Efficient Jitter Analysis for Spacecraft

Peiman G. Maghami

NASA Langley Research Center
Mail Stop 161
Hampton, Virginia 23681–0001

Efficient Jitter Analysis for Spacecraft

Peiman G. Maghami

NASA Langley Research Center
Mail Stop 161
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 perform other functions, is usually characterized in terms of jitter and stability specifications[1–2]. In order to insure 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 be developed which is more efficient than the direct approach. This paper 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, 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 $y(t)$ represent the time history of the spacecraft response at a specified location on the spacecraft due to some disturbances. Assume 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 t_j, where $t_j(i) < t_j(i+1)$, $i = 1, 2, \ldots, m$, with m denoting the number of jitter time windows in t_j.

Definition: The jitter value for the time window $t_j(i)$ is defined as maximum peak to peak excursions of the spacecraft output response, within a time window of size $t_j(i)$ seconds, over all possible positions of such 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

$$k = fix\{t_j(i)/T\} + 1 \tag{1}$$

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 in order to

achieve a drastic speed up 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. It should be noted that most of the calculations in the algorithm can be carried out by vector arithmetic operations which 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 end points of the window. To identify the extrema in the time history, compute the first difference vector of the time history array y

$$y_1 = y(2:n) - y(1:n-1) \tag{2}$$

where $y(2:n)$, 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 y_1:

$$s_1 = sgn(y_1(2:n-1)) - sgn(y_1(1:n-2)) \tag{3}$$

The locations associated with the internal maxima may be determined as

$$l_{max} = \{i+1 | s_1(i) = -1 \; or \; -2\} \tag{4}$$

Here, ones are added to the indices in order 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

$$l_{min} = \{i+1 | s_1(i) = 1 \; or \; 2\} \tag{5}$$

Let y_{max} and y_{min} represent the identified maxima and minima in the time history, corresponding to locations in vectors l_{max} and l_{min}, respectively. It is assumed that the elements of vectors l_{max} and l_{min} are in ascending order. This comprises the necessary preprocessing of the time history data which 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. Similar argument also applies to the minima, with the exception that the surrounding minima should be smaller.

The insignificant maxima are removed from the set of maxima through a sequential procedure. At each iteration, first, those maxima which 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 location of these maxima are obtained from

$$p = \{i | l_{max}(i+1) - l_{max}(i-1) < k\} \tag{6}$$

Then, those maxima in y_{max} indexed by elements of p which are smaller than their immediate surrounding maxima are deemed as insignificant to jitter computations and removed from the set of maxima. The location of those maxima are identified as

$$l_{ns} = \{i \in p | y_{max}(i+1) - y_{max}(i) \geq 0 \; \& y_{max}(i-1) - y_{max}(i) \geq 0 \; \} \tag{7}$$

These insignificant maxima are then removed from the current set of significant maxima, represented by the location array l_{max} and the value array y_{max}, and the current set is updated:

$$l_{max} \leftarrow \{i \in l_{max} | i \notin l_{ns}\} \tag{8}$$

$$y_{max} \leftarrow y_{max}(l_{max}) \tag{9}$$

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 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

$$l_{ns} = \{i \in p | y_{min}(i+1) - y_{min}(i) \leq 0 \; \& y_{min}(i-1) - y_{min}(i) \leq 0 \; \} \tag{10}$$

Note that the location and value arrays associated with the minima are denoted by l_{min} and y_{min}.

Step 3: Span of Influence of Extrema

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

a. Compute the first difference vector, y_d:

$$y_d = y_{max}(2 : n_{max}) - y_{max}(1 : n_{max} - 1) \tag{11}$$

where n_{max} denotes the size of the vector y_{max}.

b. Divide the maxima in 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

$$i_l = \{i + 1 | y_d(i) \leq 0\} \tag{12}$$

$$i_u = \{i + 1 | y_d(i) > 0\} \tag{13}$$

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

$$s_u = min\{l_{max}, n - k + 1\} \tag{14}$$

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.

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

$$s_l = max\{l_{max} - (k - 1), 1\} \tag{15}$$

e. In order to refine the span of influence of the maxima, two situations are considered:

i. If a maximum is larger than its preceding maximum, then it may effect 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 s_u as

$$s_u\left(i_u - \hat{I}\right) \leftarrow min\left\{max\{l_{max}(i_u) - k, 1\}, s_u\left(i_u - \hat{I}\right)\right\} \tag{16}$$

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

ii. 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 s_l as

$$s_l(i_l) \leftarrow max\left\{min\left\{l_{max}\left(i_l - \hat{I}\right) + \hat{I}, n - k + 1\right\}, s_l(i_l)\right\} \tag{17}$$

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

g. Initialize $y_u = max\{y(1:n-k+1), y(k:n)\}$.

h. Loop through the number of maxima, each time storing the span of influence of each maximum in appropriate elements of vector y_u as follows:

$$y_u(j) = \bar{y}_{max}(i) \;\; ; \;\; \bar{s}_l(i) \leq j \leq \bar{s}_u(i) \, , \; i = 1, 2, \ldots, n_{max}$$

This operation was performed after sorting so that, if spans of influence overlapped, the last element written into an overlapped location of y_u would be the dominant one.

A similar procedure is followed for the minima:

1. Compute the first difference vector, y_d, of the current vector of minima y_{min} using Eq. (11), with y_{min} replacing y_{max}).
2. Then, divide the minima in two groups, those that are greater than or equal to their proceeding minimum and those that are not. The location arrays for these groups are given by

$$i_l = \{i+1 | y_d(i) \geq 0\} \tag{19}$$

$$i_u = \{i+1 | y_d(i) < 0\} \tag{20}$$

3. Follow Steps c–e described for the maxima except that all occurrences of the vector l_{max} is replaced with the vector l_{min}.
4. Sort the minima in a descending order. Let $\bar{y}_{min}$, $\bar{s}_l$ and $\bar{s}_u$ represent the sorted vectors.
5. Initialize $y_l = min\{y(1:n-k+1), y(k:n)\}$.
6. Loop through the number of minima, each time storing the span of influence of each minimum in appropriate elements of vector y_l as follows:

$$y_l(j) = \bar{y}_{min}(i) \;\; ; \;\; \bar{s}_l(i) \leq j \leq \bar{s}_u(i) \, , \; i = 1, 2, \ldots, n_{min}$$

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 end points of the window. Now, with the span of influence of the maxima (minima) stored in the vector y_u (y_l), the jitter value for the jitter time window $t_j(i)$ may be computed as follows

$$y_u \leftarrow max\{y_u, max\{y(1:n-k+1), y(k:n)\}\} \tag{22}$$

and

$$y_l \leftarrow min\{y_l, min\{y(1:n-k+1), y(k:n)\}\} \tag{23}$$

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

$$jtr = max\{|y_u - y_l|\} \tag{24}$$

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 previous sections. Then, jitter values for the next largest jitter window are calculated, starting with the step 2 (reduction of extrema) of the procedure outlined and the vectors y_{max}, y_{min}, l_{max}, and 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

In order 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 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 second time history of the roll response of ASTER/TIR is presented in fig. 1. With the sampling period at 0.001 seconds, the roll response resulted in a 300,001 element output vector (corresponding to 300,000 time steps). Jitter values were computed for 3 windows at 0.1 sec, 1 sec and 10 sec, using the proposed vectorized approach and the direct approach. Both techniques provided jitter values of 0.02 arcsec, 0.09 arcsec, and 0.63 arcsec, corresponding to the 0.1 sec, 1 sec, and 10 sec, windows, respectively. The formulations were implemented in the MATLAB [5] 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 the table. It is noted that the times reported for each jitter window is 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.

Table 1

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

The feasibility and efficiency of the proposed vectorized algorithm is clearly observed from the table. 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 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 the proposed algorithm reduces the required

computational time by several orders of magnitude, thereby demonstrating the feasibility of the approach,

References

1. GE Aerospace; EOS-A Pointing and Orbit Requirements (AFM T-9), NAS5–32500, August 28, 1991.
2. GE Aerospace; EOS-A Pointing and Orbit Study Update (AFM T-9), NAS5–32500, May 22, 1992.
3. Giesy, D. P., “Efficient Calculation of a Jitter/Stability Metric,” Journal of Spacecraft and Rockets, Vol. 34, No. 4, July-August, 1997.
4. MATLAB Reference Guide-High-Performance Numeric Computation and Visualization Software. The MathWorks, Inc., 1993.
5. Asrar, Ghassem; and Dokken, David Jon, eds.: 1993 Earth Observing System Reference Handbook. NASA NP-202, 1993.

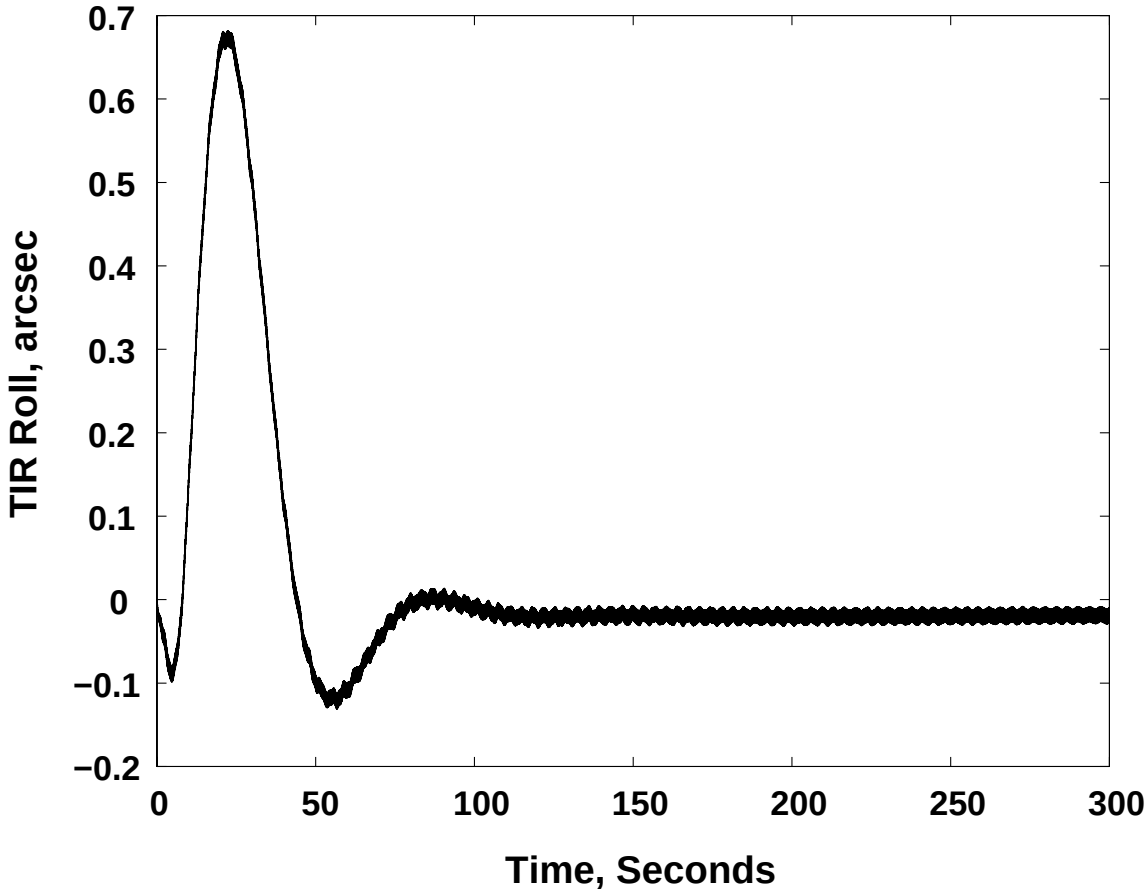
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
TIR Roll, arcsec
0
50
100
150
200
250
300
Time, Seconds

Figure 1 Time Response for MOPITT Cryocooler Disturbance Sequence