[6]Dudewicz, E. J., and Mishra, S. N., *Modern Mathematical Statistics*, Wiley, New York, 1988, p. 284.

[7]*@RISK, Risk Analysis and Modeling*, Palisade Corp., Newfield, NY, 1992.

J. A. Martin
Associate Editor

Efficient Jitter Analysis for Spacecraft

Peiman G. Maghami*
NASA Langley Research Center,
Hampton, Virginia 23681-0001

Introduction

TYPICALLY in space missions, the science instruments require a specific degree of pointing accuracy as well as dynamical quietness. This dynamical quietness, which is needed to allow the instruments to make measurements (remote sensing applications) or to perform other functions, is usually characterized in terms of jitter and stability specifications.[1,2] To ensure that the spacecraft meets the requirements of its instruments, several jitter analyses are performed throughout the design phase of the spacecraft and beyond as the models of the spacecraft, its components, and disturbances mature. Each such analysis involves the simulation of the spacecraft and instrument dynamical response to all known disturbance scenarios, followed by the computation of jitter values for each instrument based on the specified jitter time windows.[1,2] The direct approach for computing jitter values by sweeping maxima and minima throughout the time history may be costly in the computational sense as the size and number of the time histories involved could be quite large. Keeping in mind that typical spacecraft simulation time histories may easily involve hundreds of thousands or millions of points, it is imperative that a jitter analysis algorithm that is more efficient than the direct approach be developed. This Note presents a vectorized algorithm for efficient computation of spacecraft jitter values. 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 deals with the multiple jitter windows sequentially, first computing jitter values for the smallest time window and then looping over the remaining time windows until all jitter values are computed. A numerical example is carried out to demonstrate the efficiency and feasibility of the proposed jitter analysis technique.

Problem Formulation

Let $\boldsymbol{y}(t)$ represent the time history of the spacecraft response at a specified location on the spacecraft due to some disturbances. Assume $\boldsymbol{y}$ has n elements corresponding to equally stepped time values with a time increment of T. Furthermore, assume that jitter values are desired for several jitter time windows represented by the elements of $\boldsymbol{t}_j$, where $\boldsymbol{t}_j(i) < \boldsymbol{t}_j(i+1)$, $i = 1, 2, \ldots, m$, with m denoting the number of jitter time windows in t_j.

The jitter value for the time window $\boldsymbol{t}_j(i)$ is defined as maximum peak-to-peak excursions of the spacecraft output response, within a time window of size $\boldsymbol{t}_j(i)$ seconds, over all possible positions of such a window in the response time history. Reference 3 provides a complete definition of jitter.

Jitter value is a measure of the level of spacecraft motion in on-orbit conditions, as well as, to some degree, the frequency content of such motion, depending on the size of the jitter window. The direct approach for computing jitter would involve performing $n-k+1$ maximum and minimum operations on vectors of size k, where k, which denotes the number of points in a given jitter window, is given by

$$k = fix\{\boldsymbol{t}_j(i)/T\} + 1 \tag{1}$$

The direct approach may be computationally acceptable if the jitter window is very small (k is near 1) or very large (k is near n). However, it is quite inefficient for most realistic problems where the size of the jitter window is not at either extreme and the size of the time history can be in the hundreds of thousands or millions. An efficient algorithm for the computation of jitter/stability has been developed. This algorithm is based on vector operations to achieve a drastic speedup of the computational time over the direct approach. The algorithm starts with some preprocessing of the time history data and then loops through the requested jitter windows, starting from the smallest. The algorithm is described in the following sections. Note that most of the calculations in the algorithm can be carried out by vector arithmetic operations that can be executed more rapidly in an array processing language such as MATLAB[4] or on a vector or parallel processing computer than by executing a loop performing scalar calculations.

Step 1: Identification of Extrema

In the first step, the extrema in the entire time history are identified. These include maxima, minima, and level response points. The motivation behind this is that, for any given jitter window position along the time history, the jitter value is dominated by either extrema (maxima, minima, or level response points) in the window, if they are present, and/or by the response at the endpoints of the window. To identify the extrema in the time history, compute the first difference vector of the time history array y:

$$\boldsymbol{y}_1 = \boldsymbol{y}(2:n) - \boldsymbol{y}(1:n-1) \tag{2}$$

where $\boldsymbol{y}(2:n)$, for example, defines a vector formed from the elements 2 through n of y. Now, compute the sign difference vector for the first difference vector $\boldsymbol{y}_1$:

$$\boldsymbol{s}_1 = \mathrm{sgn}[\boldsymbol{y}_1(2:n-1)] - \mathrm{sgn}[\boldsymbol{y}_1(1:n-2)] \tag{3}$$

The locations associated with the internal maxima may be determined as

$$\boldsymbol{l}_{\max} = \{i+1 \mid \boldsymbol{s}_1(i) = -1 \text{ or } -2\} \tag{4}$$

Here, ones are added to the indices to identify points in the original time history and not the sign difference history. Similarly, the locations associated with the minima may be determined as

$$\boldsymbol{l}_{\min} = \{i+1 \mid \boldsymbol{s}_1(i) = 1 \text{ or } 2\} \tag{5}$$

Let $\boldsymbol{y}_{\max}$ and $\boldsymbol{y}_{\min}$ represent the identified maxima and minima in the time history, corresponding to locations in vectors $\boldsymbol{l}_{\max}$ and $\boldsymbol{l}_{\min}$, respectively. It is assumed that the elements of vectors $\boldsymbol{l}_{\max}$ and $\boldsymbol{l}_{\min}$ are in ascending order. This comprises the necessary preprocessing of the time history data that must be performed at the beginning of the procedure. The remaining steps in the procedure are applied sequentially for each jitter time window, starting from the smallest.

Step 2: Reduction of Extrema

Not all extrema contribute to the computation of jitter value for a given time window. For example, a maximum that is surrounded by larger maxima on the left and the right would not contribute to the jitter value if the distance, in time, between the surrounding maxima is not greater than the given time window. A similar argument also applies to the minima, with the exception that the surrounding minima should be smaller.

Received July 15, 1997; revision received Nov. 28, 1997; accepted for publication Jan. 5, 1998.

*Senior Research Engineer, Guidance and Control Branch, Mail Stop 161. Senior Member AIAA.

The insignificant maxima are removed from the set of maxima through a sequential procedure. At each iteration, first, those maxima that are surrounded on the left and right by maxima that are within k time points of each other are identified. Here, k is the number of data points in the time window defined in Eq. (1). The locations of these maxima are obtained from

$$\boldsymbol{p} = \{i \mid \boldsymbol{l}_{\max}(i+1) - \boldsymbol{l}_{\max}(i-1) < k\} \tag{6}$$

Then, those maxima in $\boldsymbol{y}_{\max}$ indexed by elements of $\boldsymbol{p}$ that are smaller than their immediate surrounding maxima are deemed insignificant to jitter computations and removed from the set of maxima. The locations of those maxima are identified as

$$\boldsymbol{l}_{\text{ns}} = \{i \in \boldsymbol{p} \mid \boldsymbol{y}_{\max}(i+1) - \boldsymbol{y}_{\max}(i) \geq 0 \text{ and } \boldsymbol{y}_{\max}(i-1) - \boldsymbol{y}_{\max}(i) \geq 0\} \tag{7}$$

These insignificant maxima are then removed from the current set of significant maxima, represented by the location array $\boldsymbol{l}_{\max}$ and the value array $\boldsymbol{y}_{\max}$, and the current set is updated:

$$\boldsymbol{l}_{\max} \leftarrow \{i \in \boldsymbol{l}_{\max} \mid i \notin \boldsymbol{l}_{\text{ns}}\} \tag{8}$$

$$\boldsymbol{y}_{\max} \leftarrow \boldsymbol{y}_{\max}(\boldsymbol{l}_{\max}) \tag{9}$$

This sequential procedure is continued until either all insignificant maxima are removed or the estimated processing time to remove additional maxima becomes larger than the estimated computational savings realized from removing additional insignificant maxima. One possible termination criterion may be to stop the process when the percent reduction in the size of the location array $\boldsymbol{l}_{\max}$ drops below a user-defined threshold value. The set of minima may be similarly reduced, with the exception that instead of Eq. (7) one has

$$\boldsymbol{l}_{\text{ns}} = \{i \in \boldsymbol{p} \mid \boldsymbol{y}_{\min}(i+1) - \boldsymbol{y}_{\min}(i) \leq 0 \text{ and } \boldsymbol{y}_{\min}(i-1) - \boldsymbol{y}_{\min}(i) \leq 0\} \tag{10}$$

Note that the location and value arrays associated with the minima are denoted by $\boldsymbol{l}_{\min}$ and $\boldsymbol{y}_{\min}$.

Step 3: Span of Influence of Extrema

At this point, the span of influence of the significant extrema points, insofar as affecting jitter values, is evaluated. The following procedure is used for the maxima.

1) Compute the first difference vector $\boldsymbol{y}_d$:

$$\boldsymbol{y}_d = \boldsymbol{y}_{\max}(2 : n_{\max}) - \boldsymbol{y}_{\max}(1 : n_{\max} - 1) \tag{11}$$

where $n_{\max}$ denotes the size of the vector $\boldsymbol{y}_{\max}$.

2) Divide the maxima into two groups, those that are smaller than or equal to their preceding maximum and those that are not. The location arrays for these groups are given by

$$\boldsymbol{i}_l = \{i+1 \mid \boldsymbol{y}_d(i) \leq 0\} \tag{12}$$

$$\boldsymbol{i}_u = \{i+1 \mid \boldsymbol{y}_d(i) > 0\} \tag{13}$$

3) Initialize a vector representing the span of influence of the maxima on the right from the first time step to step $n-k+1$:

$$\boldsymbol{s}_u = \min\{\boldsymbol{l}_{\max},\ n-k+1\} \tag{14}$$

Here, the min or max operators are element-by-element operators, such that their output would be a vector if one or both of their arguments are vectors.

4) Initialize a vector representing the span of influence of the maxima on the left from the first time step to step $n-k+1$:

$$\boldsymbol{s}_l = \max\{\boldsymbol{l}_{\max} - (k-1),\ 1\} \tag{15}$$

5) To refine the span of influence of the maxima, two situations are considered.

a) If a maximum is larger than its preceding maximum, then it may affect the span of influence, on the right, of the preceding maximum, depending on the vicinity of the maximum and the size of the jitter window. This possible influence is incorporated by adjusting the array $\boldsymbol{s}_u$ as

$$\boldsymbol{s}_u(\boldsymbol{i}_u - \hat{\boldsymbol{I}}) \leftarrow \min\{\max[\boldsymbol{l}_{\max}(\boldsymbol{i}_u) - k,\ 1],\ \boldsymbol{s}_u(\boldsymbol{i}_u - \hat{\boldsymbol{I}})\} \tag{16}$$

where $\hat{\boldsymbol{I}}$ is a vector of ones with appropriate dimensions.

b) If a maximum is smaller than or equal to its preceding maximum, then its span of influence, on the left, may be affected by the preceding maximum, depending on the vicinity of the maximum and the size of the jitter window. This possible influence is incorporated by adjusting the array $\boldsymbol{s}_l$ as

$$\boldsymbol{s}_l(\boldsymbol{i}_l) \leftarrow \max\{\min[\boldsymbol{l}_{\max}(\boldsymbol{i}_l - \hat{\boldsymbol{I}}) + \hat{\boldsymbol{I}},\ n-k+1],\ \boldsymbol{s}_l(\boldsymbol{i}_l)\} \tag{17}$$

6) Sort the maxima in an ascending order. Let $\bar{\boldsymbol{y}}_{\max}$, $\bar{\boldsymbol{s}}_l$, and $\bar{\boldsymbol{s}}_u$ represent the sorted vectors.

7) Initialize $\boldsymbol{y}_u = \max\{\boldsymbol{y}(1 : n-k+1),\ \boldsymbol{y}(k : n)\}$.

8) Loop through the number of maxima, each time storing the span of influence of each maximum in appropriate elements of vector $\boldsymbol{y}_u$ as follows:

$$\boldsymbol{y}_u(j) = \bar{\boldsymbol{y}}_{\max}(i), \qquad \bar{\boldsymbol{s}}_l(i) \leq j \leq \bar{\boldsymbol{s}}_u(i), \qquad i = 1, 2, \ldots, n_{\max} \tag{18}$$

This operation was performed after sorting so that, if spans of influence overlapped, the last element written into an overlapped location of $\boldsymbol{y}_u$ would be the dominant one.

A similar procedure is followed for the minima.

1) Compute the first difference vector $\boldsymbol{y}_d$ of the current vector of minima $\boldsymbol{y}_{\min}$ using Eq. (11), with $\boldsymbol{y}_{\min}$ replacing $\boldsymbol{y}_{\max}$.

2) Then divide the minima into two groups, those that are greater than or equal to their preceding minimum and those that are not. The location arrays for these groups are given by

$$\boldsymbol{i}_l = \{i+1 \mid \boldsymbol{y}_d(i) \geq 0\} \tag{19}$$

$$\boldsymbol{i}_u = \{i+1 \mid \boldsymbol{y}_d(i) < 0\} \tag{20}$$

3) Follow steps 3–5 described for the maxima, except that all occurrences of the vector $\boldsymbol{l}_{\max}$ are replaced with the vector $\boldsymbol{l}_{\min}$.

4) Sort the minima in a descending order. Let $\bar{\boldsymbol{y}}_{\min}$, $\bar{\boldsymbol{s}}_l$, and $\bar{\boldsymbol{s}}_u$ represent the sorted vectors.

5) Initialize $\boldsymbol{y}_l = \min\{\boldsymbol{y}(1 : n-k+1),\ \boldsymbol{y}(k : n)\}$.

6) Loop through the number of minima, each time storing the span of influence of each minimum in appropriate elements of vector $\boldsymbol{y}_l$ as follows:

$$\boldsymbol{y}_l(j) = \bar{\boldsymbol{y}}_{\min}(i), \qquad \bar{\boldsymbol{s}}_l(i) \leq j \leq \bar{\boldsymbol{s}}_u(i), \qquad i = 1, 2, \ldots, n_{\min} \tag{21}$$

Step 4: Computation of Jitter Value

As mentioned earlier, the jitter value for any given jitter window position along the time history is dominated by either extrema (maxima, minima, or level response points) in the window, if they are present, and/or by the response at the endpoints of the window. Now, with the span of influence of the maxima (minima) stored in the vector $\boldsymbol{y}_u$ ($\boldsymbol{y}_l$), the jitter value for the jitter time window $\boldsymbol{t}_j(i)$ may be computed as follows:

$$\boldsymbol{y}_u \leftarrow \max\{\boldsymbol{y}_u, \max[\boldsymbol{y}(1 : n-k+1], \boldsymbol{y}(k : n)]\} \tag{22}$$

and

$$\boldsymbol{y}_l \leftarrow \min\{\boldsymbol{y}_l, \min[\boldsymbol{y}(1 : n-k+1),\ \boldsymbol{y}(k : n)]\} \tag{23}$$

Now, jitter value for the first window is simply computed as

$$jtr = \max\{|\boldsymbol{y}_u - \boldsymbol{y}_l|\} \tag{24}$$

If there is more than one jitter window, the procedure begins with the smallest jitter window and computes the corresponding jitter values using the procedure outlined in the preceding sections. Then, jitter values for the next largest jitter window are calculated, starting with step 2 (reduction of extrema) of the procedure outlined, along

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

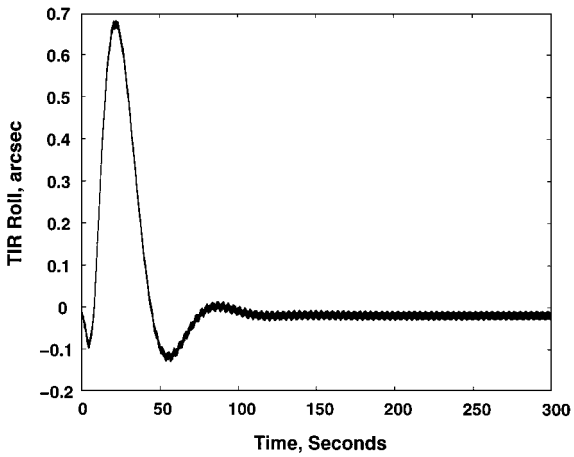

Fig. 1 Time response for Measurements of Pollution in the Tropespheric (MOPITT) instrument cryocooler disturbance sequence.

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.