing System Reference Handbook," NASA NP-202, 1993.

J. D. Gamble
Associate Editor

Iterative Learning-Based Extraction of Aerobomb Drag

Yangquan Chen* and Jian-Xin Xu†
National University of Singapore, 119260 Singapore
and
Changyun Wen‡
Nanyang Technological University, 639798 Singapore

Introduction

THIS Note focuses on extracting a single aerodynamic drag coefficient curve of an aerobomb from three-dimensional theodolite film data, i.e., from the measured spatial positions of the aerobomb. The aerobomb's drag coefficient curve plays a crucial role in increasing the bombing accuracy of aircraft. When bombing, the mechanism of the interference air flowfield between the aircraft and the aerobomb is not yet clear; therefore, the drag coefficient curve obtained from wind-tunnel measurements or from theoretical numerical prediction under the free airflow condition cannot be applied directly. The curve reduced from flight testing data is obviously advantageous in practical applications. Many efforts have been made in extracting aerodynamic properties of real or full-scale flying vehicles from flight testing data.[1,2] Most of the literature, however, emphasizes the aerodynamic coefficient extraction by parameter identification that is clearly a special case of curve extraction. In practice, to fully utilize the flight testing data, aerodynamic curve extraction or identification is preferred. The optimal dynamic fitting method[3,4] has been used to directly extract the aerodynamic coefficient curve. However, this method suffers from computational complexity and the need of a good guess of the initial curve.

A new idea, iterative learning,[5] is introduced and applied. By properly setting the learning gain, the learning convergence can be guaranteed. The convergence is shown to be quite robust to the initial control guess. To improve the learning convergence, several

Received Aug. 7, 1997; revision received Dec. 17, 1997; accepted for publication Dec. 17, 1997.

*Research Assistant, Department of Electrical Engineering, 10 Kent Ridge Crescent. E-mail: yqchen@shuya.ml.org.

†Senior Lecturer, Department of Electrical Engineering, 10 Kent Ridge Crescent. E-mail: elexujx@nus.sg.

‡Senior Lecturer, School of Electrical and Electronic Engineering, Nanyang Avenue. E-mail: ecywen@ntu.edu.sg.

schemes for the learning gain determination are discussed. Results from actual flight testing data for three bombing flight paths are presented to validate the effectiveness of the proposed iterative learning scheme. The extracted results are convincing and comprehensively reflect the effects of interference air flowfield and the angular motion around the center of mass. Conventionally, the bombing table generation is based on a single ballistic coefficient, a drag law, and some fitting factors. A more accurate firing table can be produced when the extracted C_{df} of this Note is utilized directly.

Problem Formulation

Suppose at time t, the aerobomb's position in the Earth coordinate system (ECS) is $[x(t), y(t), z(t)]^T$, and its velocity $\boldsymbol{u}$ with respect to ECS is $[u_x(t), u_y(t), u_z(t)]^T$. We have

$$\begin{aligned} \dot{u}_x(t) &= -\rho s V(t)[u_x(t) - w_x(t)]C_{df}(t)/2m \\ \dot{u}_y(t) &= -\rho s V(t)u_y(t)C_{df}(t)/2m - g \\ \dot{u}_z(t) &= -\rho s V(t)[u_z(t) - w_z(t)]C_{df}(t)/2m \\ \dot{x}(t) = u_x(t), \qquad \dot{y}(t) &= u_y(t), \qquad \dot{z}(t) = u_z(t) \end{aligned} \tag{1}$$

where $\boldsymbol{X}(t) = [x(t), y(t), z(t), u_x(t), u_y(t), u_z(t)]^T$ is the state vector of system (1), g is the gravitational acceleration, $w_x(t)$ and $w_z(t)$ are wind components in ECS, $V(t)$ is the aerobomb's relative velocity with respect to wind, where

$$V(t) = \sqrt{[u_x(t) - w_x(t)]^2 + u_y^2(t) + [u_z(t) - w_z(t)]^2} \tag{2}$$

ρ is the air density, $s = \pi d^2/4$ is the reference area of the aerobomb, d is the aerobomb's reference diameter, and m is the mass of the aerobomb. $C_{df}(t)$ is the aerodynamic drag coefficient curve with respect to the trajectory model (1), which is regarded as a control function to be optimally determined.

Denote $\{x_m(t), y_m(t), z_m(t) \mid t = t_0, t_0 + h, \ldots, t_0 + Nh\}$ the measured positional trajectories of the aerobomb from theodolite films, where N is the number of points and h is the time step. By a proper use of the spline fitting method, the velocity data $u_m(t) = \sqrt{[u_{x_m}^2(t) + u_{y_m}^2(t) + u_{z_m}^2(t)]}$ can be obtained and, hence, $\boldsymbol{X}(t_0)$ is known. The initial state $\boldsymbol{X}(t_0)$ can also be obtained from testing plans and other recording devices. Taking $u_m(t)$ as the desired trajectory to be followed, the control objective is to minimize the tracking error $e(t) \triangleq u_m(t) - u(t)$, i.e.,

$$\min_{C_{df}} J[C_{df}] \triangleq \min_{C_{df}} e_b \triangleq \min_{C_{df}} \sup_{t \in [t_0, t_0 + Nh]} |e(t)| \tag{3}$$

where

$$u(t) = \sqrt{u_x^2(t) + u_y^2(t) + u_z^2(t)} \tag{4}$$

which can be regarded as the output equation for system (1). Clearly, Eqs. (1), (3), and (4) formulate an optimal tracking control problem (OTCP). This OTCP is a singular optimal control problem (SOCP) and the performance index is a minimax one, which is difficult to solve.

Curve Extraction by Iterative Learning

The iterative learning concept (ILC) is mainly applied to improve the control performance by utilizing the operational repetitions, i.e., the system is operated repeatedly for a given task.[5] A basic feature of ILC is that the control effort of the current iteration (repetition, cycle, pass, batch) is updated by using the control effort and the tracking error information of the preceding iteration. In other words, it learns from practice and experience.

The differences between ILC and iterative learning-based curve extraction are as follows. ILC, given a desired output trajectory, aims to solve a desired control function iteratively along with the system repetitive operations. Iterative learning-based curve extraction aims to extract an unknown nonlinear function, which is regarded as a virtual control function, iteratively from I/O data, which are taken as the desired output trajectory.

The key issue in implementing an iterative learning extraction method is how the system is operated repeatedly. In fact, one system repetition here is simply to numerically integrate the trajectory model equation (1) from t_0 to $t_0 + Nh$ with a known initial condition $\boldsymbol{X}(t_0)$ under the current control $C_{df}(t)$. Therefore, the iterative learning extracting procedures can be summarized as follows.

1) Set $k = 0$, and give an arbitrary $[C_{df}(t)]_0$. Specify a maximum error ε^* allowed and a maximum number of iteration N_i for the extraction.

2) Integrate the ballistic model (1) with $[C_{df}(t)]_k$ from t_0 to $t_0 + Nh$ with initial condition $\boldsymbol{X}(t_0)$. Then obtain the tracking error $e_k(t)$ according to Eq. (4).

3) Learning update

$$[C_{df}(t)]_{k+1} = [C_{df}(t)]_k + K(t)\dot{e}_k(t) \tag{5}$$

where $K(t)$ is a learning gain, which will be discussed in the next section. Learning updating law (5) is a conventional D type[5] as the tracking error derivative is used in (5).

4) If either $e_b < \epsilon^*$ or $k \geq N_i$, then go to step 6; otherwise go to step 5.

5) Next, $k = k + 1$, and store $[C_{df}(t)]_k$ and $\dot{e}_k(t)$ for use in the next repetition. Go to step 2.

6) Stop.

Selection of Learning Gain

The selection of learning gain is mainly based on the learning convergence condition, which is given for the following problem setting. Consider a class of repetitive nonlinear time-varying systems

$$\dot{\bar{\boldsymbol{x}}}_k(t) = \boldsymbol{f}[\bar{x}_k(t), t] + \boldsymbol{B}[\bar{\boldsymbol{x}}_k(t), t]\bar{\boldsymbol{u}}_k(t), \qquad \bar{\boldsymbol{y}}_k(t) = \boldsymbol{g}[\bar{\boldsymbol{x}}_k(t), t] \tag{6}$$

where $t \in [0, T]$ and T is given; k is the kth repetition of the system operation; $\bar{\boldsymbol{x}}_k(t) \in R^n$, $\bar{\boldsymbol{u}}_k(t) \in R^m$, and $\bar{\boldsymbol{y}}_k(t) \in R^r$ are the state, control input, and output of the system, respectively; the functions $\boldsymbol{f}(\cdot, \cdot)$, $\boldsymbol{B}(\cdot, \cdot) : R^n \times [0, T] \mapsto R^n$, and $\boldsymbol{g}(\cdot, \cdot) : R^n \times [0, T] \mapsto R^r$ are piecewise continuous. Given $\bar{y}_d(t)$, the realizable desired system output, denote $\boldsymbol{e}_k(t) \triangleq \bar{\boldsymbol{y}}_d(t) - \bar{\boldsymbol{y}}_k(t)$ the tracking error. The tracking control task can be done by applying the D-type ILC updating law

$$\bar{\boldsymbol{u}}_{k+1}(t) = \bar{\boldsymbol{u}}_k(t) + \boldsymbol{K}(t)\dot{\boldsymbol{e}}_k(t) \tag{7}$$

where $\boldsymbol{K}(t)$ is the bounded time-varying learning gain matrix of proper dimension to be properly determined. The following theorem gives a sufficient condition on $\boldsymbol{K}(t)$ for learning convergence. It is a simplified version considered in Ref. 6, where a proof can be found.

Theorem: Consider the preceding finite time trajectory tracking control task on a fixed interval $[0, T]$. By using the learning updating law (7), $\|\bar{\boldsymbol{y}}_d(t) - \bar{\boldsymbol{y}}_k(t)\|$ converges asymptotically to zero pointwise, provided that

$$\left\| I - \boldsymbol{K}(t)\frac{\partial \boldsymbol{g}[\bar{\boldsymbol{x}}(t), t]}{\partial \bar{\boldsymbol{x}}}\boldsymbol{B}[\bar{\boldsymbol{x}}(t), t] \right\| \triangleq \bar{\rho} < 1$$

$$\forall [\bar{x}(t), t] \in R^n \times [0, T] \tag{8}$$

In Eq. (8), $\bar{\rho}$ is the rate of convergence. A smaller $\bar{\rho}$ gives faster convergence.[5,6] A constant learning gain may be used, provided that in all time instants $\bar{\rho}$ is smaller than 1. To have the fastest convergence, one may set $\bar{\rho} = 0$ to get the best $\boldsymbol{K}(t)$. According to the theorem, from Eqs. (1) and (4), one obtains

$$\bar{\boldsymbol{g}}_{\bar{x}} = [u_x/u, u_y/u, u_z/u, 0, 0, 0]$$

$$\boldsymbol{B}[\bar{\boldsymbol{x}}(t), t] = -(\rho V s/2m)[u_x - w_x, u_y, u_z - w_z, 0, 0, 0]$$

Then applying Eq. (8) yields

$$K(t) = -\left\{(\rho V s/2mu)\left[u^2 - u_x w_x - u_z w_z\right]\right\}^{-1} \tag{9}$$

One can get a time-varying learning gain for convenient application in learning updating. Note that $u_x \gg w_x$, $u_z \gg w_z$, and $V \approx u$. Thus, based on Eq. (9), the first choice of $K(t)$ is

$$K^{(1)}(t) = -\frac{2m}{\rho_0 s u_m^2(t)} \tag{10}$$

where ρ_0 is the standard air density at sea level and the superscript (1) is the first design of learning gain. $K^{(1)}(t)$ is simple to use as it varies only with respect to time t. In fact, one may consider that $K(t)$ is not only a function of time t but also a function of iteration number k. This is essentially because, in Eq. (9), ρ and V vary from iteration to iteration. Thus, we get the second formula of learning gain as follows:

$$\left[K^{(2)}(t)\right]_k = -\frac{2m}{(\rho)_k s[u^2(t)]_k} \tag{11}$$

Intuitively, one may wish to replace $[u^2(t)]_k$ with the "desired" one, $u_m^2(t)$, which gives the third learning gain formula

$$\left[K^{(3)}(t)\right]_k = -\frac{2m}{(\rho)_k s u_m^2(t)} \tag{12}$$

In the next section, the extraction results associated with the three varying learning gains are given and discussed. As shown in the data reduction practice, it is not possible to converge in one step due to numerical errors, although theoretically it is possible to make $\bar{\rho}$ zero.

Extracted Results from Flight Tests

The main purpose of the flight tests is to finally measure the aerodynamic drag coefficient curve of the aerobomb. Several tests were carried out under different bombing conditions. In this section, we present the curve extraction results from three-dimensional theodolite film data sets of three bombing flight paths supplied by a proving ground (see Fig. 1 for the measured trajectories). From Fig. 1, the numerically computed velocities are to be used to extract the C_{df} curve. According to the standardization processing result from the proving ground, for flights 7–9, respectively, the elevation angles (deg) when bombing are 0.15, 0.11, and −0.04; the heights (meters) when bombing are 4101, 4103, and 4106; the velocities (meters per second) when bombing are 127, 129, and 125; the obtained ballistic coefficients c are 0.60553, 0.61213, and 0.60274; and the obtained shape coefficients i_{cp1} are 0.757, 0.765, and 0.753. In the preceding results, c is obtained as a fitting factor to fit the range measurements under practical bombing conditions:

$$c = (id^2/m) \times 10^3, \qquad i = (c_x/\bar{c}_{x0}) \tag{13}$$

where i is the aerobomb's shape coefficient, c_x is the drag coefficient, and $\bar{c}_{x0}$ is the so-called drag-law. Note that i_{cp1} is also a fitting result provided by the proving ground based on a fitted c.

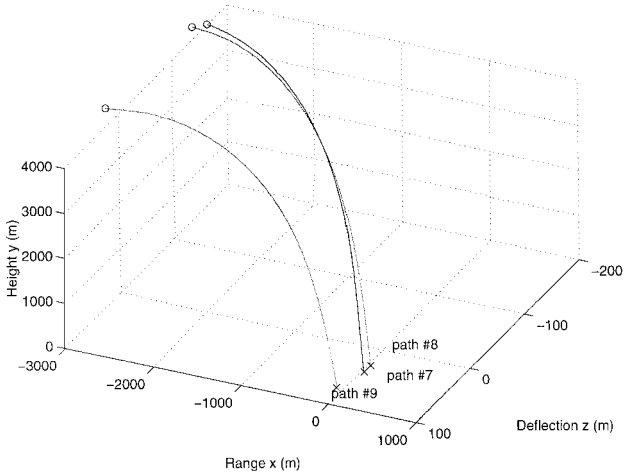

Fig. 1 Measured flight paths 7–9.

Table 1 Comparison of shape coefficients

Flight path no.	i_{cp2}	Shape coefficient error: $(i_{cp1} - i_{cp2})/i_{cp1}$,%	Range error, % $\Delta A/A$
7	0.787	−3.991	−0.598
8	0.782	−2.253	−0.338
9	0.819	−8.715	−1.307
Average value	0.796	−4.986	−0.748

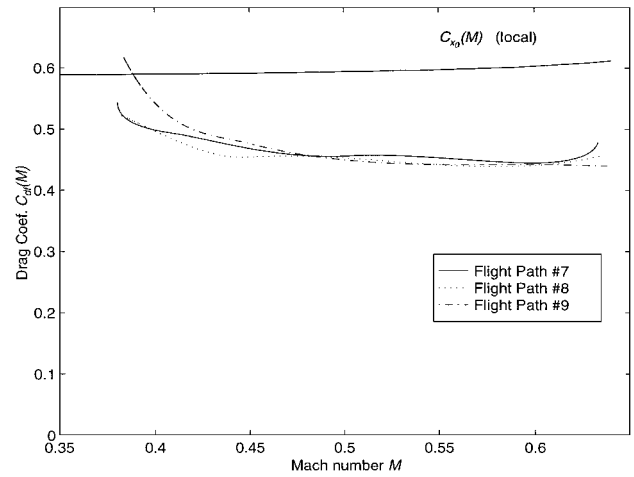

Fig. 2 Extracted $C_{df}(M)$ by iterative learning.

Using the method and relevant data reduction program of this work, three $C_{df}(M)$ curves corresponding to flight paths 7–9 have been obtained. The results are shown in Fig. 2, where the drag law $\bar{c}_{x0}(M)$ (local) is also plotted for comparison. For each flight path, the final converged results are the same using different choices of learning gain.

How to validate the extracted results is quite difficult because the effect of airflow interference interaction between aircraft and aerobomb is not yet clear. From Fig. 2, the shape coefficient curve $i(M)$ of each flight path can be determined by

$$i(M) = \frac{C_{df}(M)}{\bar{c}_{x0}(M)} \tag{14}$$

Taking the mean value of this $i(M)$, one can get the average shape coefficient i_{cp2}. Note that i_{cp2} obtained is with the comparability with respect to i_{cp1} just given. See Table 1 for the shape coefficient comparison and the induced range error comparison according to an evaluation formula of the proving ground. From Fig. 2 and Table 1, it can be validated that the $C_{df}(M)$ curve extracted is correct. The curve form of $C_{df}(M)$ fits to the qualitative theoretical conjecture. In these sets of flight testing data, the maximal Mach number is about 0.65. Theoretically, $C_{df}(M)$ should be constant when $M \leq 0.65$ (lower subsonic region). But due to the interference airflow between the aircraft and the aerobomb when bombing and the relevant induced angular motion around the center of mass of the aerobomb, the shape of $C_{df}(M)$ is reasonable. It is interesting to note that $C_{df}(M)$ is approximately in an exponential decaying form in the earlier stage of Mach range and then smoothly transits to a constant value. The latter phenomenon fits to the aerodynamic theory as effects of interference flowfield to the aerobomb's motion have died out and the aerobomb is almost in a free air flowfield.

To verify the robustness of the iterative learning method with respect to initial control guesses, a constant learning gain scheme is used. According to the preceding analysis, the gain $K(t)$ is chosen as −0.1. Under the same learning gain, three different initial guesses $[C_{df}(t)]_0$, i.e., +0.3, 0, and −0.3, respectively, are considered. The learning iteration histories are summarized in Fig. 3. Under the same exit condition ($\varepsilon^* = 0.05$ m/s), the required numbers of learning iterations are 9, 11, and 18, respectively. From aerodynamics, $[C_{df}(t)]_0 = +0.3$ is a better guess, whereas $[C_{df}(t)]_0 = -0.3$ is impractical. One may observe that the iterative learning method still works with such wrong initial guess, whereas other methods, such

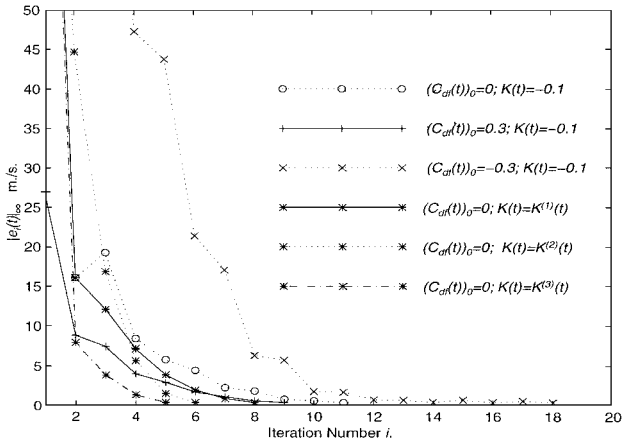

Fig. 3 Learning iteration histories for different learning gain selection schemes.

as optimal dynamic fitting proposed by Chen et al.,[3,4] do not work even when $[C_{df}(t)]_0 = 0$.

Three proposed schemes for learning gain determination have been applied. Under the same initial guess $[C_{df}(t)]_0 = 0$ and exit condition ($\varepsilon^* = 0.05$ m/s), extraction schemes with $K^{(1)}(t)$, $K^{(2)}(t)$, and $K^{(3)}(t)$ take eight, six, and five learning iterations, respectively. It can be seen from Fig. 3 that $K^{(3)}(t)$ produces the fastest learning convergence. Moreover, from Fig. 3, one can clearly observe that the convergence properties for learning gains varying with respect to time t as well as iteration number k are better than those for the constant learning gains.

Concluding Remarks

The iterative learning method is successfully applied to an optimal tracking control problem that is related to aerodynamic property curve extraction from flight testing data. The convergence is shown to be robust to the initial control guess. Several choices of learning gains for a better convergence performance have been discussed. Results from actual flight testing data for three bombing flight paths have been presented to validate the effectiveness of the proposed iterative learning scheme.

References

[1]Illif, K. W., "Parameter Estimation for Flight Vehicle," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 5, 1989, pp. 609–622.

[2]Linse, D. J., and Stengel, R. F., "Identification of Aerodynamic Coefficients Using Computational Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 6, 1994, pp. 1018–1025.

[3]Chen, Y., Lu, D., Dou, H., and Qing, Y., "Optimal Dynamic Fitting and Identification of Aerobomb's Fitting Drag Coefficient Curve," *Proceedings of the First IEEE Conference on Control Applications*, Inst. of Electrical and Electronics Engineers, New York, 1992, pp. 853–858.

[4]Chen, Y., Wen, C., Gong, Z., and Sun, M., "Drag Coefficient Curve Identification of Projectiles from Flight Tests via Optimal Dynamic Fitting," *Journal of Control Engineering Practice*, Vol. 5, No. 5, 1997, pp. 627–636.

[5]Arimoto, S., Kawamura, S., and Miyazaki, F., "Bettering Operation of Robots by Learning," *Journal of Robotic Systems*, Vol. 1, No. 2, 1984, pp. 123–140.

[6]Chen, Y., Sun, M., Huang, B., and Dou, H., "Robust Higher Order Repetitive Learning Control Algorithm for Tracking Control of Delayed Repetitive Systems," *Proceedings of the 31st IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, New York, 1992, pp. 2504–2510.

J. A. Martin
Associate Editor