NASA CONTRACTOR REPORT 181942

FATIGUE LOADING HISTORY RECONSTRUCTION BASED ON THE RAIN-FLOW TECHNIQUE

A. K. Khosrovaneh and
N. E. Dowling

VIRGINIA POLYTECHNIC INSTITUTE
AND STATE UNIVERSITY
Blacksburg, Virginia

Grant NAG1-822
December 1989

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

(NASA-CR-181942) FATIGUE LOADING HISTORY RECONSTRUCTION BASED ON THE RAIN-FLOW TECHNIQUE (Virginia Polytechnic Inst. and State Univ.) 73 p CSCL 20K

N90-15482

Unclas
G3/39 0257095

Fatigue Loading History Reconstruction Based on the Rain-Flow Technique

A.K. Khosrovaneh

Graduate Research Assistant

N.E. Dowling

Professor

Engineering Science and Mechanics Department

Virginia Polytechnic Institute and State University

Blacksburg, VA 24061

Abstract

Methods are considered of reducing a non-random fatigue loading history to a concise description and then of reconstructing a time history similar to the original. In particular, three methods of reconstruction based on a rain-flow cycle counting matrix are presented. A rain-flow matrix consist of the numbers of cycles at various peak and valley combinations. Two methods are based on a two dimensional rain-flow matrix, and the third on a three dimensional rain-flow matrix. Histories reconstructed by any of these methods produce a rain-flow matrix identical to that of the original history, and as a result the resulting time history is expected to produce a fatigue life similar to that for the original. The procedures described allow lengthy loading histories to be stored in compact form.

Introduction

The study of lengthy irregular time histories is of interest for analysis of fatigue damage. In order to make the recorded data manageable for future use, it is desirable to summarize a lengthy irregular loading history by a concise description. This concise description is needed to provide sufficient information to estimate fatigue life and also to reconstruct a loading history similar to the original one, in the form of a time sequence, which can then be used in component testing. The goal of this paper is to use loading histories described in concise form to reconstruct histories that produce fatigue lives similar to those for the original history.

(a) Reconstruction Methods

A variety of methods exist for reconstruction purposes, such as power spectral density (PSD) [1], the to-from matrix [2-4], and cycle counting [5-11]. The first two of these have the advantage of having a probabilistic basis, but they have a disadvantage in handling loads with a deterministic mean variation. Also, these methods may not produce the same life as the original history if it is not a random process. For example, Refs. [4,11] report that to-from reconstructed histories may produce lives which are excessively conservative.

Reconstruction based on cycle counting can be done using a variety of cycle counting methods, such as level crossing [6,10,12,13], peak-valley [6,9,14,15], or the rain-flow method [7-9]. In the past, the most popular reconstruction method was the programmed step test, which is based on a level crossing histogram. In the simplest version of a level crossing method [16], all positive slope level crossings above a reference (mean) load, and all negative slope level crossing below the reference load are counted. Figure 1(a) shows the results of a level crossing count. A loading history is reconstructed from the level crossing count by first constructing the largest possible cycle, using the highest level and lowest level, followed by the second largest cycle, and so on, until all level crossings in the histogram are used. Figure 1(b) shows this method. However, forming the largest possible cycles first is the most conservative choice, and other procedures may be used. Literature survey reveals that this method gives lives that often vary considerably from those for the original history. In fact, lives can be either

conservative or non-conservative depending on the details of the loading history and the exact method of reconstruction.

Another method of reconstruction based on cycle counting is the peak-valley reconstruction method. In the simplest version of the peak-valley cycle counting method, all peaks above the reference load, and all valleys below the reference load, are counted. Figure 2(a) shows this method. A reconstructed history can be obtained by first combining the highest peak and lowest valley to form a cycle, and the second highest and second lowest, and so on, until all events are used. Figure 2(b) shows this method. Again, a different and less conservative procedure could be adopted, such as randomly chosen pairings of peaks and valleys to form cycles. In our previous work [9,14], the peak-valley reconstruction method in the version of Fig. 2 was used for the reconstruction of two helicopter load histories. It was found that this method produced histories which could have excessively conservative calculated fatigue lives. Therefore, a need was identified to use a method of reconstruction that produced histories which had expected fatigue lives similar to those for the original history.

A third method of reconstruction based on cycle counting is to use rain-flow cycle counting. This more complex cycle counting method is illustrated in Figs. 3 and 4 and will be described in some detail below. It has been found in all of a limited number of cases studied [7-9,11] that reconstruction based on this method gives similar lives to those for the original history, so that this is a promising reconstruction method. It's success appears to be due to the reconstructed history producing the same rain-flow cycles as the original history.

The concise description that is proposed above is obtained by applying the rain-flow cycle counting method to the loading history and recording the result in a matrix which gives numbers of rain-flow cycles at various combinations of peak and valley loads. Although some detail is lost, such a matrix can be used with the aid of the local strain approach to place upper and lower bounds on the fatigue crack initiation life that would result from analysis of the original, unsummarized history. (The procedure is described in Ref. [14]). As such bounds are reasonably tight for most loading histories of practical interest, rain-flow cycle counting appears to preserve sufficient information from the sequence of loads so that the crack initiation

life depends mainly on the peak-valley matrix of rain-flow cycles. In a noteworthy paper, Perrett [8] experimentally studied the success of rain-flow reconstructions and considered both crack initiation and crack growth. Based on the available evidence, reordered loading histories appear to all have approximately the same life for both crack initiation and crack growth, as long as the rain-flow cycle count of the original is reproduced. Thus, rain-flow cycle counting has definite advantages as a means of reconstructing a fatigue-equivalent loading history from a concise description which involves much less data storage than the entire original history.

(b) Rain-Flow Cycle Counting

In the rain-flow counting method, cycles are counted depending on the comparison of two adjacent ranges as illustrated in Fig. 3, which also defines the range and mean of a cycle. If the first range is less than or equal to the second, a cycle is counted and the corresponding peak and valley are discarded for purposes of further cycle counting. This procedure continues until all the peaks and valleys in the history are considered. Figure 4 illustrates this process for a simple loading history, and the result is given in Table 1. Based on Table 1, a history can be reconstructed which gives the same rain-flow matrix as the original history. Figure 5(b) shows one such history.

For lengthy histories, a practical method of presenting the result of rain-flow cycle counting is to form a matrix which gives numbers of cycles at various combinations of peak and valley loads. This is illustrated in Fig. 6, where rain-flow cycle counting of the short history of (a) is given by the matrix of (b). Matrix (b) only has entries below the diagonal. In (c), the result is presented in a different form which makes a distinction whether the cycle is ordered peak-valley or valley-peak. In this matrix, there are entries on both sides of diagonal depending on the direction of the cycle, that is, on whether the peak or valley occurs first.

Based on the equations in Fig. 3, a peak-valley matrix can be converted to a matrix which gives numbers of cycles at various combinations of range and mean loads. However, any in-

formation on cycle directions is lost. Hence, a matrix of the type of Fig. 6(c) preserves more information than a range-mean matrix, whereas 6(b) provides equivalent information.

In this paper, three methods of reconstruction based on a rain-flow matrix are discussed. Two methods are based on a two dimensional rain-flow matrix, one of which considers cycle directions, and the third on a three dimensional rain-flow matrix. Note that the two dimensional matrix is a peak-valley matrix, while the three dimensional matrix is a peak-valley-peak matrix. Reconstructed histories by any of these methods will not have the same sequence as the original history. However, all such reconstructed histories produce a rain-flow matrix identical to that for the original history; therefore, the fatigue lives are expected to be similar.

Rain-flow Reconstruction Method Based on a 2-D Matrix

In this method a loading history is first summarized by applying the rain-flow cycle counting method to obtain a compact matrix giving combinations of peak and valley values which correspond to the rain-flow cycles. As already noted, such a matrix can be defined in the two forms illustrated in Fig. 6. In one form, all values above the diagonal line are zero, which indicates that the directions of the cycles are not considered. The other form of the matrix considers the directions of the cycles and has values on both sides of the diagonal. The differences between these the two matrices can be seen in cycles b-c and g-h. If the directions are not considered, these cycles are identical. But if directions are considered, b-c is plotted in the matrix as 2-3, whereas g-h is plotted as 3-2. The detailed procedures employed for reconstruction based on these two types of matrix are explained below.

In Fig. 6, a 5 by 5 matrix is used for illustration purpose only. A higher resolution matrix is needed in practical work. A 32 by 32 matrix is a good compromise in most cases, as this reproduces load levels to within 3% of the largest range and is still relatively compact. Noting that all entries on the diagonal are zero, a 32 by 32 matrix involves 992 numerical values if cycle directions are considered. If the first storage method is applied, that is, if the cycle directions are not considered, then the matrix has a triangular form as in Fig. 6(b). In order to

reconstruct, first the largest cycle, which is in the first (left) column and last (bottom) row, is considered, and then all the columns corresponding to the same row are considered in order. Then the preceding row and corresponding columns are considered in order. This procedure continues until all the elements of the matrix are covered. A cycle can be placed within any cycle in the matrix with equal or more extreme peak and valley, that is, greater or equal row number and less or equal column number. A random location is chosen among all the possibilities, and then the partially reconstructed sequence is rearranged accordingly. The simplest reconstructed history can be formed by placing all the cycles having the same peak and valley in a single location. However, if a more irregular history is desired, the number of cycles for a given peak-valley combination is divided into "n" groups, and each group is placed in a possible location randomly. The value of " n " is optional, with larger values producing a more irregular history but causing computational costs to be greater.

For the other type of matrix where the information retained includes the directions of the cycles, a potentially full 32 by 32 matrix is formed. The elements within the matrix are chosen in the order given by the numbers in Fig. 7 so that the largest cycles are employed first. In Fig. 7, an 8 by 8 matrix is used for illustration only; the same procedure is extended for a 32 by 32 matrix. Four rules for inserting a cycle into a partially reconstructed history are illustrated in Fig. 8 and described below.

If the cycle that is being inserted (inserting cycle) has a greater row than column, that is, if it is ordered peak-valley, then it can be placed within any cycle (receiving cycle), provided:

1. If the receiving cycle is ordered valley-peak, that is, if it has a row less than the column, then the receiving row must be less than or equal to the inserting column, and the receiving column must be greater than or equal to the inserting row. Figure 8(a) illustrates this case.

2. If the receiving cycle is ordered peak-valley, that is, if it has a row greater than column, then the receiving row must be greater than or equal to the inserting row, and the re-

ceiving column must be less than or equal to the inserting column. Figure 8(b) illustrates this case.

On the other hand, if the inserting cycle has a greater column than row, that is, if it is ordered valley-peak, then it can be placed within any cycle, provided:

3. If the receiving cycle is ordered peak-valley, that is, if it has a row greater than the column, then the receiving row must be greater than or equal to the inserting column, and the receiving column must be less than or equal to the inserting row. Figure 8(c) illustrates this case.

4. If the receiving cycle is ordered valley-peak, that is, if it has a row less than column, then the receiving row must be less than or equal to the inserting row, and the receiving column must be greater than or equal to the inserting column. Figure 8(d) illustrates this case.

In addition, the reconstruction must alternate between peaks and valleys. This results in the insertion being made in the rising branch of the receiving cycle for cases (1) and (2), and in the falling branch for (3) and (4). Also, the major cycle could be considered to be either a peak-valley or a valley-peak cycle; the latter is arbitrarily chosen here.

These rules simply ensure that the inserting cycle is within the bounds of the receiving cycle. As an example, consider the history of Fig. 9(a). The inserting cycle 5-4 has a row 5 and a column 4. The partially reconstructed history has the following four rain-flow cycles as identified by the peak-valley or valley-peak levels: 2-6, 6-3, 7-2, and the major cycle, 1-7. Figure 9(b) shows these cycles. If rule number 1 is applied, then the possible receiving cycles would be 2-6 and 1-7. These cycles satisfy rule (1); therefore a random location is chosen between these two possibilities. Figure 9(c) shows the history if 2-6 is chosen as the receiving cycle, while Fig. 9(d) shows the history if 1-7 is chosen as the receiving cycle.

If rule number 2 is applied, then the possible receiving cycle would be 6-3 and 7-2. Note that both cycles satisfy rule number 2. Figures 9(e) and 9(f) show the history with the receiving cycle being 6-3 and 7-2, respectively.

Consider the history in Fig. 10(a). The inserting cycle is 4-5, and the partially reconstructed history is the same as for Fig. 9(a), having the same cycles 2-6, 6-3, 7-2 and 1-7, as already shown in Fig 9(b). If rule number 3 is applied, then the possible receiving cycles would be 6-3 and 7-2. Note that both cycles satisfy rule number 3. Figures 10(b) and (c) show the history with receiving cycles 6-3 and 7-2, respectively.

If rule number 4 is applied, then the possible receiving cycles would be 2-6 and 1-7. Both cycles satisfy rule (4). Figures 10(d) and (e) show the history with the receiving cycle 2-6 and 1-7, respectively.

Note that the location for the cycle under consideration is chosen among all the possibilities, and as before there exist two general options for the reconstructed history, namely, the simplest form and the irregular form. In the latter case, the number of cycles for a given peak-valley combination is divided into "n" groups, and each group is placed in a possible location randomly.

The reconstructed histories by either of the above described procedures will not have the same sequence as the original history. The reason is that a given minor cycle can be placed in a variety of locations, as its original location was not preserved by the peak-valley matrix of the rain-flow cycles. However, all reconstructed histories produce a rain-flow matrix identical to that for original history.

For the first method of reconstruction, that is , for the method not considering cycle directions, recall that all cycles in a given row were used for reconstruction before the next row was considered. Other orders of using cycles could have been employed, with the only absolute requirement being that the largest cycle be used first. The procedure used for the second method, where successively smaller ranges are considered, is preferable. The reason is that

this approach maximizes the number of possible locations where a given cycle can be placed as all larger cycles are already present.

Rain-flow Reconstruction Method Based on a 3-D Peak-Valley-Peak Matrix

This method is similar to the two dimensional reconstruction method (cycle directions considered), except another dimension is added to the matrix. Also, the rain-flow matrix was used with a size of 16 by 16 by 16. Greater resolution, say 32 by 32 by 32, could be used subject to limitations on computational time and cost. This method follows rules 2 and 4 plus an additional rule which must also be introduced. The new rule states that if the inserting cycle is ordered peak-valley, then the receiving cycle must also be ordered peak-valley, and the receiving peak must be equal to the third dimension of the inserting cycle. However, if the inserting cycle is ordered valley-peak, then the receiving cycle must also be ordered valley-peak, and the receiving valley must be equal to the third dimension of the inserting cycle.

For example, consider Fig. 6(c). In the two dimensional approach, cycle g-h, has a starting level of 3 and a target level of 2, while in the three dimensional approach, this cycle is stored as g-h-a, which has a starting level of 3, an intermediate level of 2, and a target level of 5.

Rain-Flow Reconstruction of Maneuver History

A loading history for the tail rotor pitch beam of an AUH-76 helicopter was selected as representative, and loading histories from each of 30 distinct severe maneuver were assumed to occur once in a specific sequence. This produced a loading history containing 33,470 cycles, which was then modified by the University of Dayton Research Institute (UDRI) to eliminate minor events, shortening it to 8777 cycles, that is 8777 peaks and 8777 valleys.

In addition, the modified history was further shortened by filtering all the rain-flow cycles with the range less than 0.45 units. Note that the history is scaled so that the highest peak is 1 unit, which results in the lowest valley being -0.517 units, and the largest rain-flow range 1.517 units. This filtered history contains 510 cycles and is shown in Fig. 11(a). Figure 12 shows the peak-valley matrix of this history, specifically the matrix with cycle directions considered. This history is explained in detail in Refs. [9,11].

Computer programs based on the above descriptions were developed and used for reconstruction of the filtered maneuver history. The filtered history was first summarized using rain-flow cycle counting into the compact form of a 32 by 32 matrix giving combinations of peak and valley values which correspond to rain-flow cycles. Recall that a history can be reconstructed in two ways depending on two variations of the rain-flow matrix.

Figures 11(b) and 11(c) show reconstructed histories where the cycle directions are not considered. The simplest reconstructed history is 11(b), and a more irregular reconstructed history is 11(c). The simplest history is obtained by placing all the cycles having the same peak and valley in a single location. The more irregular version can be formed by simply dividing the number of cycles for a given peak-valley combination into "n" groups, and each group is placed into a possible location randomly. For Fig. 11(c), a value of $n=3$ was used. Figure 11(d) shows the simplest reconstructed history where the directions of the cycles are retained.

The filtered history was also summarized by using the rain-flow method in the compact form of a three dimensional 16 by 16 by 16 matrix giving combinations of peak, valley, and peak, or valley, peak, and valley. Figure 13 shows the simplest form of the reconstructed history.

Discussion

Based on the discussion presented in this paper, it appears that a promising reconstruction method is some version of a rain-flow reconstruction. Although rain-flow reconstructed histories do not generally produce the same loading sequence as the original history, this is not

expected to affect the life significantly. This occurs because the rain-flow matrix will be identical to the original rain-flow matrix; therefore, a similar life is expected.

As already noted, Perrett's work [8] suggests that reconstructed histories producing identical rain-flow cycles give similar lives as original histories. This argument is strengthened by [11] based on the comparison of calculated fatigue lives of original and reconstructed histories. Note that Perrett's conclusion is based on test data for total life for crack initiation plus growth, and also on analysis of crack growth for standard aircraft spectra. He also suggests that, in all cases where the reconstruction process has been randomized, there is not a significant load interaction effect beyond that accounted for by rain-flow counting. The evidence to date suggests that all of the possible rain-flow reconstructions from a given rain-flow matrix cause similar lives for both crack initiation and growth.

Availability of testing equipment is important in determining the best method of rain-flow reconstruction. If the test equipment is limited to constant amplitude cycling, it would be the best choice if the rain-flow directions are not considered, and if all cycles with the same peak and valley are put in one location. However, for a more advanced equipment, where there is no restriction on the irregularity of the loading history, the realism of the reconstructed history can be increased by retaining the rain-flow cycles directions, and putting cycles with the same peak and valley in more than one location.

Comparison between the three dimensional and two dimensional methods indicates that the three dimensional method involves additional complexities, greater storage by a factor of 32, and greater computational time without any apparent benefit. Therefore, the two dimensional approach appears to be sufficient.

This work on rain-flow reconstruction is currently being extended to obtain experimental verification.

Conclusion

Reconstruction of loading histories from a concise description based on rain-flow cycle counting appears to be a promising approach. This produces a history with an altered ordering of events but with a rain-flow cycle counting matrix identical to the one for the original history. As a result of reproducing the rain-flow cycles, similar fatigue lives are expected for original and reconstructed histories.

Acknowledgement

This work was supported by Grant No. NAG-1-822, funded by the U.S. Army Aerostructures Directorate at NASA Langley Research Center, Hampton, VA. Gratitude is expressed to R. A. Everett, the technical monitor, for his aid.

References

1. Wirsching, P. H. and Light, M. C., "Fatigue Under Wide Band Random Stresses," *Journal of the Structural Division, ASCE,* Vol. 106, No. ST7, Proc. paper 15574, July 1980, pp. 1593-1607.

2. Haibach, E., Fischer, W., and Huck, M., "A Standard Random Loading Sequence of Gaussian Type Recommended for General Application in Fatigue Testing: Its Mathematical Background and Digital Generation," *Fatigue Testing and Design,* Vol 2, 1976.

3. Fash, J. W., Conle, F. A., and Minter, G.L., "Analysis of Irregular Loading Histories for SAE Biaxial Fatigue Program," to be published by *the Society of Automotive Engineers,* Warrendale, Pa.

4. Dowling, N. E. and Thangitham, S., "Concise Description and Reconstruction of Spectrum Loading," *Proceeding of the 14th ICAF Symposium,* Ottawa, Canada, June 1987.

5. Dowling, N. E., "Fatigue Failure Prediction for Complicated Stress-Strain Histories," *Journal of Materials,* ASTM, Vol. 7, No. 1, March 1972, pp. 71-87.

6. Conle, A., "An Examination of Variable Amplitude Histories in Fatigue," Ph.D. Thesis, Dept. of Civil Engineering, University of Waterloo, Ontario, Canada. 1979.

7. ten Have, A. A.,"European Approaches in Standard Spectrum Development," NLR-MP-87007-U, National Aerospace Laboratory, Emmerloord, The Netherlands. 1987. Pub. in ASTM STP 1006, *Development of Fatigue Loading Spectra,* American Society for Testing and Materials, Philadelphia, PA, 1989.

8. Perrett, B., "An Evaluation of a Method for Reconstructing Fatigue Test Loading Sequences From Load Data Acquired via Rain-flow Counting," *Proceedings of the 14th ICAF Symposium,* Ottawa, Ontario, Canada, June 1987.

9. Khosrovaneh, A. K. and Dowling, N. E.,"Analysis and Reconstruction of Helicopter Load Spectra," *Paper for the American Helicopter Society National Technical Specialists Meeting on Advanced Rotorcraft Structures*, Williamsburg, VA, Oct. 1988

10. Schijve, J., "The Analysis of Random Load-Time Histories with Relation to Fatigue Tests and Life Calculations," 2nd ICAF-AGARD Symp., Paris, France, May 1961.

11. Khosrovaneh, A. K., "Fatigue Analysis and Reconstruction of Helicopter Load Spectra," Ph.D. Thesis, Dept. of Engineering Science and Mechanics, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1989.

12. van Dijk, G. M., "Statistical Load Data Processing," 6th ICAF Symp., Miami, Fla., NASA SP 309, 1971.

13. Gassner, E., "Festigkeitsversuche mit wiederholter Beansprunchung im Flugzeugbau," Luftwissen, Vol. 6, 1939, Translation NACA TM-1087.

14. Dowling, N. E. and Khosrovaneh, A. K., "Simplified Analysis of Helicopter Load Spectra," ASTM STP 1006, *Development of Fatigue Loading Spectra*, American Society for Testing and Materials, Philadelphia, PA, 1989.

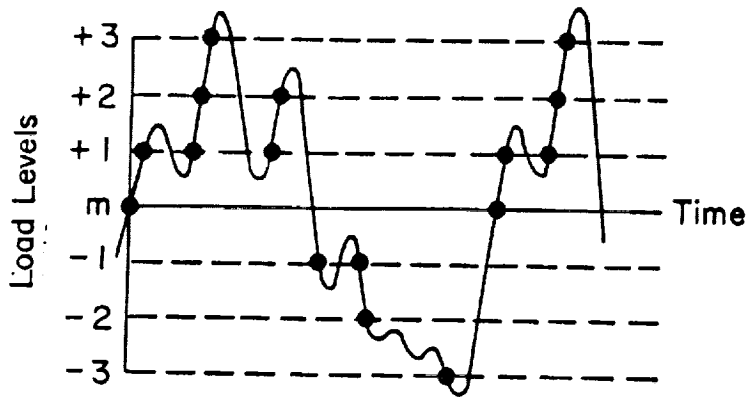
15. Conle, A. and Topper, T. H., "Fatigue Service Histories: Technique for Data Collection and History Reconstruction," Paper No. 820093, Society of Automotive Engineers, 1983.

16. "Standard Practice for Cycle Counting in Fatigue Analysis," 1986 Annual Book of ASTM Standards, Vol. 03.01, Standard No. 1049, pp. 836-848.

Table 1. Cycle counting result for Fig. 4.

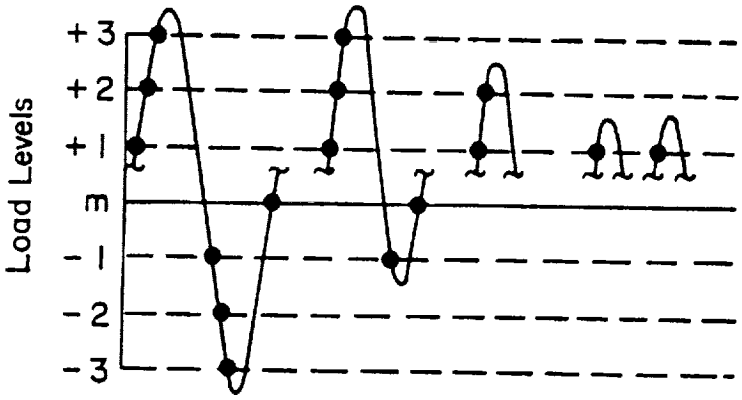
Cycle†	Load Units		Load Units		Levels (0 to 32)	
	Start	Target	Range	Mean	Start	Target
E-F	-1	3	4	1	11	25
A-B	-2	1	3	-0.5	8	18
H-C	4	-3	7	0.5	29	4
D-G	5	-4	9	0.5	32	1

†Each of these cycles occurs only once.



(a) Level Crossing Counting

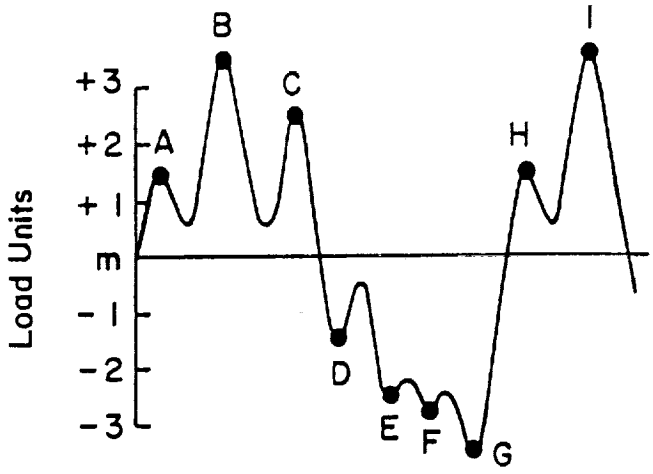
Level	Counts
+3	2
+2	3
+1	5
0	2
-1	2
-2	1
-3	1



(b) Cycles Derived from Level Crossing Count of (a)

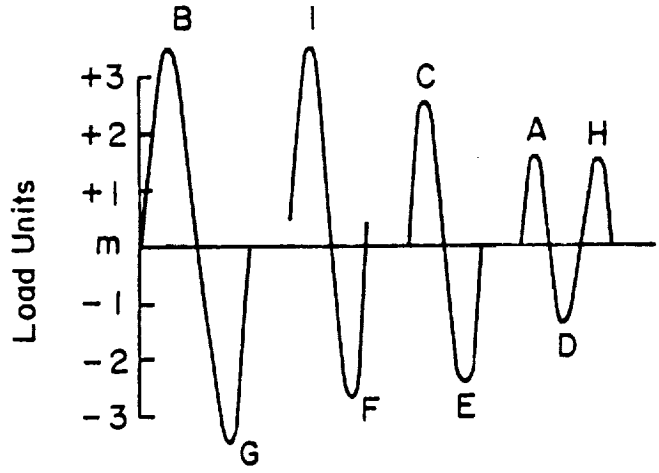
Range (levels)	Cycle Counts
7	1
6	0
5	1
4	0
3	0
2	1
1	2

Figure 1. Illustration of level crossing counting (a), and level crossing reconstruction (b) [16]



Peak	Counts
+3.5	2
+2.5	1
+1.5	2
-1.5	1
-2.5	1
-2.7	1
-3.5	1

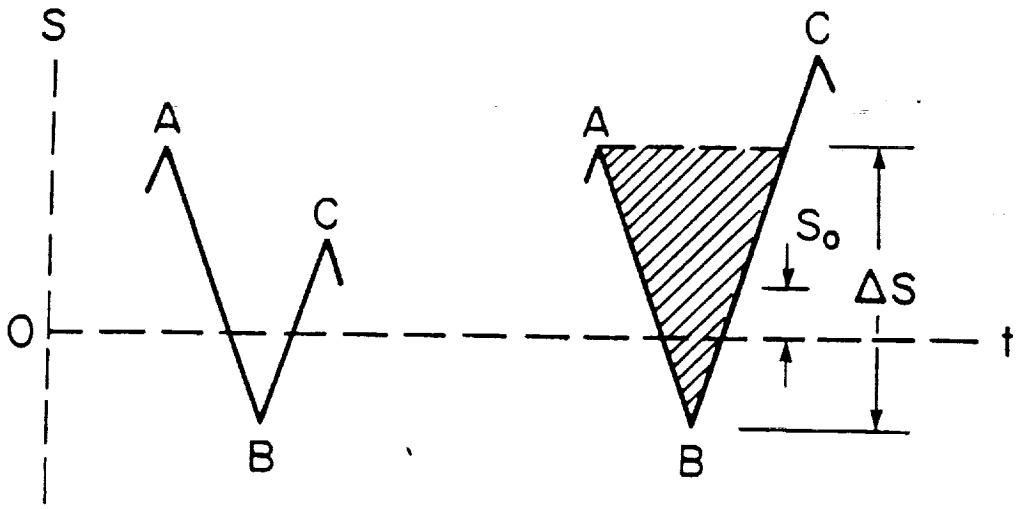
(a) Peak Counting



Range (units)	Cycle Counts
7	1
6.2	1
5	1
3	1.5

(b) Cycles Derived from Peak Count of (a)

Figure 2. Illustration of peak-valley counting (a), and peak-valley reconstruction(b) [16]



$\overline{BC} < \overline{AB}$ $\quad$ $\overline{BC} \geq \overline{AB}$

No cycle $\quad$ AB = cycle

For cycle A - B

Peak = S_A

Valley = S_B

Range = $\Delta S = S_A - S_B$

Mean = $S_o = (S_A + S_B)/2$

Figure 3. Condition for recording an event during rain-flow cycle counting

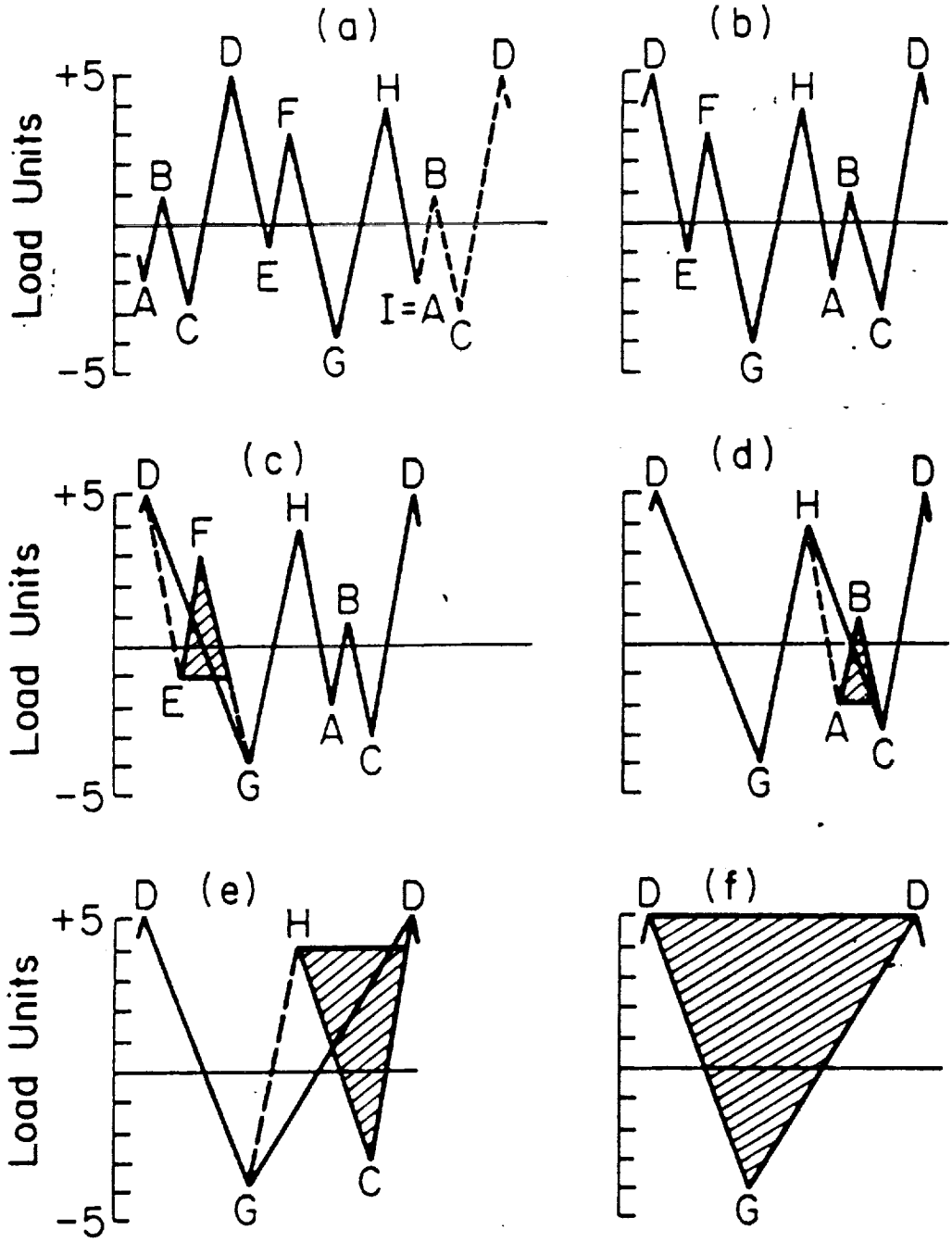


Figure 4. **Example of rain-flow cycle counting from ASTM standards [16]:** Before cycle counting begins, the most extreme point in the history should be located, and the history rearranged to start and finish at this point as shown in (b).

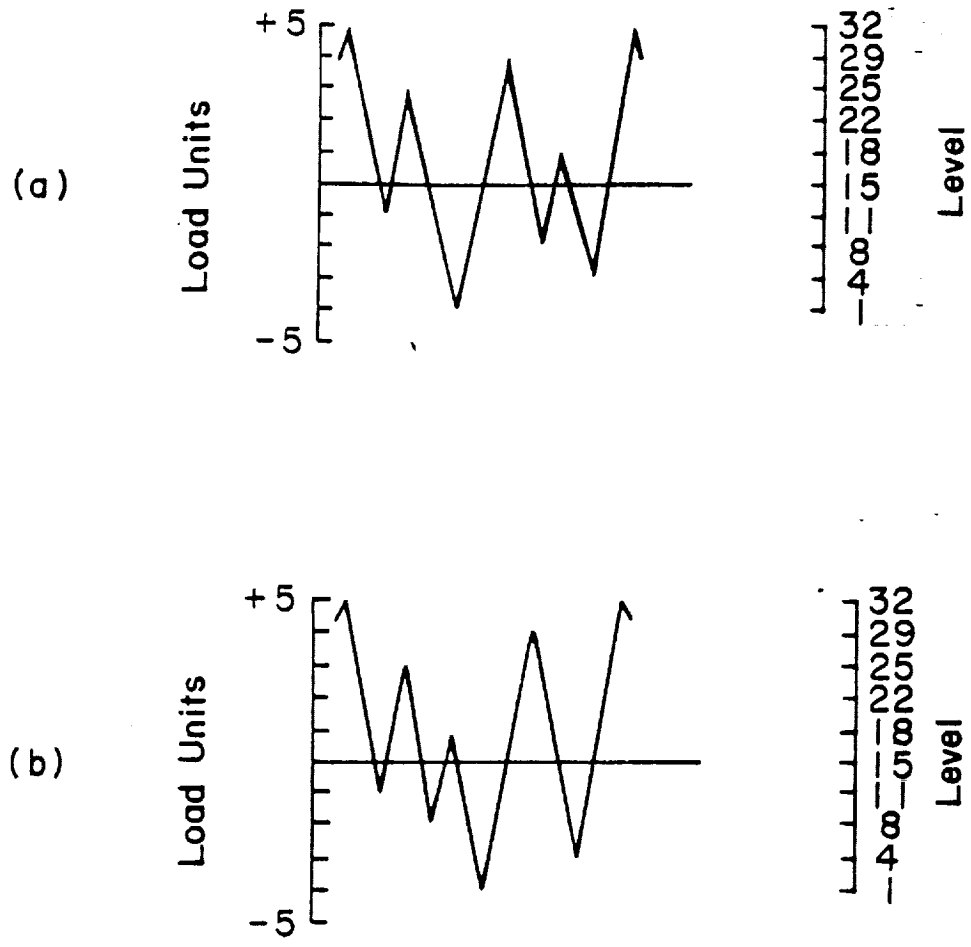


Figure 5. Comparison of original (a), and reconstructed (b) histories for the Fig.4 example

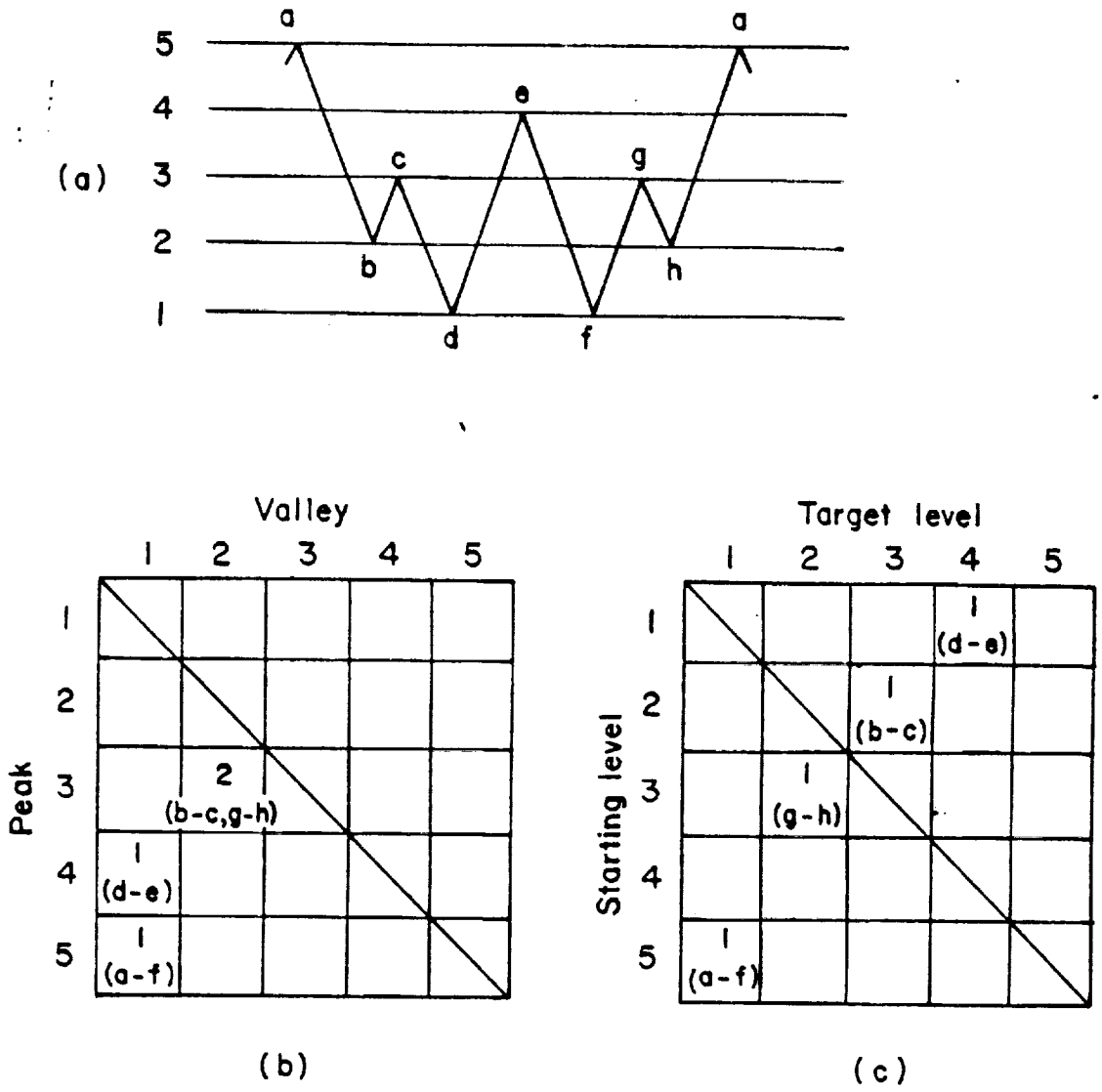


Figure 6. Two forms of matrix definition for the purpose of rain-flow reconstruction: (a) example history, and matrices without (b) and with (c) directions of cycles recorded.

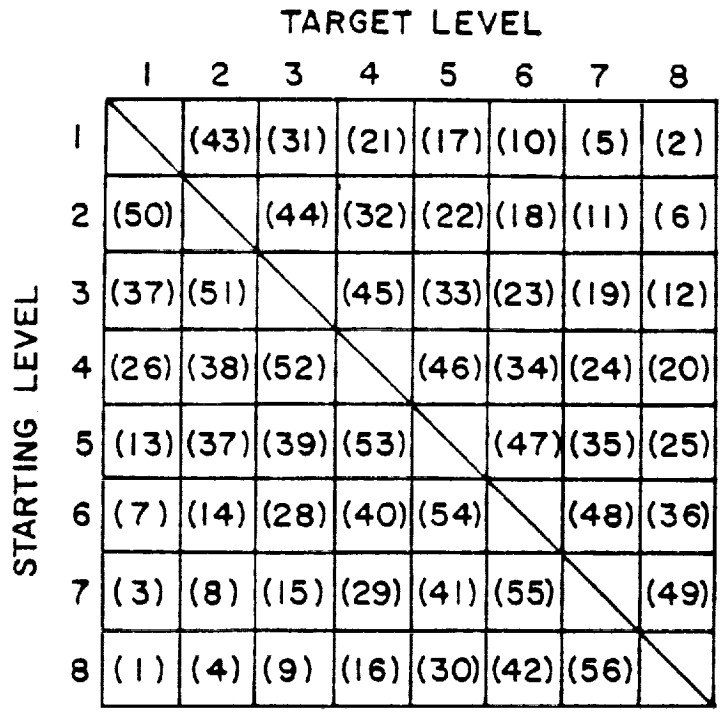


Figure 7. Order of insertion of cycles into the partially reconstructed history where the directions of the cycles are considered

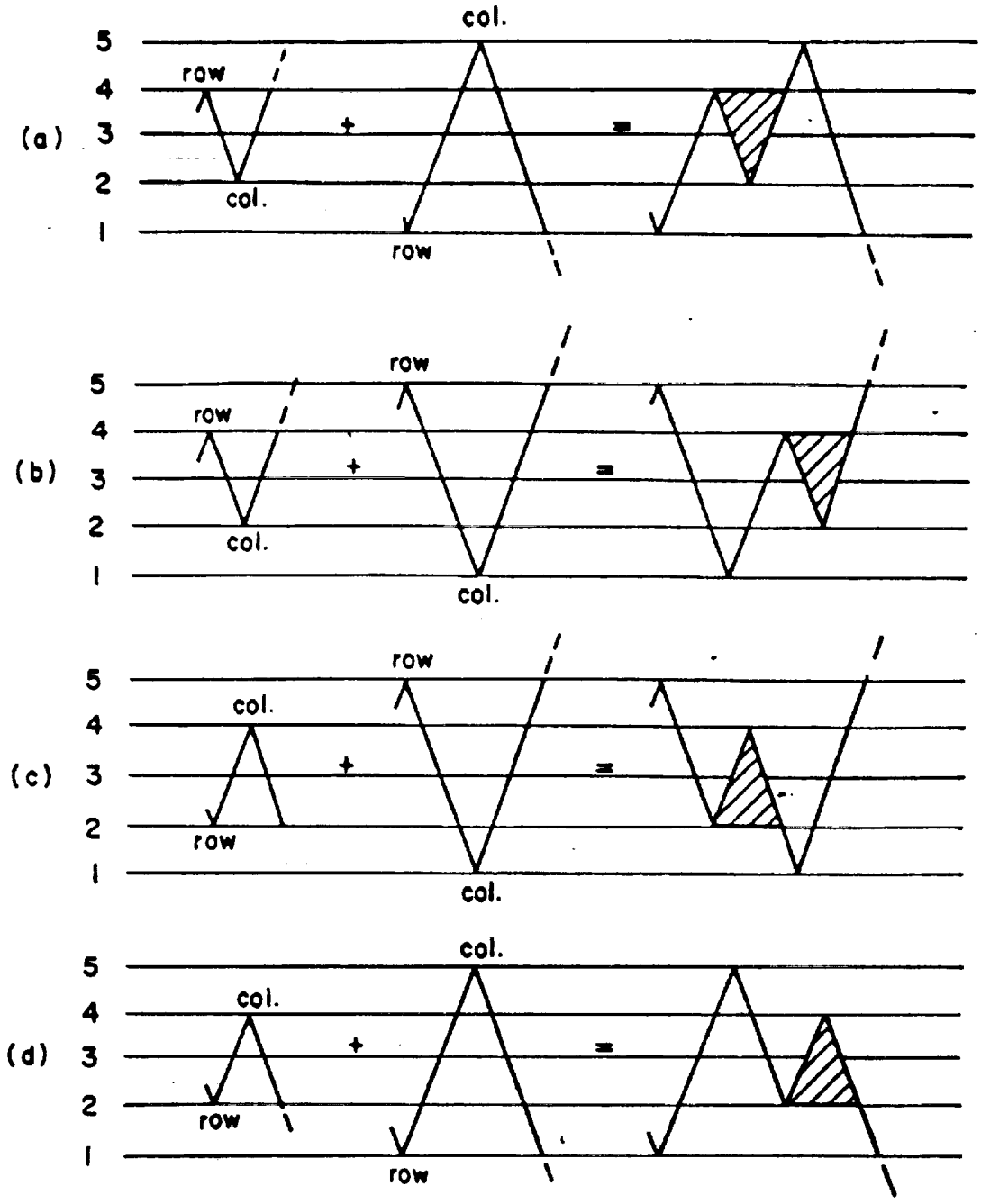


Figure 8. Illustration of rules 1-4 for rain-flow reconstruction where the directions of the cycles are considered

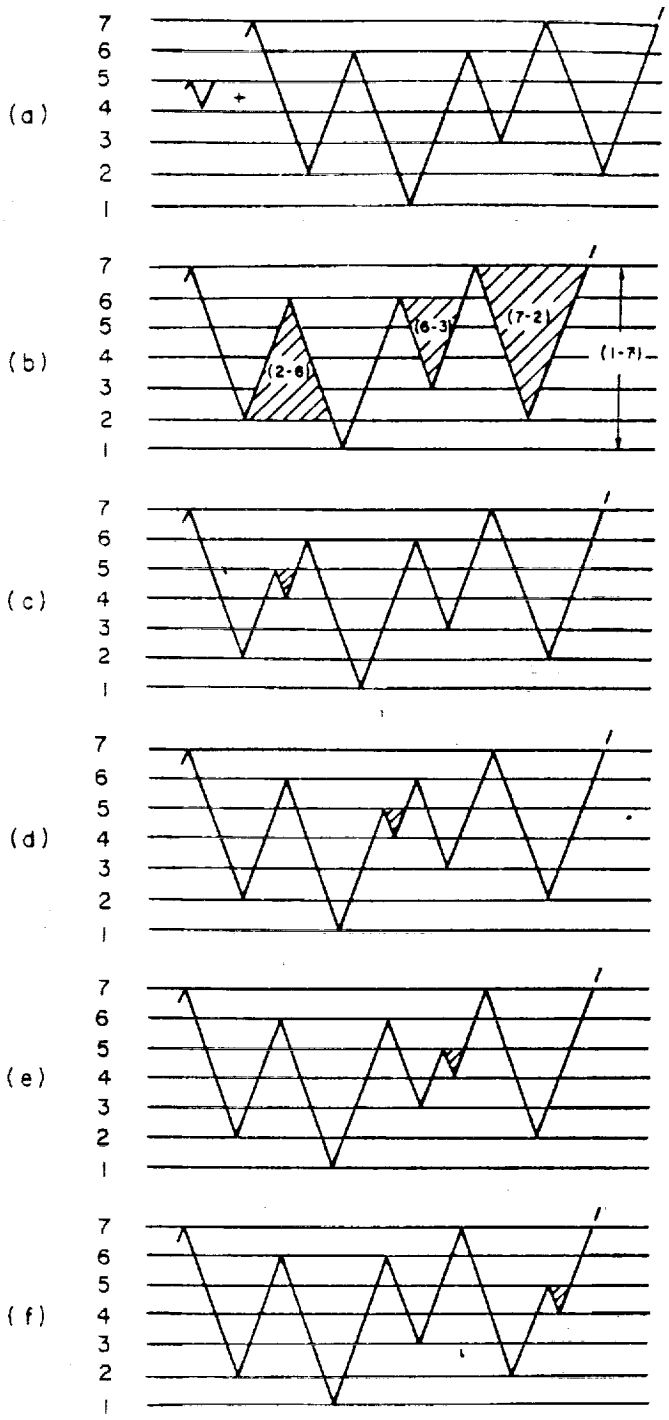


Figure 9. Example of rain-flow reconstruction where the inserting cycle is ordered peak-valley: The insertion of (a) can be made into any of the rain-flow cycles of (b), with the four possibilities being (c), (d), (e) and (f).

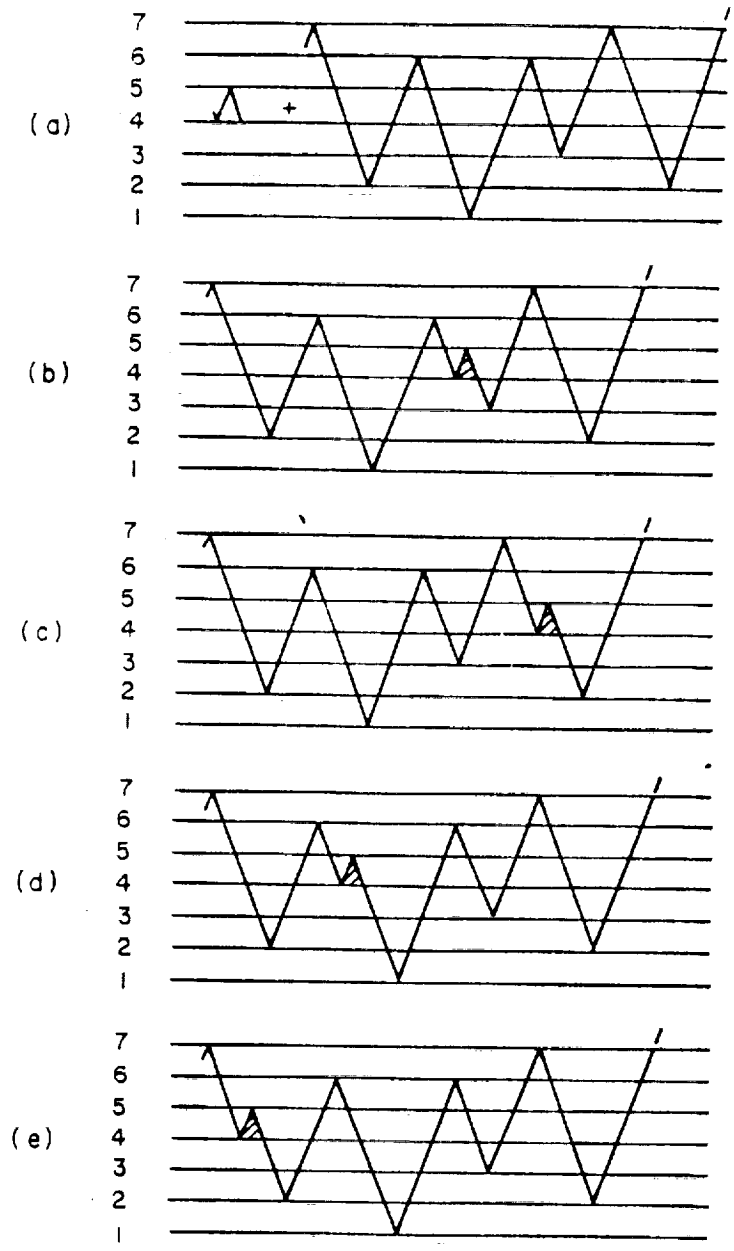


Figure 10. Example of rain-flow reconstruction where the inserting cycle is ordered valley-peak: The insertion of (a) can be made into any of the rain-flow cycles of Fig. 9(b), with the four possibilities being (b), (c), (d) and (e).

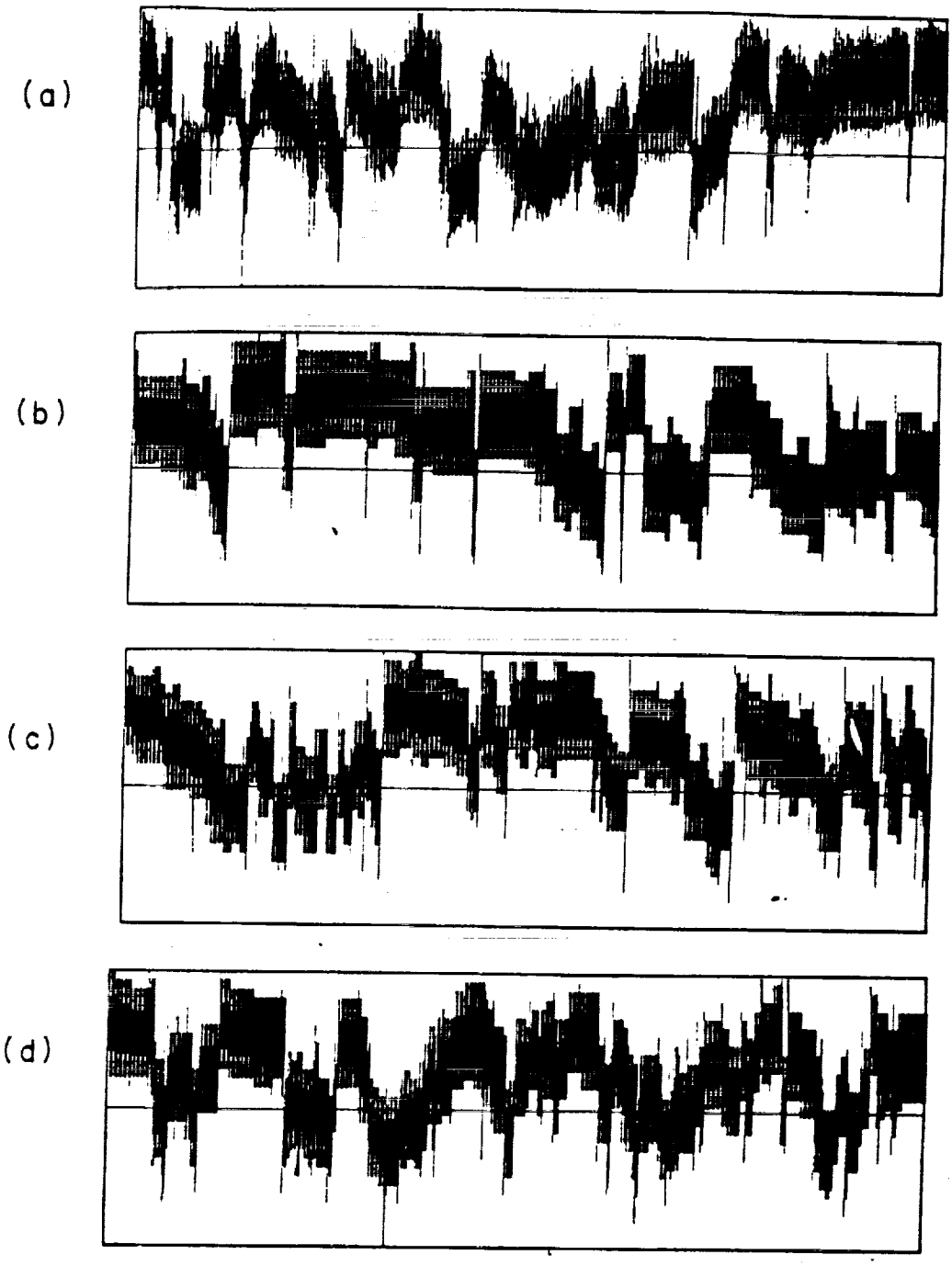


Figure 11. Rain-flow reconstruction of the filtered maneuver history: (a)filtered maneuver history, (b) reconstructed with all cycles of a given peak and valley in one location, where the direction of the cycle not considered, (c) reconstructed similarly but with large blocks of cycles split into three different locations, and (d) reconstructed with all cycles of a given peak and valley in one location, but where the directions of the cycles are now considered.

ORIGINAL PAGE IS
OF POOR QUALITY

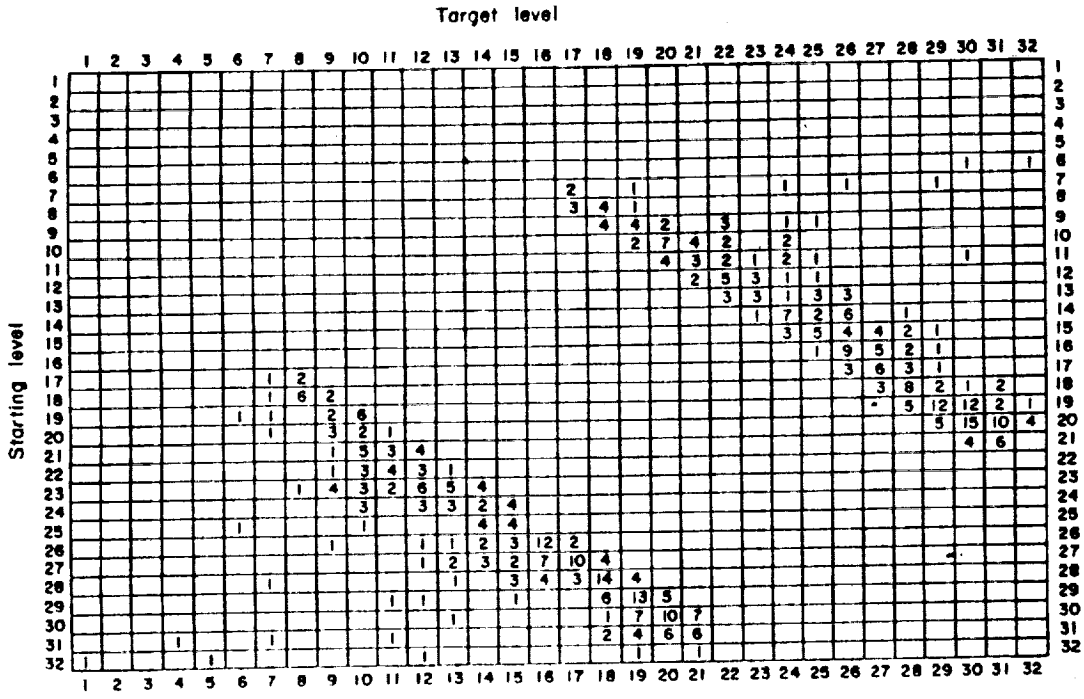


Figure 12. Peak-valley matrix with directions considered from rain-flow cycle counting for the filtered history

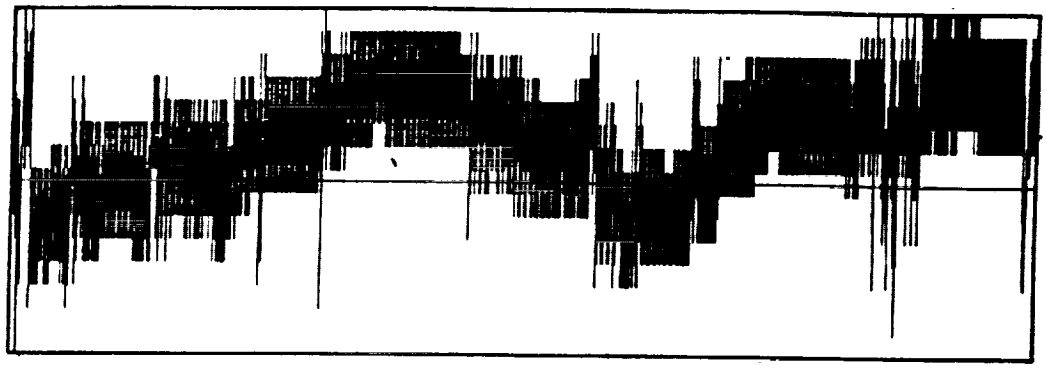


Figure 13. Three-dimensional matrix reconstruction of the filtered maneuver history

Appendix A. Computer Program RAINF2 for Rain-flow Cycle Counting Analysis

A computer program for rain-flow cycle counting is provided. The program can take a lengthy load history and reduce it to a compact form of a matrix giving combinations of range and mean or peak and valley values. This information can be used for fatigue analysis. The input values are defined, and two examples using different options of the program are provided. The differences with an earlier version (RAINF1) are minor and are confined to two areas of the program as indicated by comment statements.

Program Logic

The following logic is used consistent with the ASTM Standard E1049 (Ref. 16 of this report): Let x denote the absolute value of the range under consideration, and y the previous absolute range adjacent to x.

Step 1: Determine the maximum absolute value in the history. (Note that this value can be either a peak or a valley.)

Step 2: Arrange the history to start with the maximum absolute value. Move all peaks and valleys which occur prior to the maximum load to the end as illustrated in Fig. A.1(b).

Step 3: Read the next value. If out of data, go to step 9.

Step 4: Three points are needed to define x and y. If there are less than three points, go back to step 3. Define x and y using the three most recent peaks and valleys that have not been discarded.

Step 5: Compare the two ranges, namely x and y. If x is less than y, go to step 3; otherwise go to step 6.

Step 6. If a rain-flow filtered history is not desired, go to step 8.

Step 7: If y is less than or equal to the filter level specified in the program input, discard the peak and valley of the range y in the array in memory, which is the original history of step 2.

Step 8: Count range y as one cycle, determine the mean value of the peak and valley of y, discard the peak and valley of y in the array set up in step 3, and go to step 4.

Step 9: Stop.

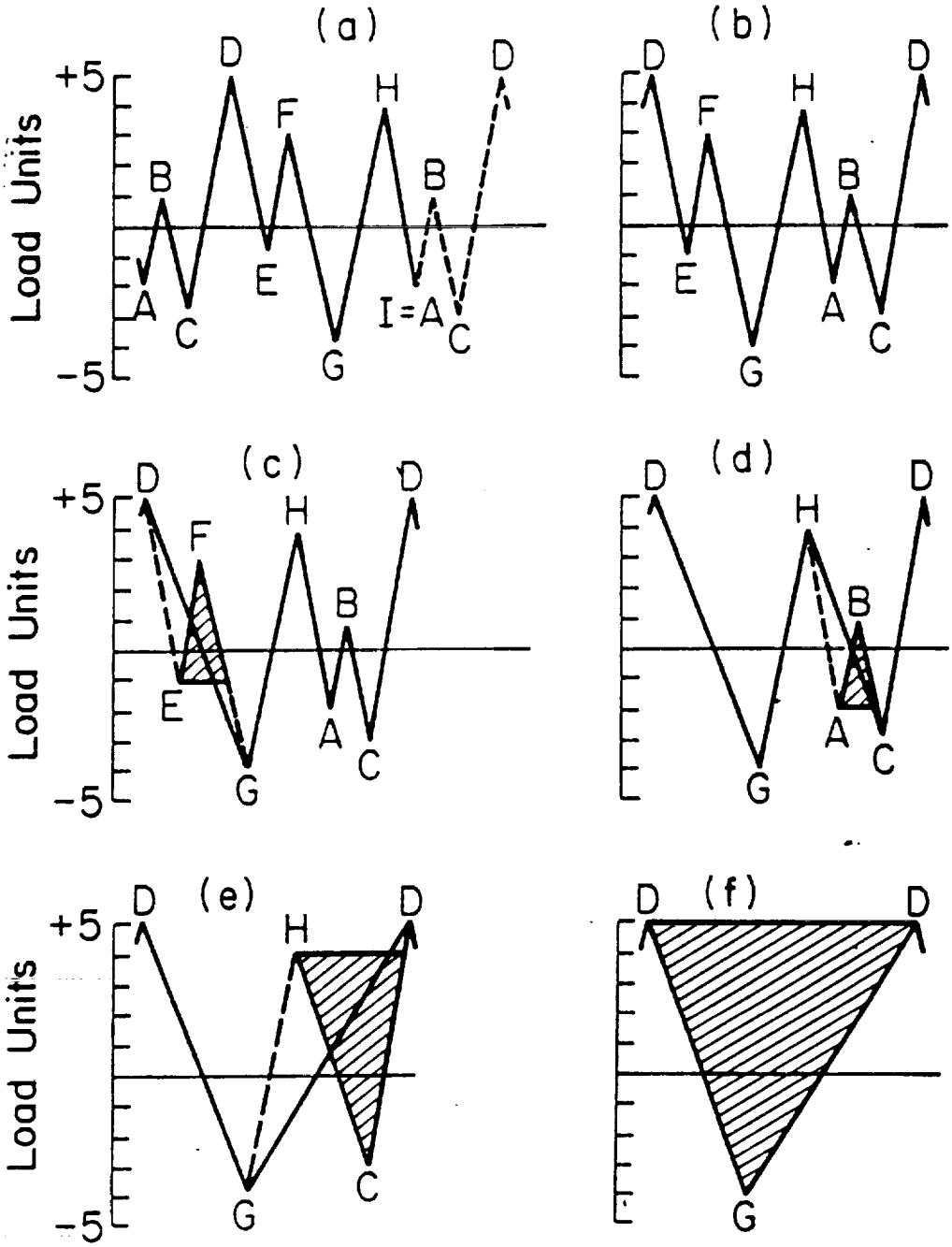


Figure A.1. Example of rain-flow program logic.

Definition of Input Data

Data line 1: OPTION = 1 List filtered history as peak-valley sequence; also print range-mean matrix of rain-flow cycles for original history.

2 List range, mean, minimum, and maximum of rain-flow cycles not in matrix form.

3 Print range-mean matrix of rain-flow cycles.

4 Print starting versus target level, 32 by 32, matrix of rain-flow cycles. Note the directions of cycles are stored in this matrix. The starting level for a cycle is either a peak (max) value or a valley (min) value, whichever occurs first, and the target level is the other peak or valley that defines the cycle. Also, the history is converted to a minimum value of 1 and maximum value of 32.

Data line 2: FL = Filter level as a range value. Note that this data is required only for OPTION = 1, and is otherwise omitted.

Data line 3: NN = Number of peak-valley points in the history. The history must start and end with the same value, so that this starting/ending value is counted twice in NN, and NN must be an odd number.

Data line 4: XIM = Constant increment between mean values in the range-mean matrix. Note that this data is required only if OPTION = 1 or 3, and is otherwise omitted.

XIR = Constant increment between range values in the range-mean matrix. Note that this data is required only if OPTION = 1 or 3, and is otherwise omitted.

Data line 5: P( ) = Input load history as peaks and valleys in sequence. Note that the history must start and end with the same value, and the direction of loading must reverse at each value.

Example 1

The history of Fig. A.1 is used for this rain-flow cycle counting example. Option 2 of the program is used; Therefore, the result is shown as a list of range, mean, minimum(valley), and maximum (peak) values. The entire program listing and program input and output for this example follow.

Program Input

```
2
9
-2.,1.,-3.,5.,-1.,3.,-4.,4.,-2.
```

PROGRAM LISTING

```
C           RAIN-FLOW COUNTING PROGRAM (RAINF2)
C    NOTE THAT THE HISTORY MUST START AND END WITH THE SAME VALUE.
C
C    INPUT
C    DATA LINE 1.  OPTION=1   LIST FILTER HISTORY AS PEAK/VALLEY
C                       SEQUENCE.ALSO PRINT RANGE/MEAN
C                       MATRIX OF RAINFLOW CYCLES FOR
C                       ORIGINAL HISTORY.
C                  =2   LIST RANGE,MEAN,MIN,AND MAX OF RAIN-
C                       FLOW CYCLES NOT IN MATRIX FORM.
C                  =3   PRINT RANGE/MEAN MATRIX OF RAINFLOW
C                       CYCLES.
C                  =4   PRINT START/TARGET MATRIX OF RAINFLOW
C                       CYCLES.
C    DATA LINE 2.       FL=FILTER VALUE AS A RANGE
C    DATA LINE 3.       NN=NUMBER OF PEAK/VALLEY POINTS IN HISTORY
C    DATA LINE 4.       XIM=CONSTANT INCREMENT BETWEEN MEAN VALUES
C                       IN THE RANGE/MEAN MATRIX.
C                  XIR=CONSTANT INCREMENT BETWEEN RANGE VALUES
C                       IN THE RANGE/MEAN MATRIX.
C              NOTE THAT XIM AND XIR REQUIRED FOR OPTION=1 OR 3
C    DATA LINE 5.       P( )=INPUT LOAD HISTORY AS PEAKS AND VALLEYS
C                       IN SEQUENCE.
C
C
C
      REAL P(10000),PE(10000),PP(10000),PC(10000),PCC(10000),MM(64),
     *RA(64),PI(10000),MEAN(5005),R(5005)
      INTEGER M(64,64),SUM(64),SUMM,MI(32,32),OPTION
      READ(5,*)OPTION
      IF(OPTION.EQ.1)READ(5,*)FL
      READ(5,*)NN
      IF(OPTION.EQ.4)GO TO 40
      IF(OPTION.EQ.2)GO TO 40
      READ(5,*)XIR,XIM
40    READ(5,*)(P(I),I=1,NN)
      N=NN
C
C
C     DETERMINATION OF LARGEST PEAK OR VALLEY
      LCOUNT=1
      DO 100 I=1,N
      PE(I)=P(I)
100   CONTINUE
      PMAX=ABS(P(1))
      DO 200 I=2,N
      IF(PMAX.GE.PE(I)) GO TO 200
      PMAX=ABS(PE(I))
      LCOUNT=I
200   CONTINUE
      IF(OPTION.EQ.4)THEN
      SMAX=P(1)
      SMIN=P(1)
      DO 301 I=2,NN
```

```
      IF(P(I).GT.SMAX)SMAX = P(I)
      IF(P(I).LT.SMIN)SMIN = P(I)
  301  CONTINUE
      CF1 = SMIN
      CF2 = SMAX
      CF3 = SMAX-SMIN
      END IF
C
C
C     ARRANGE THE PEAK OR VALLEY
      JK = LCOUNT + 1
      J = N-JK + 1
      KKK = LCOUNT
      DO 300 I = 1,J
      PP(I) = P(KKK)
      KKK = KKK + 1
  300  CONTINUE
      J = J + 1
      DO 350 I = 1,LCOUNT
      PP(J) = P(I)
      J = J + 1
  350  CONTINUE
      DO 500 I = 1,NN
      PC(I) = PP(I)
  500  CONTINUE
      NNN = N + 1
      IF(OPTION.EQ.2)WRITE(6,210)
C
C
C     FINDING THE CYCLE
      AA = 3.1422
      DO 194 I = 1,32
      DO 195 J = 1,32
      MI(I,J) = 0
  195  CONTINUE
  194  CONTINUE
      I = 0
      K = 1
      IF(OPTION.EQ.2)WRITE(6,107)
      J = 1
    2  I = I + 1
      IF(I.LT.3) GO TO 2
      J = J + 1
      IF(I.EQ.NNN) GO TO 400
   50  IF(PP(J).EQ.AA) THEN
      J = J-1
      GO TO 50
      END IF
      JM1 = J-1
   60  IF(PP(JM1).EQ.AA) THEN
      JM1 = JM1-1
      GO TO 60
      END IF
   70  IF(I.GT.NNN) GO TO 400
      X = ABS(PP(I)-PP(J))
      Y = ABS(PP(J)-PP(JM1))
```

```
      XX=(PP(J)+PP(JM1))/2.
5     IF(X.GE.Y)THEN
      IF(OPTION.NE.1)GO TO1600
      IF(Y.LE.FL) THEN
      PC(J)=AA
      PC(JM1)=AA
      END IF
1600  IF(OPTION.EQ.2)GO TO 41
      IF(OPTION.EQ.4)THEN
      PI(J)=((32.*(-CF1+PP(J)))+CF2-PP(J))/CF3
      PI(JM1)=((32.*(-CF1+PP(JM1)))+CF2-PP(JM1))/CF3
      PMAX=PI(J)
      PMIN=PI(JM1)
C
C     THE NEXT 4 LINES REPLACE 8 LINES IN RAINF1.
C
      END IF
      IF(OPTION.EQ.4)THEN
      EIJ=PMIN+.5
      EJI=PMAX+.5
      IJ=INT(EIJ)
      JI=INT(EJI)
      MI(IJ,JI)=MI(IJ,JI)+1
      END IF
      IF(OPTION.EQ.4)GO TO 42
      R(K)=Y
      MEAN(K)=XX
      K=K+1
      GO TO 42
41    PMAX=PP(J)
      PMIN=PP(JM1)
      IF(PMAX.LT.PMIN)THEN
      PMAX=PP(JM1)
      PMIN=PP(J)
      END IF
      WRITE(6,108)Y,XX,PMAX,PMIN
42    PP(J)=AA
      PP(JM1)=AA
      J=J-1
3     J=J-1
      IF(J.LT.1) GO TO 11
      IF(PP(J).EQ.AA) GO TO 3
      IF(I.EQ.4) THEN
      I=5
      J=I-1
      END IF
      JM1=J-1
      IF(JM1.LT.1)THEN
      JM1=J
      J=I
      I=I+1
      GO TO 70
      END IF
6     IF(PP(JM1).EQ.AA) GO TO 4
      X=ABS(PP(I)-PP(J))
      Y=ABS(PP(J)-PP(JM1))
```

```
      XX=(PP(J)+PP(JM1))/2.
      GO TO 5
 4    JM1=JM1-1
      IF(JM1.LT.1)THEN
      JM1=J
      J=I
      I=I+1
      END IF
      GO TO 6
 11   I=I+2
      J=I-1
      JM1=J-1
      GO TO 70
      ELSE
      XX=(PP(J)+PP(JM1))/2.
      J=I-1
      END IF
      GO TO 2
C
C
C
400   IF(OPTION.EQ.2) GO TO 999
      IF(OPTION.EQ.4)GO TO 998
      K=K-1
      RMAX=R(1)
      RMIN=R(1)
      RMMAX=MEAN(1)
      RMMIN=MEAN(1)
      DO 1800 I=2,K
      IF(R(I).GT.RMAX)RMAX=R(I)
      IF(R(I).LT.RMIN)RMIN=R(I)
      IF(MEAN(I).GT.RMMAX)RMMAX=MEAN(I)
      IF(MEAN(I).LT.RMMIN)RMMIN=MEAN(I)
1800  CONTINUE
      DIFR=RMAX-RMIN
      DIFM=RMMAX-RMMIN
      ER=DIFR/XIR
      EM=DIFM/XIM
      ER=ER+2
      EM=EM+2
      LEVLM=INT(EM)
      LEVLR=INT(ER)
      DO 192 L=1,LEVLR
      DO 193 LL=1,LEVLM
      M(L,LL)=0
193   CONTINUE
192   CONTINUE
      YA=RMIN-XIR
      XA=RMMIN-XIM
      WRITE(6,111)RMIN,RMAX,RMMIN,RMMAX
      XB=.50
      YB=.50
      DO 1900 I=1,K
      EI=((R(I)-YA)/XIR)+YB
      EJ=((MEAN(I)-XA)/XIM)+XB
      II=INT(EI)
```

```
      JJ=INT(EJ)
      IF(II.EQ.0)II=1
      IF(JJ.EQ.0)JJ=1
      M(II,JJ)=M(II,JJ)+1
 1900 CONTINUE
C
C
C     FILTERING PROCESS
      IF(OPTION.NE.1)GO TO 1102
      KN=1
      DO 1000 II=1,NN
      IF(PC(II).EQ.AA) GO TO 1000
      PCC(KN)=PC(II)
      KN=KN+1
 1000 CONTINUE
      KN=KN-1
      WRITE(6,112)
  112 FORMAT('1',//15X,'FILTER HISTIRY-PEAK/VALLEY SEQUENCE')
      WRITE(6,113)FL
  113 FORMAT(//15X,'FILTER LEVEL=',F7.3)
      WRITE(6,1103) KN
 1103 FORMAT(//15X,'NUMBER OF POINTS IN FILTER HISTORY=',I5,//)
      WRITE(6,1001)(PCC(I),I=1,KN)
 1001 FORMAT(1X,8(2X,F6.1))
C
C
C     MATRIX PREPRATION
      GO TO 1102
  998 LEVLM=32
      LEVLR=32
      XIR=1
      XIM=1
      RMMIN=1
      RMIN=1
      DO 201 I=1,32
      DO 202 J=1,32
      M(I,J)=MI(I,J)
  202 CONTINUE
  201 CONTINUE
 1102 IF(OPTION.EQ.2)GO TO 999
      MM(1)=RMMIN
      DO 900 L=2,LEVLM
      LL=L-1
      MM(L)=MM(LL)+XIM
  900 CONTINUE
      RA(1)=RMIN
      DO 1100 L=2,LEVLR
      LL=L-1
 1100 RA(L)=RA(LL)+XIR
      I=0
   99 I=I+1
      IF(I.GT.LEVLR) GO TO 1153
      SUM(I)=0.
      DO 98 J=1,LEVLM
      SUM(I)=SUM(I)+M(I,J)
   98 CONTINUE
```

```
      GO TO 99
 1153 CONTINUE
 999  IF(OPTION.EQ.2)GO TO 997
 1151 L=1
      LB=8
 1152 IF(OPTION.EQ.1)GO TO 996
      WRITE(6,116)
      GO TO 1154
 996  WRITE(6,114)
      WRITE(6,115)FL
 1154 IF(OPTION.EQ.4)GO TO 604
      GO TO 1150
 604  WRITE(6,605)
 C
 C    THE NEXT STATEMENT DIFFERS FROM RAINF1
 C
 605  FORMAT(77X,'TOTAL',/1X,'START /**********************************TARGET
     **********************************',4X,'CYCLES',/)
      GO TO 2100
 1150 WRITE(6,600)
 600  FORMAT(77X,'TOTAL',/1X,'RANGE  /**********************************MEA
     *N**********************************',3X,'CYCLES',/)
 2100 WRITE(6,101)(MM(LL),LL=L,LB)
      DO 1300 I=1,LEVLR
      WRITE(6,102)RA(I),SUM(I)
      WRITE(6,103)(M(I,J),J=L,LB)
 1300 CONTINUE
      IF(LB.EQ.LEVLM)GO TO 1400
      L=L+8
      LB=LB+8
      IF(LB.GT.LEVLM)LB=LEVLM
      GO TO 1152
 1400 SUMM=0
      DO 1500 I=1,LEVLR
 1500 SUMM=SUMM+SUM(I)
      WRITE(6,104)SUMM
 101  FORMAT(12X,8(F6.1,2X))
 102  FORMAT(2X,F6.1,69X,I4)
 103  FORMAT('+',11X,8(I6,2X))
 104  FORMAT(//,5X,'TOTAL NO OF CYCLES=',3X,I5)
 997  CONTINUE
 107  FORMAT(15X,'RANGE',15X,'MEAN',15X,' MAX',15X,'   MIN')
 108  FORMAT(14X,F7.3,12X,F7.3,13X,F7.3,13X,F7.3)
 111  FORMAT('1',//15X,'MIN RANGE=',F8.3,//15X,'MAX RANGE=',F8.3,
     *//15X,'MIN MEAN=',F8.3,//15X,'MAX MEAN=',F8.3)
 116  FORMAT('1',//35X,'RAINFLOW CYCLES ')
 114  FORMAT('1',//20X,'RAINFLOW CYCLES FOR ORIGINAL HISTORY')
 115  FORMAT(/5X,'NO RANGE LESS THAN OR EQUAL TO FILTER LEVEL=',F7.3
     *,3X,'OCCUR IN FILTER HISTORY')
 210  FORMAT('1',//10X,'RANGES COUNTED AS CYCLES BY RAINFLOW CYCLE COUNT
     *ING METHOD.')
      STOP
      END
```

PROGRAM OUTPUT:

RANGES COUNTED AS CYCLES BY RAINFLOW CYCLE COUNTING METHOD.

RANGE	MEAN	MAX	MIN
4.000	1.000	3.000	-1.000
3.000	-0.500	1.000	-2.000
7.000	0.500	4.000	-3.000
9.000	0.500	5.000	-4.000

Example 2

The history of Fig. A.1 is again used for the rain-flow cycle counting example. Option 4 is used; therefore, the results are given in the form of a compact 32 by 32 matrix containing the peak and valley values of rain-flow cycles. Also, the directions of the cycles are stored in this matrix. Note that the history is converted using linear interpolation to have a minimum value of 1 and a maximum value of 32.

This option of the cycle counting program produces the output needed for use in a separate program (RECON2) that reconstructs a loading sequence having the same rain-flow cycles as the original sequence. The reconstructed history also produces rain-flow cycles with directions the same as in the original history. RECON2 is described in Appendix B.

In scaling to 32 levels, the level number, P_m, is related to any peak or valley value from the original history, P_0, as follows:

$$P_m = \frac{31 \times (P_0 - P_{0min})}{(P_{0max} - P_{0min})} + 1 \qquad (B-1)$$

except that P_m is rounded to the nearest integer. P_{0min} and P_{0max} are the lowest valley and highest peak, respectively, in the original history. For this example, $P_{0min} = -4$, and $P_{0max} = 5$. Cycle EF, which starts at $P_0 = -1$, and goes to $P_0 = 3$, has a starting level of 11 and a target level of 25.

Program Input

```
4
9
-2.,1.,-3.,5.,-1.,3.,-4.,4.,-2.
```

PROGRAM OUTPUT:

RAINFLOW CYCLES

START /**********TARGET**********

	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	TOTAL CYCLES
1.0	0	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	0	0	0	0
3.0	0	0	0	0	0	0	0	0	0
4.0	0	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0	0
6.0	0	0	0	0	0	0	0	0	0
7.0	0	0	0	0	0	0	0	0	0
8.0	0	0	0	0	0	0	0	0	1
9.0	0	0	0	0	0	0	0	0	0
10.0	0	0	0	0	0	0	0	0	0
11.0	0	0	0	0	0	0	0	0	1
12.0	0	0	0	0	0	0	0	0	0
13.0	0	0	0	0	0	0	0	0	0
14.0	0	0	0	0	0	0	0	0	0
15.0	0	0	0	0	0	0	0	0	0
16.0	0	0	0	0	0	0	0	0	0
17.0	0	0	0	0	0	0	0	0	0
18.0	0	0	0	0	0	0	0	0	0
19.0	0	0	0	0	0	0	0	0	0
20.0	0	0	0	0	0	0	0	0	0
21.0	0	0	0	0	0	0	0	0	0
22.0	0	0	0	0	0	0	0	0	0
23.0	0	0	0	0	0	0	0	0	0
24.0	0	0	0	0	0	0	0	0	0
25.0	0	0	0	0	0	0	0	0	0
26.0	0	0	0	0	0	0	0	0	0
27.0	0	0	0	0	0	0	0	0	0
28.0	0	0	0	0	0	0	0	0	0
29.0	0	0	0	1	0	0	0	0	1
30.0	0	0	0	0	0	0	0	0	0
31.0	0	0	0	0	0	0	0	0	0
32.0	1	0	0	0	0	0	0	0	1

RAINFLOW CYCLES
TOTAL CYCLES
START /**********TARGET**********
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0
24.0
25.0
26.0
27.0
28.0
29.0
30.0
31.0
32.0

RAINFLOW CYCLES

START /	TARGET								TOTAL CYCLES
	17.0	18.0	19.0	20.0	21.0	22.0	23.0	24.0	
1.0	0	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	0	0	0	0
3.0	0	0	0	0	0	0	0	0	0
4.0	0	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0	0
6.0	0	0	0	0	0	0	0	0	0
7.0	0	0	0	0	0	0	0	0	0
8.0	0	1	0	0	0	0	0	0	1
9.0	0	0	0	0	0	0	0	0	0
10.0	0	0	0	0	0	0	0	0	0
11.0	0	0	0	0	0	0	0	0	1
12.0	0	0	0	0	0	0	0	0	0
13.0	0	0	0	0	0	0	0	0	0
14.0	0	0	0	0	0	0	0	0	0
15.0	0	0	0	0	0	0	0	0	0
16.0	0	0	0	0	0	0	0	0	0
17.0	0	0	0	0	0	0	0	0	0
18.0	0	0	0	0	0	0	0	0	0
19.0	0	0	0	0	0	0	0	0	0
20.0	0	0	0	0	0	0	0	0	0
21.0	0	0	0	0	0	0	0	0	0
22.0	0	0	0	0	0	0	0	0	0
23.0	0	0	0	0	0	0	0	0	0
24.0	0	0	0	0	0	0	0	0	0
25.0	0	0	0	0	0	0	0	0	0
26.0	0	0	0	0	0	0	0	0	0
27.0	0	0	0	0	0	0	0	0	0
28.0	0	0	0	0	0	0	0	0	0
29.0	0	0	0	0	0	0	0	0	1
30.0	0	0	0	0	0	0	0	0	0
31.0	0	0	0	0	0	0	0	0	0
32.0	0	0	0	0	0	0	0	0	1

RAINFLOW CYCLES

START /	TARGET								TOTAL CYCLES
	25.0	26.0	27.0	28.0	29.0	30.0	31.0	32.0	
1.0	0	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	0	0	0	0
3.0	0	0	0	0	0	0	0	0	0
4.0	0	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0	0
6.0	0	0	0	0	0	0	0	0	0
7.0	0	0	0	0	0	0	0	0	0
8.0	0	0	0	0	0	0	0	0	1
9.0	0	0	0	0	0	0	0	0	0
10.0	0	0	0	0	0	0	0	0	0
11.0	1	0	0	0	0	0	0	0	1
12.0	0	0	0	0	0	0	0	0	0
13.0	0	0	0	0	0	0	0	0	0
14.0	0	0	0	0	0	0	0	0	0
15.0	0	0	0	0	0	0	0	0	0
16.0	0	0	0	0	0	0	0	0	0
17.0	0	0	0	0	0	0	0	0	0
18.0	0	0	0	0	0	0	0	0	0
19.0	0	0	0	0	0	0	0	0	0
20.0	0	0	0	0	0	0	0	0	0
21.0	0	0	0	0	0	0	0	0	0
22.0	0	0	0	0	0	0	0	0	0
23.0	0	0	0	0	0	0	0	0	0
24.0	0	0	0	0	0	0	0	0	0
25.0	0	0	0	0	0	0	0	0	0
26.0	0	0	0	0	0	0	0	0	0
27.0	0	0	0	0	0	0	0	0	0
28.0	0	0	0	0	0	0	0	0	0
29.0	0	0	0	0	0	0	0	0	1
30.0	0	0	0	0	0	0	0	0	0
31.0	0	0	0	0	0	0	0	0	0
32.0	0	0	0	0	0	0	0	0	1

TOTAL NO OF CYCLES= 4

Appendix B. Computer Program RECON2 for Two Dimensional Rain-flow Reconstruction

A computer program for reconstructing a load history from the result of rain-flow cycle counting in matrix form is provided. Note that the directions of the cycles are indicated by the position in the matrix. The numbers in the matrix , that is, the " matrix elements", are the numbers of cycles at the various peak-valley-direction combinations. The program takes the results of rain-flow cycle counting in the form of such a matrix and uses this information to reconstruct a load history. The resulting history has the same rain-flow matrix as the original history but is not generally identical as to ordering of the cycles. The program logic is explained in detail in the main text of this report, with a few examples.

Note that in its simplest application, the program places all cycles having the same peak, valley and direction in a single location in the history. If a more irregular version is desired, then the number of cycles for a given peak-valley-direction combination is divided into n groups. In this program n is called NOC. This is done only if the number of cycles is greater than a specific value (NP). The group size is then rounded down to the nearest whole number, and this many cycles are placed in the first location, the same in the second location, etc., except that the number in the last location includes the rounded down number plus all of the

residual from rounding. If NOC=NP=1, then cycles are placed individually, but this choice may be costly due to computer run time for lengthy histories.

Definition of Input Data

Data line 1: NP= Largest matrix element which is placed in a single location. If a given row and column has this value or less, then all of these cycles are placed in one location.

Data line 2: NOC= The number of different locations in which cycles from a matrix element are placed. If the number of cycles for a given row and column is greater than NP, then these cycles are placed in this many randomly chosen locations.

Data line 3: AB(,)= The rain-flow matrix elements themselves, where directions of rain-flow cycles are considered.

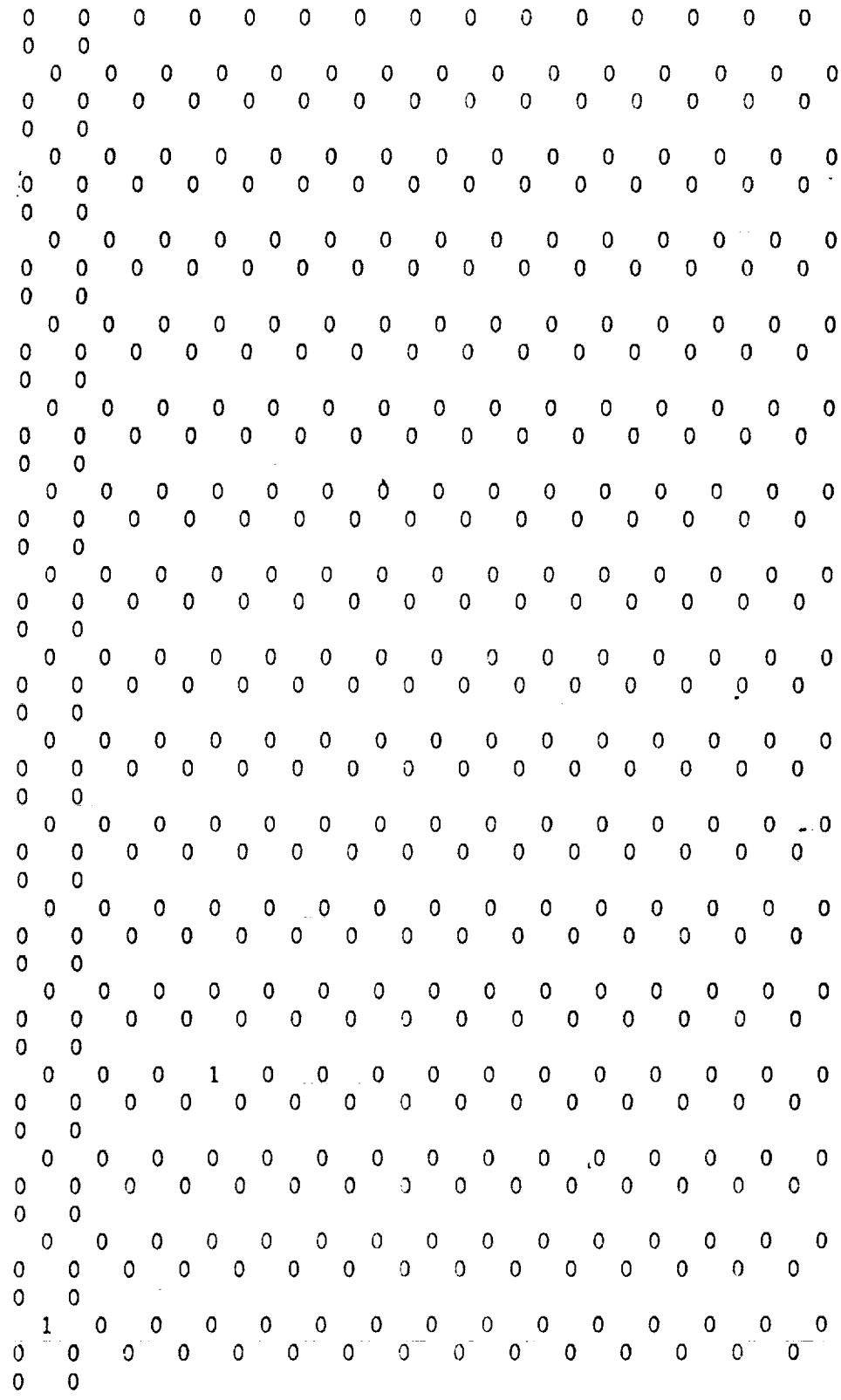
Note that the rain-flow peak-valley matrix AB(,) is read into a two dimensional array, starting with the first row, moving left to right, and completing each row before going to the next row. This data can be obtained using the RAINF2 computer program (OPTION=4). The standard size of the matrix for this program is 32 by 32. However, if a different size is desired, say 16 by 16, then the LEVEL value inside the program should be changed from 32 to 16. The columns are starting levels (peaks or valleys) of cycles, and the rows are target levels.

Example 1

A history containing 9 peak-valley points is used with NP = NOC = 1. Figure B.1 shows this history. The input is the rain-flow matrix of this history, with directions of rain-flow cycles considered, as obtained using the RAINF2 computer program (Appendix A, OPTION 4). Note that the reconstructed history gives the same rain-flow peak-valley matrix (with directions considered) as the original history; however, it does not have the same sequence as the original history. Figure B.1 compares the original and reconstructed histories. Note that the user must convert the history obtained (min = 1, max = 32) using linear interpolation to get a history compatible with the original history. The entire program listing and program input and output for this example follow.

PROGRAM INPUT

```
1
1
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```



PROGRAM LISTING

```
C       2-D RAIN-FLOW RECONSTRUCTION PROGRAM (RECON2)
C
C
C       THIS PROGRAM CONSTRUCT A HISTORY USING THE R-F MATRIX
C       (DIRECTIONS OF THE RAIN-FLOW CYCLES ARE CONSIDERD)
C
C       INPUT:
C
C       DATA LINE 1.  NP=A GIVEN ROW AND COLUMN HAS THIS VALUE OR
C                     LESS, THEN ALL PLACED IN ONE LOCATION.
C       DATA LINE 2.  NOC=THE NUMBER OF CYCLES FOR EACH ROW AND
C                     COLUMN, IF GREATER THAN NP, IS PLACED
C                     IN THIS MANY RANDOMLY CHOSEN LOCATIONS.
C       DATA LINE 3.  AB( , )=THE RAIN-FLOW MATRIX.
C                   (CYCLES DIRECTIONS CONSIDERED.)
C       NOTE: SIMPLEST FORM OF A RAIN-FLOW RECONSTRUCTED HISTORY
C           IS OBTAINED BY SETTING NP=100000
C
      INTEGER A(32,32),KA(15024),P(15000),PP(15000),IL(15000)
     &, PE(15000),DUM,DUM1,DUM2,JL(15000),AB(32,32)
      DIMENSION R(20000)
C
C       LEVEL IS THE NO. OF COLUMNS OR ROWS.
C
      LEVEL=32
C
      READ(5,*)NP
      READ(5,*)NOC
      DO 90 I=1,LEVEL
      READ(5,*)(AB(I,J),J=1,LEVEL)
90     CONTINUE
C
C       RANDOM NUMBER GENERATION
C
      DO 91 I=1,20000
      CALL TR(U)
      R(I)=U
91     CONTINUE
C
C       REARRANGING THE R-F MATRIX
C
      ISS=0
      IB=1
      ILA=LEVEL
      IL(1)=1
      JL(1)=LEVEL
      P(1)=1
      P(2)=LEVEL
      PP(1)=1
      PP(2)=LEVEL
      IL(2)=1
      IL(2)=LEVEL
      AB(LEVEL,1)=0
      AB(1,LEVEL)=1
```

```
      MTOTAL=2
      IS=1
      ILAST=LEVEL
      ICOUNT=0
      ISUM=3
      GO TO 3
4     J=1
      ISUM=ISUM+(2*IS)-ICOUNT
3     KI=ILAST-1
      IF(KI.LT.1)GO TO 5
      IL(ISUM)=ILAST-1
      ILAST=ILAST-1
      J=1
      JL(ISUM)=J
2     ILL=IL(ISUM)
      ILL=ILL+1
      IF(ILL.GT.LEVEL)GO TO 1
      ISUM=ISUM+1
      J=J+1
      IL(ISUM)=ILL
      JL(ISUM)=J
      GO TO 2
1     IS=IS+1
      ICOUNT=ICOUNT+1
      GO TO 4
C
C
C
5     IS=1
      ICOUNT=0
      ILAST=1
      ISUM=2
      J=LEVEL+1
      ILAST=1
      GO TO 7
6     ILAST=1
      ISUM=ISUM+(2*IS)-ICOUNT
7     JL(ISUM)=J-1
      IF(J.LT.1)GO TO 10
      J=J-1
      IL(ISUM)=ILAST
8     JLL=JL(ISUM)
      JLL=JLL+1
      IF(JLL.GT.LEVEL)GO TO 9
      ISUM=ISUM+1
      ILAST=ILAST+1
      JL(ISUM)=JLL
      IL(ISUM)=ILAST
      GO TO 8
9     IS=IS+1
      ICOUNT=ICOUNT+1
      GO TO 6
C
C     RECONSTRUCTION BEGINS
C
10    ITOTAL=LEVEL*LEVEL
```

```
      KCOUNT=1
      ISUM=2
9999    ISUM=ISUM+1
      KCOUNT=1
      IF(ISUM.GT.ITOTAL)GO TO 999
      L=0
      ILAST=IL(ISUM)
      J=JL(ISUM)
      IF(AB(ILAST,J).EQ.0)GO TO 9999
      NOC2=10000
      INUM=0
99     IF(INUM.EQ.NOC2)GO TO 9999
      IF(AB(ILAST,J).GT.NP) THEN
      INUM=INUM+1
      KNUM=AB(ILAST,J)/NOC
      A(ILAST,J)=KNUM
      NOC1=KNUM*NOC
      IF(NOC1.NE.AB(ILAST,J))THEN
      IF(INUM.EQ.NOC)A(ILAST,J)=KNUM+(AB(ILAST,J)-NOC1)
      NOC2=NOC
      ELSE
      NOC2=NOC
      END IF
      ELSE
      A(ILAST,J)=AB(ILAST,J)
       NOC2=INUM
      END IF
      IF(ILAST.LT.J)GO TO 300
      JM1=J-1
      ISU=ISUM-1
      DO 100 LI=1,ISU
      IPU=ILA-ILAST+1
      IDUM=IL(LI)
      JDUM=JL(LI)
      IF(A(IDUM,JDUM).EQ.0)GO TO 100
      IF(IL(LI).LT.JL(LI))THEN
      IF(IL(LI).LT.J.AND.JL(LI).GT.ILAST)GO TO 102
      GO TO 100
      END IF
      IF(IL(LI).GT.JL(LI))THEN
      IF(IL(LI).GT.ILAST.AND.JL(LI).LT.J)GO TO 102
      GO TO 100
      END IF
102     L=L+1
      KA(LI)=L
100     CONTINUE
710     IR=L*R(IB)
      KCOUNT=KCOUNT+1
      IB=IB+1
      NRAN=IR+1
C
C
C
      IF(KCOUNT.EQ.5)GO TO 711
776     IF(NRAN.EQ.1)THEN
```

```
711     KAM=0
98      DO 600 IA=1,MTOTAL
        IF(P(IA).EQ.1.AND.IA.EQ.1)GO TO 700
        IF(MTOTAL.LE.2)THEN
        IF(P(IA).EQ.1)GO TO 700
        END IF
        IF(P(IA).EQ.1.AND.P(IA+1).GT.LEVEL)THEN
        IF(P(IA-1).GT.LEVEL)THEN
        IF(P(IA-1).GT.116)GO TO 700
        IAX=IA-1
        IAMX=IAX
7009    IAMX=IAMX-1
        IF(IAMX.LT.1)GO TO 600
        IF(P(IAMX).EQ.P(IA-1))GO TO 700
        GO TO 7009
        END IF
        END IF
600     CONTINUE
700     IA3=IA+1
721     IF(P(IA3).GT.LEVEL)THEN
        IA3=IA3+1
        GO TO 721
        END IF
762     IF(P(IA3).LT.ILAST)THEN
        IA4=IA3-1
        IT=P(IA4)
        IA3=IA3+1
        DO 722 LE=IA3,MTOTAL
        IF(P(LE).EQ.IT)GO TO 723
722     CONTINUE
723     IA1=LE
        IA=LE
        LEE=LE+1
761     IF(LEE.GT.MTOTAL)GO TO 777
        IF(P(LEE).GT.LEVEL)THEN
        LEE=LEE+1
        GO TO 761
        END IF
        IA3=LEE
        GO TO 762
        END IF
777     IA1=IA
        ISS=ISS+A(ILAST,J)
724     KK=0
713     JI=IA
        DO 714 IM=1,JI
714     PP(IM)=P(IM)
        LLL=JI+1
        PP(LLL)=(ILAST*100)+J
        JK=JI+(A(ILAST,J)*2)+1
        JE=JI+2
        DO 701 IMM=JE,JK,2
        PP(IMM)=ILAST
        IM1=IMM+1
        PP(IM1)=J
701     CONTINUE
```

```
      MTOTAL=MTOTAL+(A(ILAST,J)*2)+2
      IM1=IM1+1
      PP(IM1)=(ILAST*100)+J
      IM1=IM1+1
      J11=JI
      DO 702 IB=IM1,MTOTAL
      J11=J11+1
      PP(IB)=P(J11)
702    CONTINUE
      DO 703 IN=1,MTOTAL
703    P(IN)=PP(IN)
      GO TO 99
      END IF
      IF(KCOUNT.EQ.5)GO TO 99
C
C
C
      K=1
      DO 200 LI=1,ISU
      IPU=ILA-ILAST+1
      IDUM=IL(LI)
      JDUM=JL(LI)
      IF(A(IDUM,JDUM).EQ.0)GO TO 200
      IF(IL(LI).LT.JL(LI))THEN
      IF(IL(LI).LT.J.AND.JL(LI).GT.ILAST)GO TO 201
      GO TO 200
      END IF
      IF(IL(LI).GT.JL(LI))THEN
      IF(IL(LI).GT.ILAST.AND.JL(LI).LT.J)GO TO 201
      GO TO 200
      END IF
201    K=K+1
      KA(I)=K
      IF(NRAN.EQ.K)GO TO 500
200    CONTINUE
500    I=IL(LI)
      II=JL(LI)
      ISS=A(ILAST,J)+ISS
      IF(I.EQ.1.AND.II.EQ.LEVEL)THEN
      NRAN=1
      GO TO 776
      END IF
      IF(I.LT.II)THEN
      IF(I.GE.J)GO TO 710
      END IF
      CALL SORT (I,II,ILAST,J,LK,KC,P,PP,MTOTAL,LEVEL,A)
      IF(I.EQ.1.AND.II.EQ.LEVEL)GO TO 98
      GO TO 99
C
C
C
300    ISU=ISUM-1
      DO 800 LB=1,ISU
      IPU=ILA-ILAST+1
      IDUM=IL(LB)
      JDUM=JL(LB)
```

```
      IF(A(IDUM,JDUM).EQ.0)GO TO 800
      IF(IL(LB).GT.JL(LB))THEN
      IF(IL(LB).GT.J.AND.JL(LB).LT.ILAST)GO TO 801
      GO TO 800
      ELSE
      IF(IL(LB).LT.ILAST.AND.JL(LB).GT.J)GO TO 801
      GO TO 800
      END IF
801     L=L+1
      KA(I)=L
800    CONTINUE
      KCOUNT=1
1711    IR=L*R(IB)
      KCOUNT=KCOUNT+1
      IB=IB+1
      NRAN=IR+1
      IF(KCOUNT.EQ.5)GO TO 1712
C
C
C
      IF(NRAN.EQ.1)THEN
1712    NAM=0
101     IC=A(ILAST,J)
      DO 1901 IA=1,MTOTAL
      IF(P(IA).EQ.1.AND.IA.EQ.1)GO TO 1916
      IF(MTOTAL.LE.2)THEN
      IF(P(IA).EQ.1)GO TO 1916
      END IF
      IF(P(IA).EQ.1.AND.P(IA+1).GT.LEVEL)THEN
      IF(P(IA-1).GT.LEVEL)THEN
      IF(P(IA-1).GT.116)GO TO 1916
      IAX=IA-1
      IAMX=IAX
7010    IAMX=IAMX-1
      IF(IAMX.LT.1)GO TO 1901
      IF(P(IAMX).EQ.P(IA-1))GO TO 1916
      GO TO 7010
      END IF
      END IF
1901    CONTINUE
1916    IF(IA.EQ.1)GO TO 2103
      IAM1=IA-1
      PIAM1=P(IAM1)
      IAM2=IA-1
2101    IAM2=IAM2-1
      IF(P(IAM2).EQ.PIAM1)GO TO 2100
      GO TO 2101
2100    IAM1=IAM2+1
2108    IF(P(IAM1).LT.ILAST)THEN
      IA1=IAM2
      IA=IA1
      IAM3=IAM2-1
2105    IF(IAM3.LE.0)GO TO 2103
      IF(P(IAM3).GT.LEVEL)GO TO 2104
      IAM3=IAM3-1
```

```
        GO TO 2105
2104      IAM4=IAM3-1
2107      IF(IAM4.LE.0)GO TO 2103
        IF(P(IAM4).EQ.P(IAM3))GO TO 2106
        IAM4=IAM4-1
         GO TO 2107
2106      IAM1=IAM4+1
        IAM2=IAM4
        GO TO 2108
        END IF
2103      IA1=IA
2102      JI=IA
        JIM1=JI-1
        IF(JI.EQ.1)GO TO 1909
        DO 1910 I=1,JIM1
1910      PP(I)=P(I)
1909      JK=JI+(IC*2)-1
        PP(JI)=(ILAST*100)+J
        JE=JI+1
        DO 1911 I=JE,JK,2
        PP(I)=ILAST
        IP=I+1
        PP(IP)=J
1911      CONTINUE
        MTOTAL=MTOTAL+(IC*2)+2
        IP=IP+1
        PP(IP)=(ILAST*100)+J
        IP=IP+1
        J11=JI-1
        DO 1912 IB=IP,MTOTAL
        J11=J11+1
        PP(IB)=P(J11)
1912      CONTINUE
        DO 1914 KB=1,MTOTAL
1914      P(KB)=PP(KB)
        GO TO 99
        END IF
        IF(KCOUNT.EQ.5)GO TO 99
C
C
C
        K=0
        DO 900 LB=1,ISU
        IPU=ILA-ILAST+1
        IDUM=IL(LB)
        JDUM=JL(LB)
        IF(A(IDUM,JDUM).EQ.0)GO TO 900
        IF(IL(LB).GT.JL(LB))THEN
        IF(IL(LB).GT.J.AND.JL(LB).LT.ILAST)GO TO 901
        GO TO 900
        ELSE
        IF(IL(LB).LT.ILAST.AND.JL(LB).GT.J)GO TO 901
        GO TO 900
        END IF
901       K=K+1
        KA(I)=K
```

```
      IF(NRAN.EQ.K)GO TO 1200
900     CONTINUE
1200    I=IL(LB)
      II=JL(LB)
      ISS=ISS+A(ILAST,J)
      CALL SORT (I,II,ILAST,J,LK,KC,P,PP,MTOTAL,LEVEL,A)
      IF(I.EQ.1.AND.II.EQ.LEVEL)GO TO 101
      GO TO 99
999     MT=1
      PE(1)=LEVEL
      DO 1001 I=1,MTOTAL
      IF(PP(I).GT.LEVEL)GO TO 1001
      MT=MT+1
      PE(MT)=PP(I)
1001    CONTINUE
      WRITE(6,*)MT
      WRITE(6,1300)(PE(I),I=1,MT)
1300    FORMAT(11(I5,1X))
      STOP
      END
C
C
      SUBROUTINE TR(X)
C
C       THIS SUBROUTINE GENERATES RANDOM NUMBER BETWEEN 0 AND 1.
C       (APPLIED TIME SERIES ANALYSIS BY OTNES.)
C
      DATA L/783637/
      L=125*L
      L=L-(L/2796203)*2796203
      X=FLOAT(L)/2796202.
      RETURN
      END
C
      SUBROUTINE SORT(I,II,ILAST,J,LK,KC,P,PP,MTOTAL,LEVEL,A)
C
C       THIS SUBROUTINE PUT THE INSERTING CYCLES INTO A PARTIALLY
C       RECONSTRUCTED HISTORY.
C
C       ILAST,J= INSERTING PEAK AND VALLEY , OR INSERTING VALLEY
C       AND PEAK.
C       I,II= RECEIVING PEAK AND VALLEY , OR RECEIVING VALLEY
C       AND PEAK.
C
      INTEGER PP(15000),A(32,32),P(15000)
      IF(I.EQ.1.AND.II.EQ.LEVEL)GO TO 200
      LEVEL=32
      IP=(I*100)+II
C
C
      IF(I.GT.II)THEN
C
      IF(ILAST.GT.J)THEN
      DO 1 L=1,MTOTAL
      LP1=L+1
1     IF(P(L).EQ.II.AND.P(LP1).EQ.IP)GO TO 2
```

```
2       LPL=LP1
500     LPL=LPL+1
      IF(P(LPL).GT.LEVEL)GO TO 500
      LPL1=LPL
501     LPL1=LPL1+1
      IF(P(LPL1).GT.LEVEL)GO TO 501
      IF(P(LPL).LT.ILAST.AND.P(LPL1).LT.J)THEN
      I=1
      II=LEVEL
      GO TO 200
      END IF
      DO 3 LK=LP1,MTOTAL
3     IF(P(LK).LE.LEVEL)GO TO 4
4     IF(P(LK).GT.ILAST)GO TO 45
      LK1=LK-1
48    DO 410 LK2=LK,MTOTAL
410   IF(P(LK2).EQ.P(LK1))GO TO 42
42    LK3=LK2
      LK3=LK2+1
      IF(P(LK3).LE.LEVEL)GO TO 43
      LK4=LK3
      LK4=LK4+1
      IF(P(LK4).GT.ILAST)GO TO 44
      LK1=LK3
      LK=LK4
      GO TO 48
43    IF(P(LK3).EQ.LEVEL)THEN
      LP1=LK3-1
      END IF
      GO TO 45
44    LP1=LK3-1
45    DO 5 LI=1,LP1
5     PP(LI)=P(LI)
      IPM1=LP1+1
      PP(IPM1)=(ILAST*100)+J
      IPM2=IPM1+1
      K=A(ILAST,J)
      KK=1
      L=IPM2
6     PP(L)=ILAST
      L=L+1
      PP(L)=J
      IF(K.EQ.KK)GO TO 8
      KK=KK+1
      L=L+1
      GO TO 6
8     L=L+1
      PP(L)=(ILAST*100)+J
      L=L+1
      L2=IPM1
      DO 9 IK=L2.MTOTAL
      PP(L)=P(IK)
      L=L+1
9     CONTINUE
      MTOTAL=L-1
```

```
      ELSE
C
C
      DO 10 JI=2,MTOTAL
      JI1=JI-1
10     IF(P(JI).EQ.I.AND.P(JI1).EQ.IP)GO TO 11
11     JIP1=JI+1
      DO 12 LE=JIP1,MTOTAL
12     IF(P(LE).EQ.P(JI1))GO TO 13
13     LE=LE-2
      IF(P(LE).GT.LEVEL)GO TO 14
      JI=LE
      GO TO 16
14     LEE=LE
140    LEE=LEE-1
      IF(P(LEE).EQ.P(LE))GO TO 150
      GO TO 140
150    LE1=LEE+1
      IF(P(LE1).GT.ILAST)THEN
      DO 17 LC=LE1,MTOTAL
      LC1=LC+1
17     IF(P(LC).EQ.P(LE))GO TO 18
18     JI=LC
      ELSE
220    LE2=LEE-1
      IF(P(LE2).LE.LEVEL)GO TO 190
      LE3=LE2
210    LE3=LE3-1
      IF(P(LE3).EQ.P(LE2))GO TO 180
      GO TO 210
180    LE4=LE3+1
      IF(P(LE4).LT.ILAST)THEN
      LEE=LE3
      GO TO 220
      ELSE
      JI=LE2
      END IF
      GO TO 16
190    IF(P(LE2).EQ.I)THEN
      JI=LE2
      END IF
      END IF
16     DO 19 IK=1,JI
19     PP(IK)=P(IK)
      ICC=A(ILAST,J)*2
      IK=0
      IC=JI
      IC=IC+1
      PP(IC)=(ILAST*100)+J
20     IC=IC+1
      IK=IK+2
      PP(IC)=ILAST
      IC=IC+1
      PP(IC)=J
      IF(IK.EQ.ICC)GO TO 21
```

```
      GO TO 20
21     MTOTAL=MTOTAL+ICC+2
      IC=IC+1
      PP(IC)=(ILAST*100)+J
      IC=IC+1
      IA=JI
      DO 22 ID=IC,MTOTAL
      IA=IA+1
22     PP(ID)=P(IA)
      END IF
      END IF
      IF(I.GT.II)GO TO 2000
C
C
      IF(ILAST.LT.J)THEN
      DO 23 KI=1,MTOTAL
      KI1=KI+1
23     IF(P(KI).EQ.II.AND.P(KI1).EQ.IP)GO TO 24
24     DO 25 KC=KI1,MTOTAL
25     IF(P(KC).LE.LEVEL)GO TO 250
250    IF(P(KC).LT.ILAST)GO TO 231
      KCM1=KC-1
      DO 233 KB=KC,MTOTAL
233    IF(P(KB).EQ.P(KCM1))GO TO 234
234     KI1=KB
      GO TO 24
231    KTRY=KI1+1
      IF(KC.EQ.KTRY)THEN
      LIP=KC-1
      ELSE
      LIP=KC-2
      END IF
      IP1=LIP+1
      DO 26 KB=1,LIP
26     PP(KB)=P(KB)
      PP(IP1)=(ILAST*100)+J
      KK=1
      K=A(ILAST,J)
      L=IP1+1
27     PP(L)=ILAST
      L=L+1
      PP(L)=J
      IF(K.EQ.KK)GO TO 28
      KK=KK+1
      L=L+1
      GO TO 27
28     L=L+1
      PP(L)=(ILAST*100)+J
      L=L+1
      L2=IP1
      DO 29 IK=L2,MTOTAL
      PP(L)=P(IK)
      L=L+1
29     CONTINUE
      MTOTAL=L-1
```

```
      ELSE
C
C
       DO 30 NI=2,MTOTAL
       NI1=NI-1
30      IF(P(NI1).EQ.IP.AND.P(NI).EQ.I)GO TO 31
31      DO 32 NC=NI,MTOTAL
       NC1=NC+1
32      IF(P(NC).EQ.II.AND.P(NC1).EQ.IP)GO TO 333
333     NC1=NC1-2
       GO TO 41
33      NC1=NC-2
41      IF(P(NC1).LE.LEVEL)GO TO 300
       NC2=NC1
320     NC2=NC2-1
       IF(P(NC2).EQ.P(NC1))GO TO 310
       GO TO 320
310     NC3=NC2+1
       IF(P(NC3).LT.ILAST)GO TO 330
       NC=NC3
       GO TO 33
300     IF(P(NC1).EQ.I)THEN
       JI=NC1
       END IF
       GO TO 380
330     JI=NC1
380     DO 37 IE=1,JI
37      PP(IE)=P(IE)
       PP(IE)=(ILAST*100)+J
       K=A(ILAST,J)
       KK=1
       L=IE+1
39      PP(L)=ILAST
       L=L+1
       PP(L)=J
       IF(K.EQ.KK)GO TO 38
       KK=KK+1
       L=L+1
       GO TO 39
38      L=L+1
       PP(L)=(ILAST*100)+J
       L=L+1
       L2=JI+1
       DO 40 IS=L2,MTOTAL
       PP(L)=P(IS)
       L=L+1
40      CONTINUE
       MTOTAL=L-1
       END IF
2000    DO 1000 K=1,MTOTAL
1000    P(K)=PP(K)
200     RETURN
       END
```

GENERATED HISTORY

32 11 25 8 18 1 29 4 32

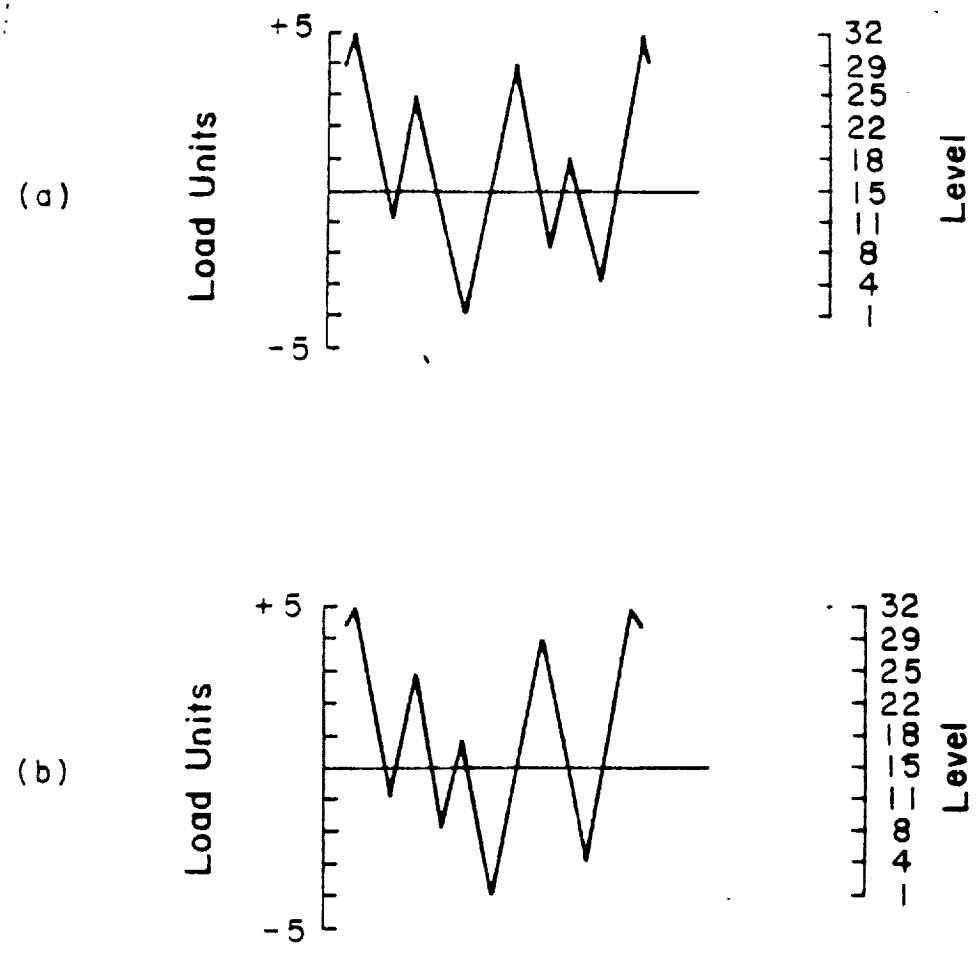


Figure B.1. Comparison of original (a), and reconstructed (b) histories.

Example 2

A history (Maneuver History) containing 1021 peak-valley points is used. This history is explained in detail in the main text of this report. The input is the rain-flow matrix of this history with directions of rain-flow cycles considered, as obtained using the RAINF2 computer program (Appendix A, OPTION 4). Note that the reconstructed history gives the same rain-flow peak-valley matrix as the original history, and also the directions of the cycles are preserved. Note that as mentioned before, the user must convert the history obtained (min = 1, max = 32) using linear interpolation to get a history on a scale compatible with the original history. The program input and output are attached. Cycles from each matrix element are placed in two randomly chosen locations (NOC = 2), provided these are more than eight (NP = 8). Figure B.2 shows the original and the reconstructed histories.

Input for the Reconstruction Program

```
8
2
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
0  1
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  2  0  1  0  0  0  0  1  0  1  0  0  1  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  4  1  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  4  4  2  0  3  0  1  1  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  2  7  4  2  0  2  0  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  4  3  2  1  2  1  0  0  0  0  1
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  2  5  3  1  1  0  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  3  3  1  3  3  0  0  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  1  7  2  6  0  1  0  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  3  5  4  4  2  1  0
0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
 0   0   0   0   0   0   0   0   0   1   9   5   2   1   0
 0   0
   0   0   0   0   0   0   1   2   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0   0   0   3   6   3   1   0
 0   0
   0   0   0   0   0   0   1   6   2   0   0   0   0   0   0
 0   0   0   0   0   0   0   0   0   0   0   3   8   2   1
 2   0
   0   0   0   0   0   1   1   0   2   6   0   0   0   0   0
 0   0   0   0   0   0   0   0   0   0   0   0   5  12  12
 2   1
   0   0   0   0   0   0   1   0   3   2   1   0   0   0   0
 0   0   0   0   0   0   0   0   0   0   0   0   0   5  15
10   4
   0   0   0   0   0   0   0   0   1   5   3   4   0   0   0
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   4
 6   0
   0   0   0   0   0   0   0   0   1   3   4   3   1   0   0
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   1   4   3   2   6   5   4   0
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   0   0   3   0   3   3   2   4
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   1   0   0   0   1   0   0   0   4   4
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   0   1   0   0   1   1   2   3
12   2   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   0   0   0   0   1   2   3   2
 7  10   4   0   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   1   0   0   0   0   0   1   0   3
 4   3  14   4   0   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1
 0   0   6  13   5   0   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
 0   0   1   7  10   7   0   0   0   0   0   0   0   0   0
 0   0
   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0
 0   0   2   4   6   6   0   0   0   0   0   0   0   0   0
 0   0
   1   0   0   0   1   0   0   0   0   0   0   1   0   0   0
 0   0   0   1   0   1   0   0   0   0   0   0   0   0   0
 0   0
```

GENERATED HISTORY

32	21	31	21	31	21	31	21	31	21	31
21	31	20	32	20	32	20	32	20	32	19
32	20	30	20	30	20	30	20	30	20	30
20	30	20	30	20	30	6	17	7	18	7
23	10	23	10	23	10	26	16	26	16	26
16	26	16	26	16	26	16	27	15	27	17
26	17	26	17	26	15	27	12	29	14	25
14	25	12	26	15	26	15	26	15	28	17
28	17	28	17	29	15	27	15	27	15	27
15	27	11	21	12	21	12	21	12	21	12
30	20	30	20	30	20	30	20	30	20	31
20	31	20	31	20	31	20	31	20	31	20
31	19	31	19	31	19	31	19	32	20	31
20	31	20	31	20	31	20	31	21	30	21
30	21	30	21	30	20	31	20	31	20	31
20	31	20	31	20	30	20	30	20	30	20
30	20	30	20	30	20	30	19	31	19	31
19	30	19	30	19	30	19	30	19	30	19
30	20	29	20	29	20	29	20	29	20	29
19	30	19	30	19	30	19	30	19	30	19
30	18	31	18	31	17	27	17	27	17	27
17	27	17	27	17	27	16	26	16	26	16
26	16	26	16	26	16	26	16	26	16	26
16	26	15	28	15	27	18	27	18	27	18
27	18	28	16	25	10	19	10	19	7	22
11	22	11	22	11	22	11	26	12	23	12
23	12	23	12	21	12	21	12	25	15	24
15	24	15	24	9	20	10	20	10	28	13
29	18	28	18	28	18	28	18	28	18	28
18	28	18	28	18	28	18	27	18	27	18
27	16	28	16	28	8	18	8	18	8	18
8	18	7	24	6	17	8	17	8	20	11
22	13	23	12	23	12	23	12	23	12	23
12	23	12	24	14	24	14	24	12	24	12
24	12	24	10	24	10	24	10	25	10	29
18	29	18	29	18	29	18	29	18	29	19
28	19	28	19	28	19	28	19	28	18	30
15	29	14	28	14	24	14	24	14	24	14
24	14	24	14	24	14	24	9	21	10	21
10	21	10	21	10	21	10	24	13	23	13
23	13	23	11	22	11	22	10	21	10	21
10	21	10	21	10	20	10	20	10	20	10
20	10	20	10	20	10	20	9	22	9	22
9	22	7	23	9	23	9	23	9	23	9
22	12	22	12	22	12	26	11	24	11	24
10	24	10	24	10	22	10	22	9	19	9
19	9	19	9	19	7	17	7	17	1	18
9	18	9	18	8	18	8	18	8	18	8

18	8	18	8	19	8	17	8	17	8	17
7	19	6	23	11	23	11	23	9	20	9
20	8	24	13	24	13	24	13	26	17	26
17	26	16	26	16	26	16	26	16	26	16
26	16	27	16	27	16	27	16	27	16	27
16	27	16	27	16	27	14	27	14	27	14
28	19	28	19	28	19	28	19	28	18	28
18	28	18	28	18	28	18	28	18	28	18
29	19	29	19	29	19	29	19	29	19	29
19	29	15	28	16	28	16	28	16	28	16
29	19	29	19	29	19	29	19	29	19	29
19	29	19	30	21	30	21	30	21	30	21
30	21	30	21	30	21	30	20	30	20	30
20	30	20	30	20	30	19	30	19	30	19
30	19	30	19	30	19	30	19	30	18	29
20	29	20	29	20	29	20	29	20	31	21
31	21	31	21	31	21	31	21	31	21	31
18	31	18	31	4	19	9	19	9	22	9
19	10	19	10	19	10	19	10	19	10	19
10	25	7	19	9	18	9	18	9	18	9
18	6	22	10	22	10	22	10	28	14	26
14	26	14	26	14	26	14	26	14	26	11
23	7	32	21	32	19	32	19	29	19	29
19	29	19	29	19	29	19	29	12	23	13
23	13	23	13	23	13	23	13	28	15	28
15	28	15	27	17	27	17	27	17	27	17
27	17	27	17	27	17	27	17	27	17	27
17	32	19	29	19	29	19	29	19	29	19
29	19	29	17	28	17	28	17	28	13	22
13	22	13	22	11	25	12	24	5	20	8
19	7	20	9	20	9	20	9	26	15	25
15	25	15	25	15	25	15	25	12	23	14
23	14	23	14	23	14	31	18	30	18	29
18	29	11	26	13	30	16	28	18	28	18
28	18	28	18	28	18	28	18	28	18	29
16	27	16	27	16	27	16	27	16	27	13
25	13	25	13	25	13	24	12	22	12	22
12	22	12	22	12	22	11	21	11	21	11
21	11	20	11	20	11	20	11	20	9	25
7	21	9	21	11	21	11	21	11	27	13
27	15	26	15	26	15	26	15	26	13	25
15	25	15	25	15	25	15	30	17	29	13
25	14	25	14	25	14	25	14	31	13	26
13	26	13	24	15	24	15	24	15	24	15
26	14	23	11	26	14	26	14	32		

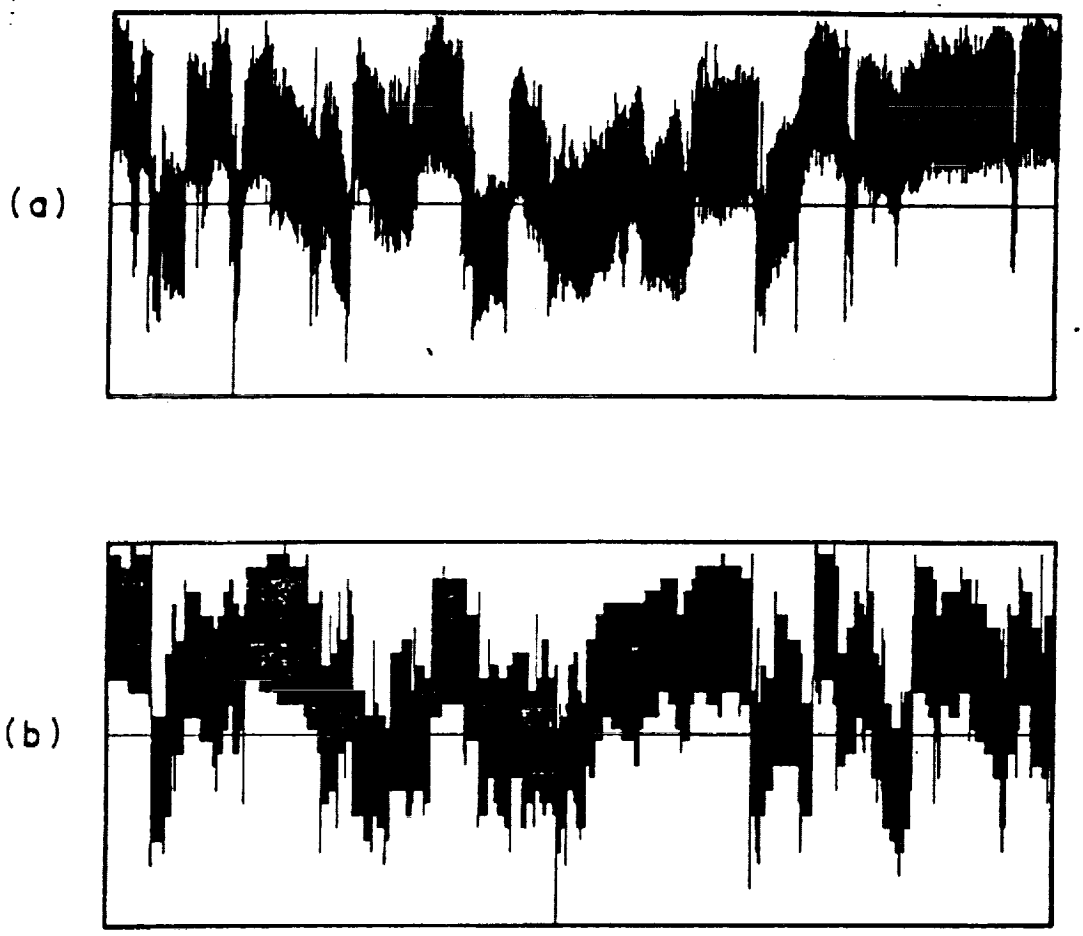


Figure B.2. Comparison of original (a), and reconstructed (b) histories.

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181942		

4. Title and Subtitle	5. Report Date
Fatigue Loading History Reconstruction Based on the Rain-Flow Technique	December 1989
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
A.K. Khosrovaneh and N.E. Dowling	
	10. Work Unit No. 505-63-01-05
9. Performing Organization Name and Address	11. Contract or Grant No.
Virginia Polytechnic Institute and State University Engineering Science and Mechanics Department Blacksburg, VA 24061	NAG1-822
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: R. A. Everett, Jr.

16. Abstract

Methods are considered of reducing a non-random fatigue loading history to a concise description and then of reconstructing a time history similar to the original. In particular, three methods of reconstruction based on a rain-flow cycle counting matrix are presented. A rain-flow matrix consists of the numbers of cycles at various peak and valley combinations. Two methods are based on a two dimensional rain-flow matrix, and the third on a three dimensional rain-flow matrix. Histories reconstructed by any of these methods produce a rain-flow matrix identical to that of the original history, and as a result the resulting time history is expected to produce a fatigue life similar to that for the original. The procedures described allow lengthy loading histories to be stored in compact form.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Rain-flow Spectrum Reconstruction Fatigue Local Strain Cycle Counting	Unclassified - Unlimited Subject Category - 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	72	A04