Simple rainflow counting algorithms

S. D. Downing and D. F. Socie

Two simple algorithms for performing rainflow counting are presented in this paper. The second algorithm is suitable for microcomputer devices that are placed in vehicles to record field data.

Key words: fatigue tests; rainflow counting; algorithms; load monitoring; ground vehicles

In the land-vehicle industry, cumulative damage fatigue analysis procedures are usually employed to estimate endurance.[1–3] They allow the engineer to relate the endurance of actual components to simple laboratory specimens. Fatigue lives of specimens are determined from constant amplitude tests. Real structures seldom, if ever, experience constant amplitude loading. Therefore, some type of cycle counting scheme must be employed to reduce a complex irregular loading history into a series of constant amplitude events. The most accurate fatigue life estimates are obtained using an analysis based on the strain at the most highly stressed/strained location. Rainflow counting[4] is an essential part of these procedures. This method defines cycles as closed stress/strain hysteresis loops as illustrated in Fig. 1. Four cycles (bc, ed, fg, ad) are identified by the method.

Several algorithms are available to perform the counting, however, they all require that the entire load history be known before the counting process starts.[5–7] As a result, they are not suitable for 'on-board' data processing since the entire load history isn't known until the end of the test. The first algorithm described in this paper has this same limitation; that is, the load history must be rearranged to begin and end with the maximum peak (or minimum valley). It is presented because of its simplicity and because it is useful as a control program for determining stress/strain response under variable amplitude loading. The 'one-pass' rainflow counting algorithm described later overcomes this limitation and identifies the same cycles as the first algorithm. Thus, it can operate in 'real-time' and has been successfully implemented in a histogram recorder.[8]

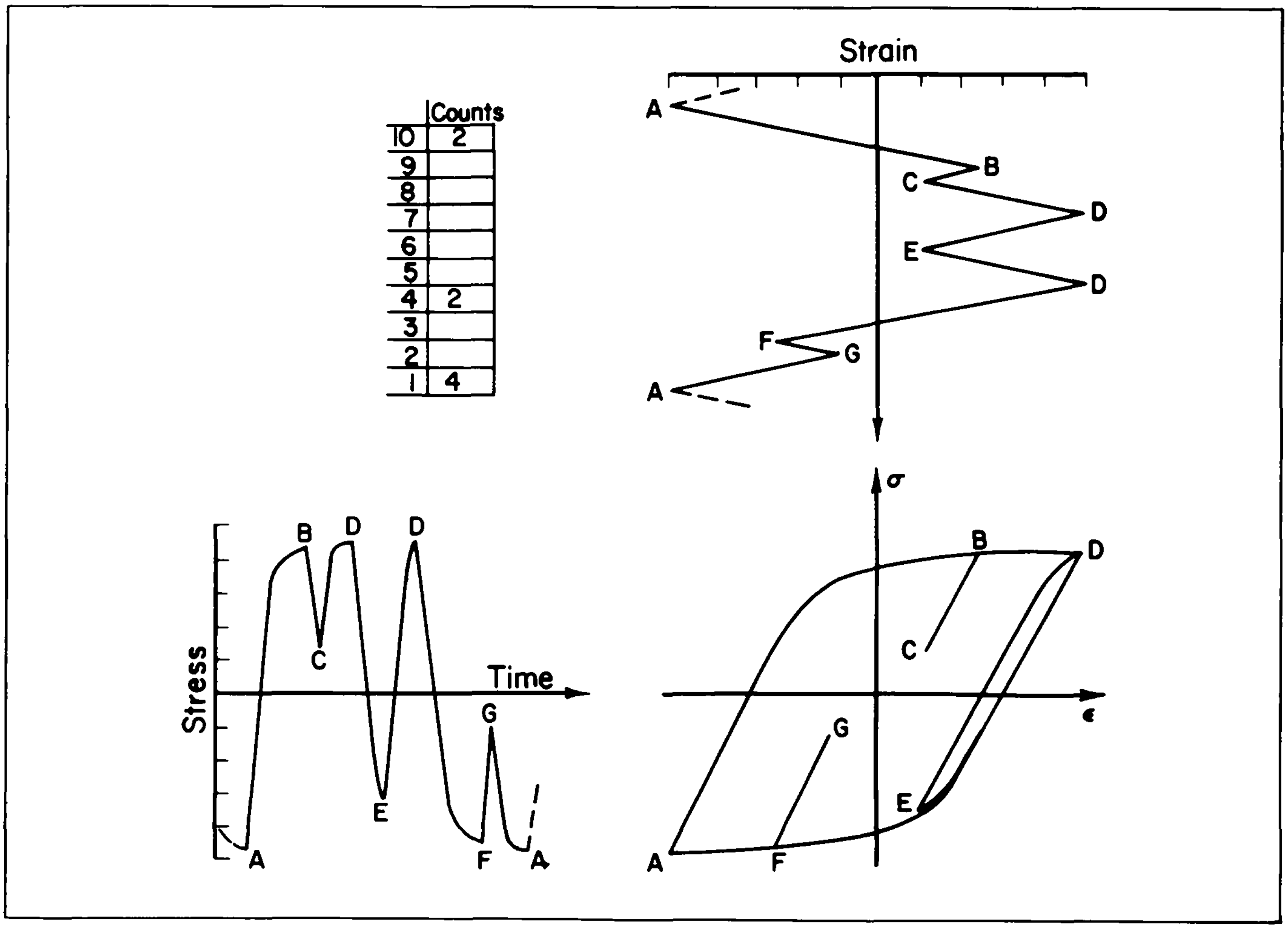


Fig. 1 Stress/strain response and rainflow counting

PREVIOUS WORK

Most practical rainflow counting algorithms are based on either the 'availability matrix' or the 'vector' mathematical concepts. The 'availability matrix' algorithm developed by Wetzel[9] requires that the input signal be divided into a finite number of bands which are used to define the numerical value of the range and mean of each reversal. Corresponding to each band is an element in the availability matrix. Simply speaking, this matrix is used to determine when a rainflow counted cycle is formed.

'Vector' based rainflow counting algorithms use a one dimensional array to keep track of those peaks and valleys which have not formed a closed loop. In other words, once a closed loop has been determined, the peak and valley associated with it can be eliminated from the vector. This technique was first demonstrated by Downing *et al*[2] and was modified by Okamura *et al*[10] to account for half cycles. Both algorithms described in this paper use the 'vector' concept.

RULES FOR BOTH ALGORITHMS

Let the range of each peak and valley be identified as follows:

X = range under consideration
Y = previous range adjacent to X

As each peak or valley is encountered, it is put in a vector $E(n)$. In addition, the starting peak or valley is designated S.

RAINFLOW ALGORITHM I

This algorithm rainflow counts a history of peaks and valleys in sequence which has been rearranged to begin and end with the maximum peak (or minimum valley). Rainflow counting then proceeds according to the following steps:

1 – Read the next peak or valley
(if out of data, STOP)
2 – Form ranges X and Y
(if the vector contains less than 3 points, go to Step 1)
3 – Compare ranges X and Y
a. If $X < Y$, go to Step 1
b. If $X \geqslant Y$, go to Step 4
4 – Count range Y
Discard the peak and valley of Y
Go to Step 2

RAINFLOW ALGORITHM II (ONE-PASS)

This algorithm rainflow counts a history of peaks and valleys in sequence as they occur. It calculates the same ranges and means as Rainflow Algorithm I which required that the history be rearranged to begin and end with the maximum peak (or minimum valley). Rainflow counting then proceeds according to the following steps:

1 – Read the next peak or valley
(if out of data, go to Step 6)
2 – Form ranges X and Y
(if the vector contains less than 2 points past the starting point, go to Step 1)
3 – Compare ranges X and Y
a. If $X < Y$, go to Step 1
b. If $X = Y$ and Y contains S, go to Step 1
c. If $X > Y$ and Y contains S, go to Step 4
d. If $X \geqslant Y$ and Y does not contain S, go to Step 5
4 – Move S to the next point in the vector
Go to Step 1
5 – Count range Y
Discard the peak and valley of Y
Go to Step 2
6 – Read the next peak or valley from the beginning of the vector $E(n)$
(if the starting point, S, has already been reread, STOP)
7 – Form ranges X and Y
(if the vector contains less than 2 points past the starting point, go to Step 6)
8 – Compare ranges X and Y
a. If $X < Y$, go to Step 6
b. If $X \geqslant Y$, go to Step 9
9 – Count range Y
Discard the peak and valley of Y
Go to Step 7

EXAMPLES

Both algorithms will be illustrated by rainflow counting the strain/time history shown in Fig. 2. Fig. 3 shows the same history after it has been rearranged to begin and end with the maximum peak, point C. Also given is the resulting stress/strain response which shows a number of closed hysteresis loops. Rainflow counting should identify the ranges of strain which correspond to these closed hysteresis loops.

Rainflow Algorithm I is illustrated in conjunction with Figs 4–16. In each figure, the strain/time history shown corresponds to the contents of the vector $E(n)$. Also shown is the stress/strain plot, the values of ranges X and Y, and the decisions which correspond to Step 3 of the rules for this algorithm. The history to be rainflow counted is given in Fig. 3. In Fig 4, the first peak has been read into the vector. This establishes the origin of the stress/strain plot since either the maximum peak or the minimum valley lies on the cyclic stress/strain curve. Since there are less than 3 points in the vector, ranges X and Y are undetermined and the next peak or valley must be read. In Fig. 5,

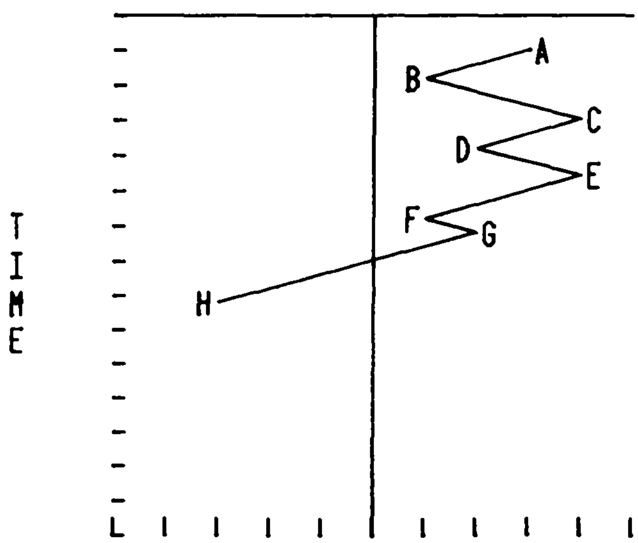


Fig. 2 Variable amplitude history

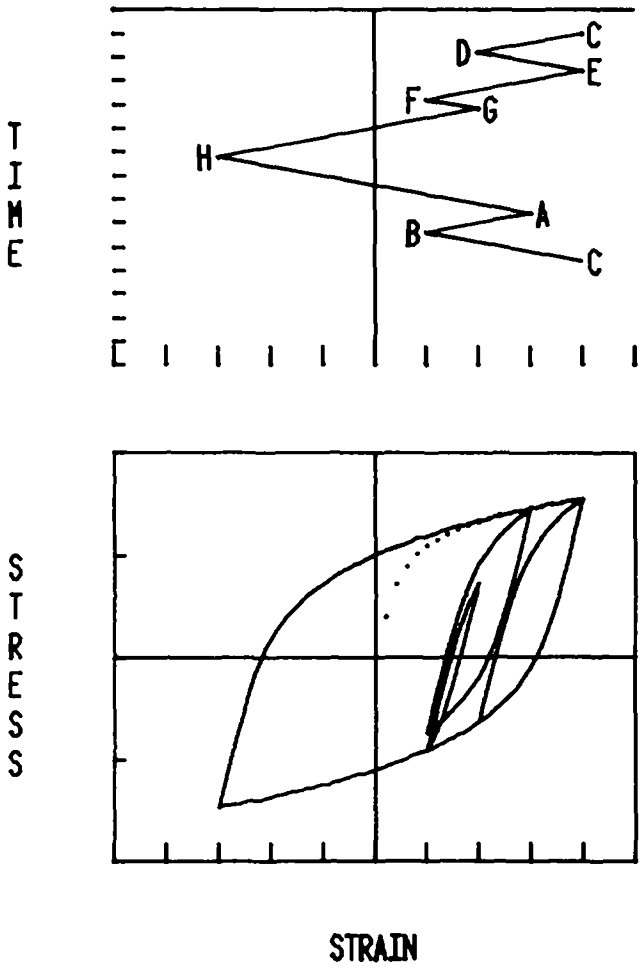


Fig. 3 Stress/strain response

point D has been read into the vector and the stress is unloaded from C to D along the outer loop curve. Range Y is still undetermined so the next peak or valley must be read. In Fig. 6, point E has been read and the stress increases from D to E along the outer loop curve. A closed hysteresis loop has been formed and, according to the counting rules, range Y should be counted and its points discarded since they have no bearing on future events. The counting algorithm identified the same cycle, DC, as was determined from the stress/strain response. In Fig. 7, points D and C have been eliminated from the contents of the vector.
It is left to the reader to follow Figs 7–16 along with the counting rules to see that the algorithm identifies the same cycles (DC, GF, BA, HE) as were determined from the stress/strain response.

Rainflow Algorithm II (One-Pass) will be used to count the strain/time history given in Fig. 2. It should identify the same cycles as the previous algorithm without the restriction that history be rearranged to begin and end with the maximum peak. Figs 17–31 show the contents of the vector $E(n)$ and the counting decisions for each step in the counting process. It should be noted that the starting point, S, is always the first occurrence of either the maximum peak or the minimum valley at that point in the history. When all the peaks and valleys of the history have been read, we begin reading points from the beginning of the vector as seen in Fig. 17. The counting procedure continues until all the points up to and including the starting point have been reread. When we try to read a point beyond the starting point, the counting procedure stops and all the cycles have been determined. Fig. 31 shows that the same cycles (DC, GF, BA, HE) have been identified as in the previous algorithm. Again, the reader should carefully follow Figs 17–31 to fully understand this algorithm.

FORTRAN PROGRAM LISTINGS

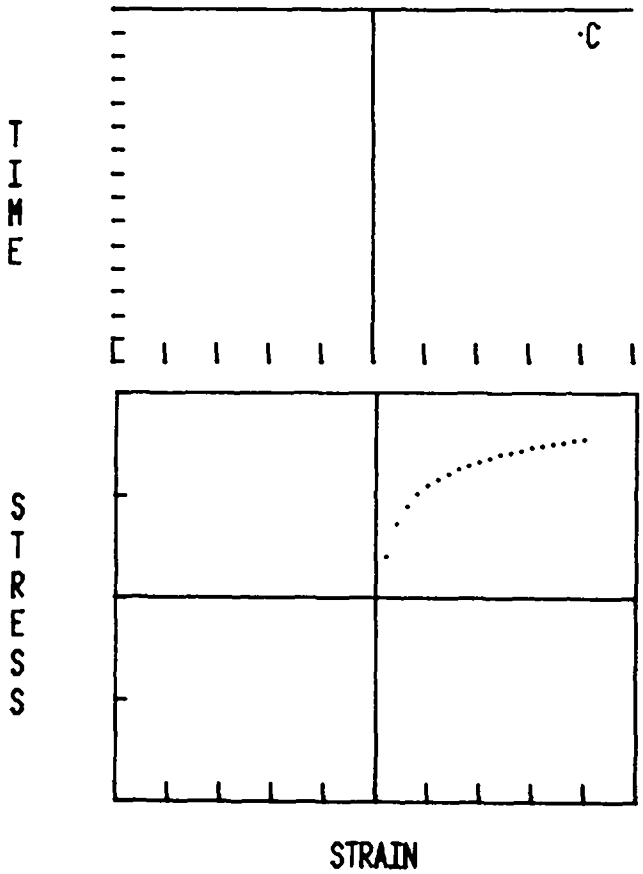
A *Fortran* listing for Rainflow Algorithm I is contained in Appendix I. The reader needs to write his own version of *Subroutine Data* (P, K) compatible with his data files. The variable, P, is the value of the data point. The variable, K, should be defined as follows:

$K = 0$ when the data is valid;
$K = 1$ when the history is finished.

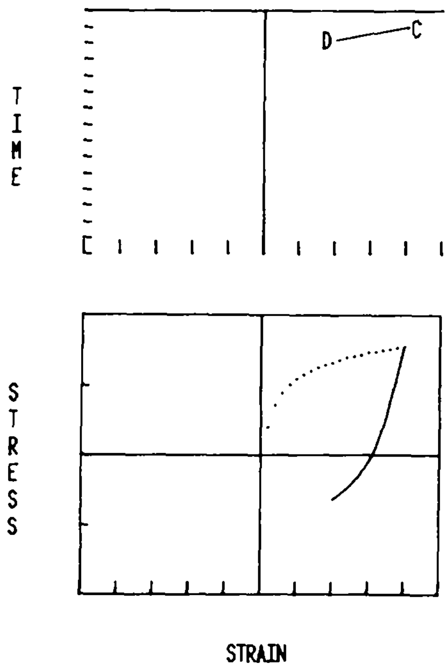
The data returned from this subroutine must be peaks and valleys in sequence and must begin and end with the maximum peak (or minimum valley). The maximum size of the vector, $E(n)$, is equal to the number of counting ranges.

Appendix II gives the *Fortran* listing for Rainflow Algorithm II (One-Pass). This program checks for data sequence so that the variable, P, in *Subroutine Data* (P, K) may be timed data samples. The meaning of variable, K, remains the same as above. For this algorithm the maximum size of the vector $E(n)$ is equal to twice the number of counting ranges.

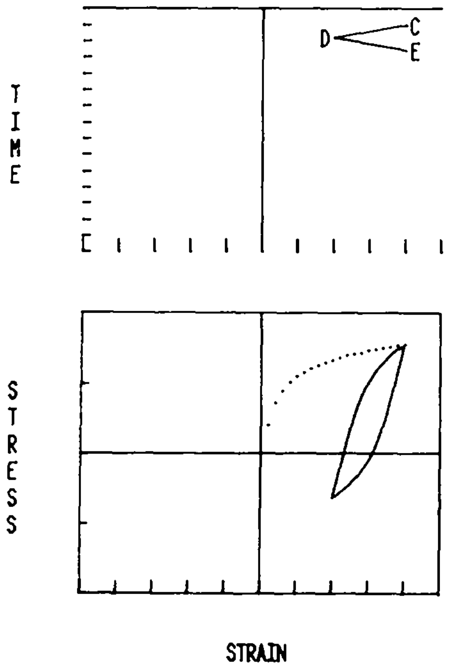
Rainflow Algorithm I is illustrated in conjunction with Figs 4–16



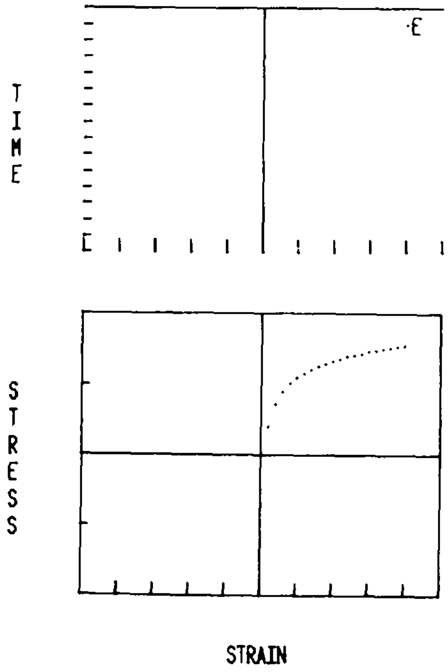
X = UNDETERMINED
Y = UNDETERMINED
READ NEXT PEAK OR VALLEY
FIG. 4



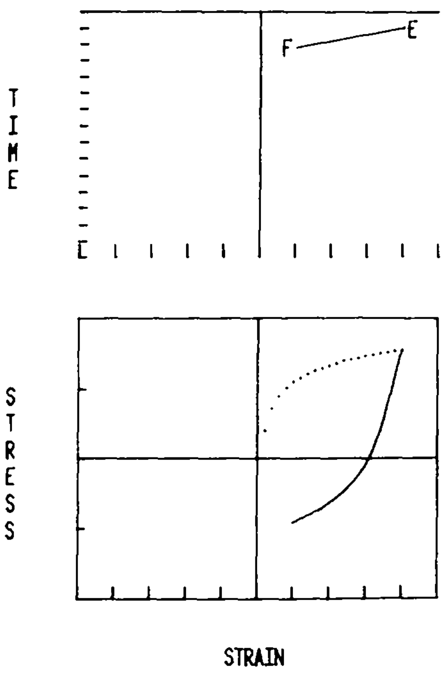
X = ABS(D-C)
Y = UNDETERMINED
READ NEXT PEAK OR VALLEY
FIG. 5



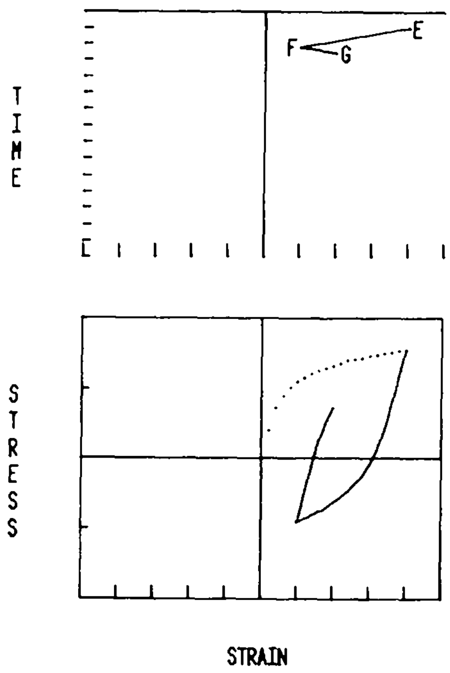
X = ABS(E-D)
Y = ABS(D-C)
X=Y,COUNT Y AND DISCARD IT'S POINTS
FIG. 6



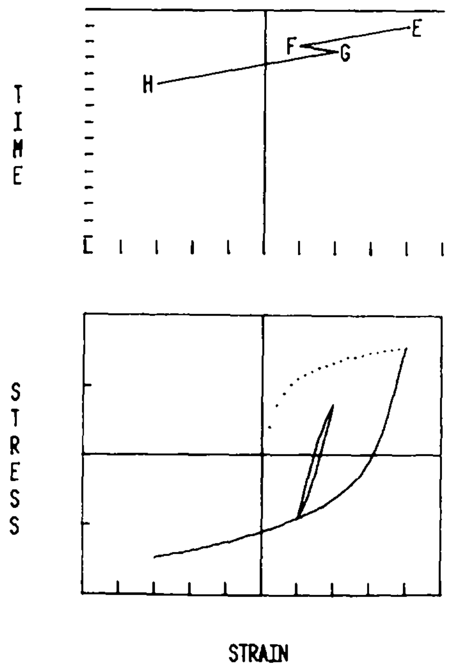
CYCLES = DC
X = UNDETERMINED
Y = UNDETERMINED
READ NEXT PEAK OR VALLEY
FIG. 7



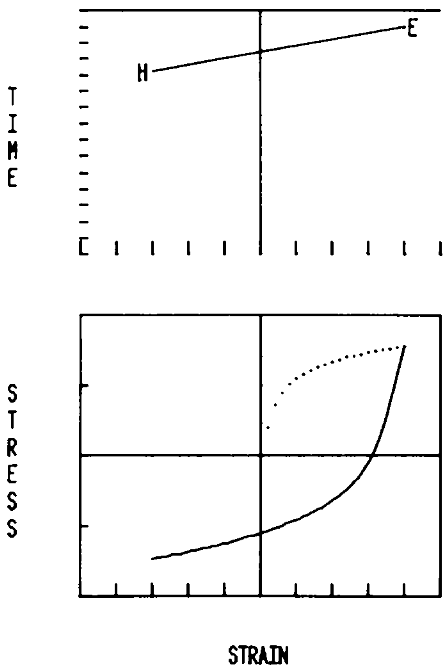
CYCLES = DC
X = ABS(F-E)
Y = UNDETERMINED
READ NEXT PEAK OR VALLEY
FIG. 8



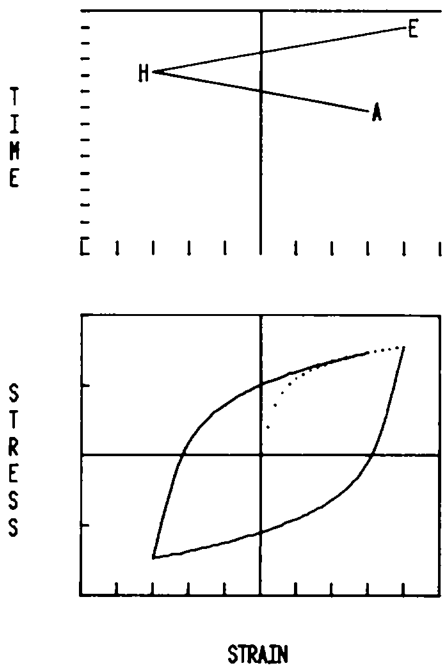
CYCLES = DC
X = ABS(G-F)
Y = ABS(F-E)
X<Y,READ NEXT PEAK OR VALLEY
FIG. 9



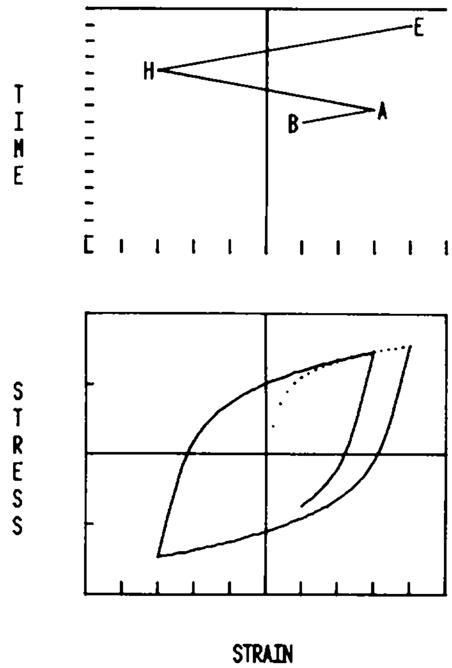
CYCLES = DC
X = ABS(H-G)
Y = ABS(G-F)
X>Y,COUNT Y AND DISCARD IT'S POINTS
FIG. 10



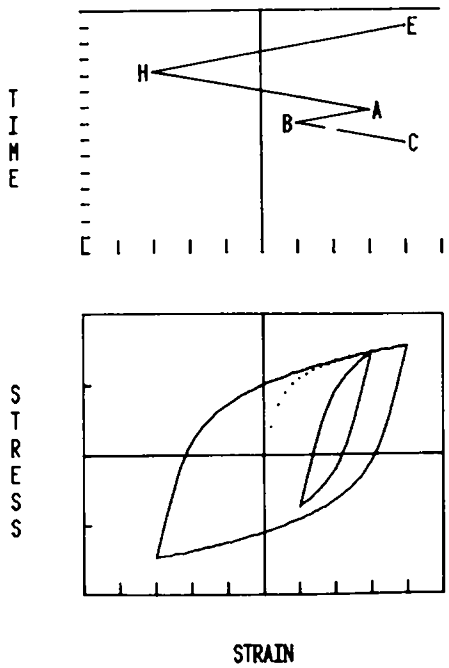
CYCLES = DC,GF
X = ABS(H-E)
Y = UNDETERMINED
READ NEXT PEAK OR VALLEY
FIG. 11



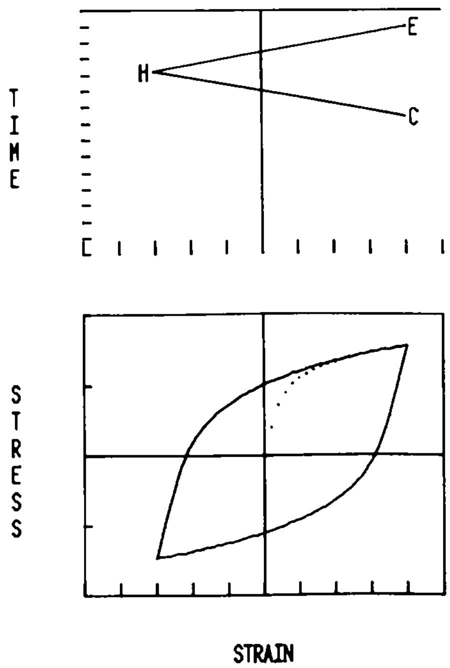
CYCLES = DC,GF
X = ABS(A-H)
Y = ABS(H-E)
X<Y,READ NEXT PEAK OR VALLEY
FIG. 12



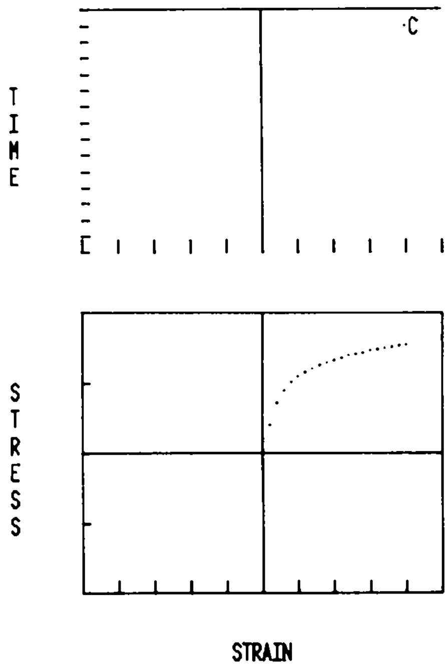
CYCLES = DC,GF
X = ABS(B-A)
Y = ABS(A-H)
X<Y,READ NEXT PEAK OR VALLEY
FIG. 13



CYCLES = DC,GF
X = ABS(C-B)
Y = ABS(B-A)
X>Y,COUNT Y AND DISCARD IT'S POINTS
FIG. 14



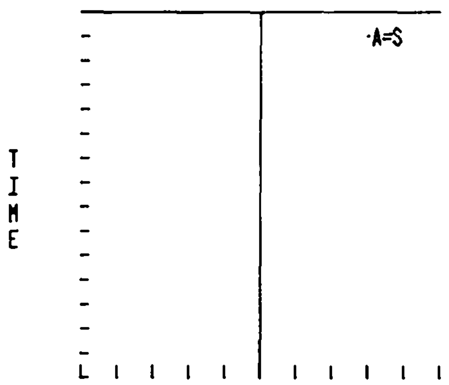
CYCLES = DC,GF,BA
X = ABS(C-H)
Y = ABS(H-E)
X=Y,COUNT Y AND DISCARD IT'S POINTS
FIG. 15



CYCLES = DC,GF,BA,HE
WE ARE FINISHED

FIG. 16

ACTIVELY TAKING DATA

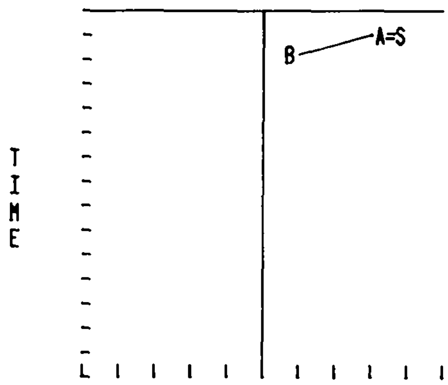


X = UNDETERMINED
Y = UNDETERMINED

READ NEXT PEAK OR VALLEY

FIG. 17

ACTIVELY TAKING DATA

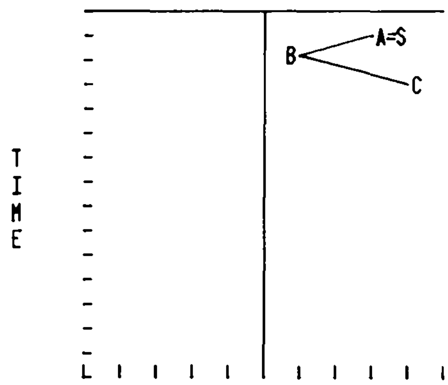


X = ABS(B-A)
Y = UNDETERMINED

READ NEXT PEAK OR VALLEY

FIG. 18

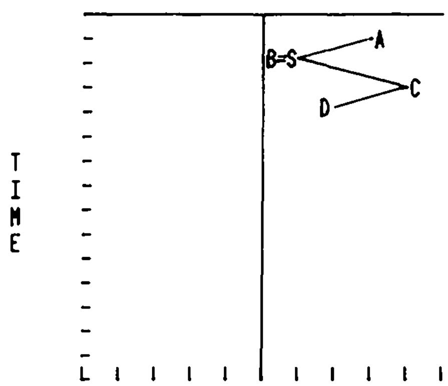
ACTIVELY TAKING DATA



X = ABS(C-B)
Y = ABS(B-A)
X>Y AND Y CONTAINS S
MOVE UP STARTING POINT
READ NEXT PEAK OR VALLEY

FIG. 19

ACTIVELY TAKING DATA

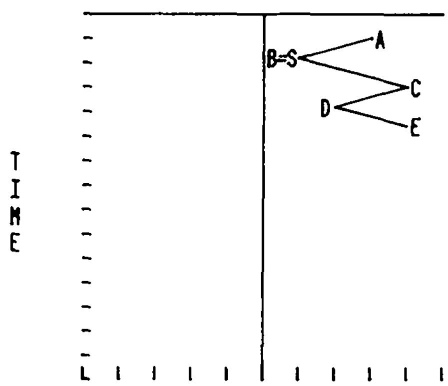


X = ABS(D-C)
Y = ABS(C-B)
X<Y

READ NEXT PEAK OR VALLEY

FIG. 20

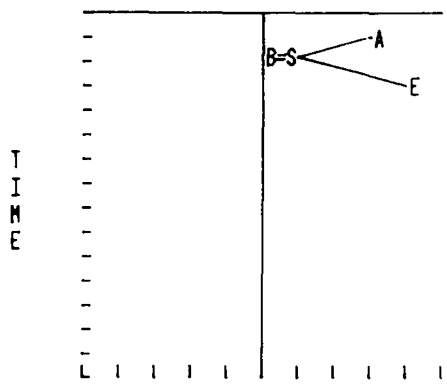
ACTIVELY TAKING DATA



X = ABS(E-D)
Y = ABS(D-C)
X=Y AND Y DOES NOT CONTAIN S
COUNT Y AND DISCARD IT'S POINTS

FIG. 21

ACTIVELY TAKING DATA



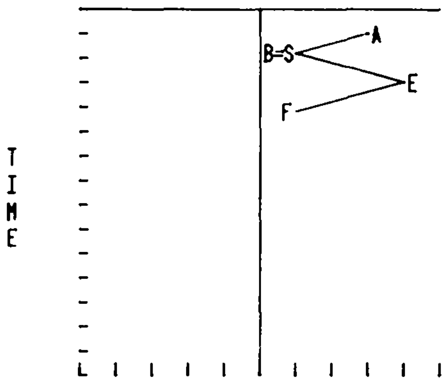
CYCLES = DC

X = ABS(E-B)
Y = UNDETERMINED

READ NEXT PEAK OR VALLEY

FIG. 22

ACTIVELY TAKING DATA



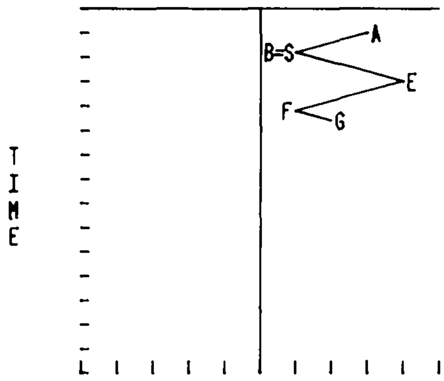
CYCLES = DC

X = ABS(F-E)
Y = ABS(E-B)
X=Y AND Y CONTAINS S

READ NEXT PEAK OR VALLEY

FIG. 23

ACTIVELY TAKING DATA



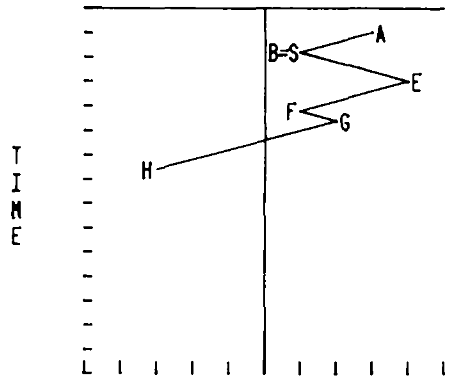
CYCLES = DC

X = ABS(G-F)
Y = ABS(F-E)
X<Y

READ NEXT PEAK OR VALLEY

FIG. 24

ACTIVELY TAKING DATA

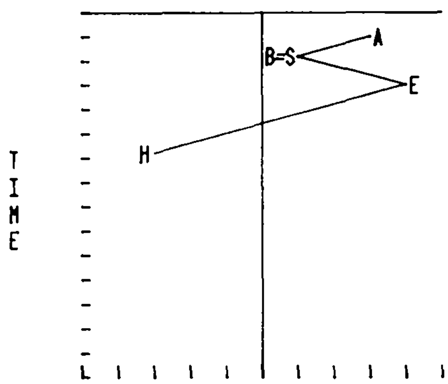


CYCLES = DC

X = ABS(H-G)
Y = ABS(G-F)
X>Y AND Y DOES NOT CONTAIN S
COUNT Y AND DISCARD IT'S POINTS

FIG. 25

ACTIVELY TAKING DATA

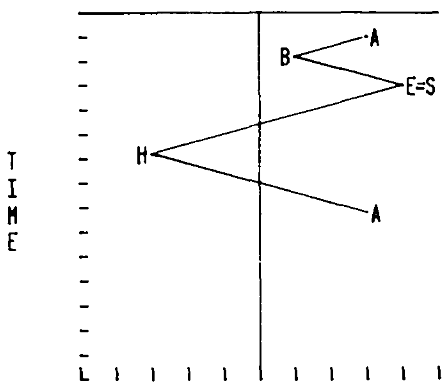


CYCLES = DC,GF

X = ABS(H-E)
Y = ABS(E-B)
X>Y AND Y CONTAINS S
MOVE UP STARTING POINT
READ NEXT PEAK OR VALLEY

FIG. 26

OUT OF DATA



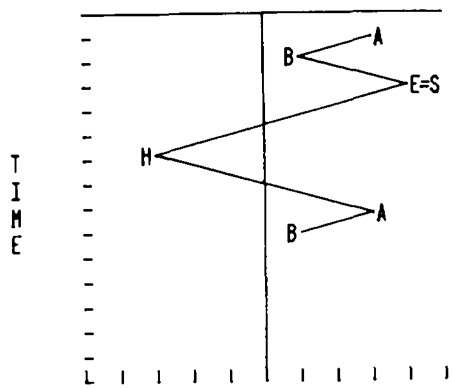
CYCLES = DC,GF

X = ABS(A-H)
Y = ABS(H-E)
X<Y

READ NEXT PEAK OR VALLEY

FIG. 27

OUT OF DATA



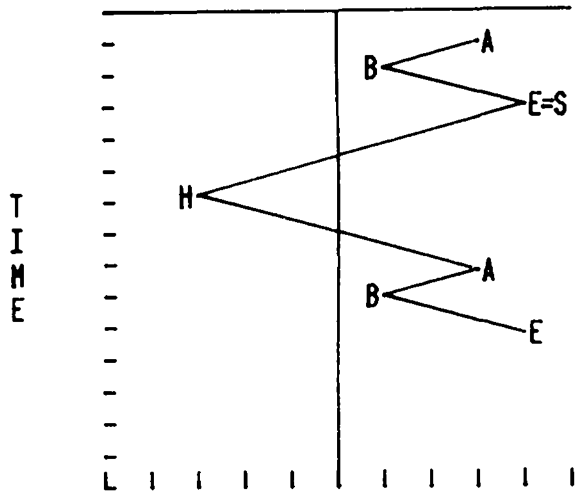
CYCLES = DC,GF

X = ABS(B-A)
Y = ABS(A-H)
X<Y

READ NEXT PEAK OR VALLEY

FIG. 28

OUT OF DATA

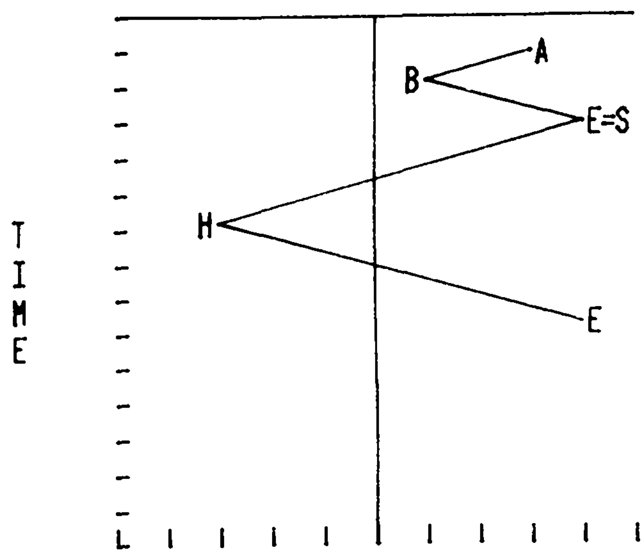


CYCLES = DC,GF

X = ABS(E-B)
Y = ABS(B-A)
X>Y AND Y DOES NOT CONTAIN S
COUNT Y AND DISCARD IT'S POINTS

FIG. 29

OUT OF DATA

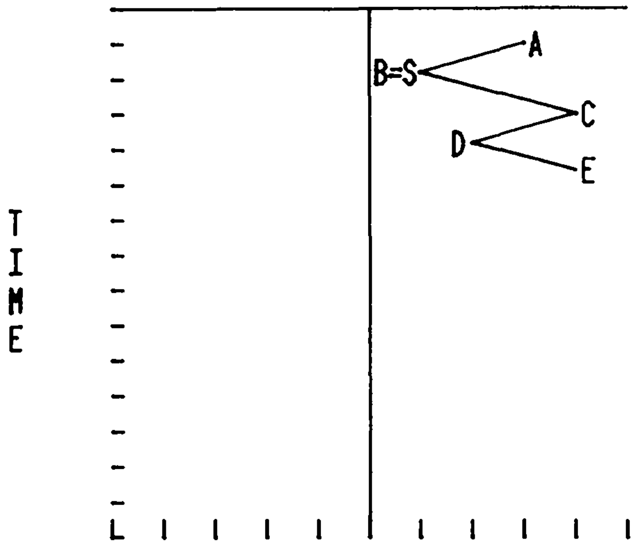


CYCLES = DC,GF,BA

X = ABS(E-H)
Y = ABS(H-E)
X=Y AND Y CONTAINS S
COUNT Y AND DISCARD IT'S POINTS

FIG. 30

ACTIVELY TAKING DATA



X = ABS(E-D)
Y = ABS(D-C)
X=Y AND Y DOES NOT CONTAIN S
COUNT Y AND DISCARD IT'S POINTS

FIG. 31

SUMMARY

Two simple rainflow counting algorithms for processing field data have been presented. The first is useful as a control program for following stress/response under irregular loading. The second algorithm gives identical results as the first and has the advantage that the counting can begin before the entire history is known.

REFERENCES

1. **Dabell, B. J., Hill, S. J., Eaton, D. E. and Watson, P.** 'Fatigue life predictions for notched components' *J Soc Environmental Engrs* (December 1977)
2. **Downing, S., Galliart, D. and Bereyni, T.** 'A Neubers rule fatigue analysis procedure for use with a mobile computer' SAE Paper 760317 presented at: *SAE Automotove Engineering Congress* (Detroit, Michigan, 1976)
3. *Fatigue Under Complex Loading: Analysis and Experiment* Edited by: R. M. Wetzel (SAE Inc, Warrendale, Pennsylvania, 1977)
4. **Matsuishi, M. and Endo, T.** 'Fatigue of metals subjected to varying stress' paper presented to *Japan Soc Mech Engrs* (Jukvoka, Japan, 1968)
5. **Richards, F., LaPointe, N. and Wetzel, R.** 'A cycle counting algorithm for fatigue damage analysis' Paper No 740278 presented at: *SAE Automotive Engineering Congress* (Detroit, Michigan, 1974)
6. **Nelson, D. V. and Fuchs, H. O.** 'Predictions of cumulative damage using condensed load histories' Paper 750045 presented at: *SAE Automotive Engineering Congress* (Detroit, Michigan, 1975)

7. **Socie, D. F.** 'Fatigue-life prediction using local stress/strain concept' *Experimental Mech* **17** No 2 (1977) pp 50–56

8. **Socie, D. F., Shifflet, G. and Berns, H.** 'A field recording system with applications to fatigue analysis' *Int J Fatigue* **1** No 2 (April 1979) pp 103–111

9. **Wetzel, R. M.** 'A method of fatigue damage analysis' *PhD Thesis* (Department of Civil Engineering, University of Waterloo, Ontario, Canada, 1971)

10. **Okamura, H., Sakai, S. and Susuki, I.** 'Cumulative fatigue damage under random loads' *Fatigue Engng Mater and Struct* **1** (1979) pp 409–419

AUTHORS

Stephen Downing is with Deere and Company's Engineering Mechanics Group in Moline and Darrell Socie is with the Department of Mechanical and Industrial Engineering in the University of Illinois at Urbana-Champaign. In the first instance inquiries should be addressed to: Mr S. D. Downing, Engineering Mechanics, Deere and Company, 3300 River Drive, Moline, Illinois 61265, USA.

Appendix 1

```
C     RAINFLOW ALGORITHM I
C
C     THIS PROGRAM RAINFLOW COUNTS A HISTORY OF PEAKS
C     AND VALLEYS IN SEQUENCE WHICH HAS BEEN REARRANGED
C     TO BEGIN AND END WITH THE MAXIMUM PEAK (OR MINIMUM
C     VALLEY).  STATEMENT LABELS CORRESPOND TO THE STEPS IN
C     THE RAINFLOW COUNTING RULES.
C
      DIMENSION E(50)
      N=0
    1 N=N+1
      CALL DATA(E(N),K)
      IF(K.EQ.1) STOP
    2 IF(N.LT.3) GO TO 1
      X=ABS(E(N)-E(N-1))
      Y=ABS(E(N-1)-E(N-2))
    3 IF(X.LT.Y) GO TO 1
    4 RANGE=Y
      XMEAN=(E(N-1)+E(N-2))/2.
      N=N-2
      E(N)=E(N+2)
      GO TO 2
      END
```

Appendix 2

```
C     RAINFLOW ALGORITHM II (ONE-PASS)
C
C     THIS PROGRAM RAINFLOW COUNTS A HISTORY AS IT OCCURS AND
C     IDENTIFIES THE SAME CYCLES AS RAINFLOW ALGORITHM I WHICH
C     REQUIRES THAT THE HISTORY BE REARRANGED.  STATEMENT LABELS
C     1-9 CORRESPOND TO THE STEPS IN THE RAINFLOW COUNTING RULES.
C
      DIMENSION E(100)
      N=2
      J=0
      ISTART=1
      CALL DATA(E(1),K)
  100 CALL DATA(E(2),K)
      IF(E(1).EQ.E(2)) GO TO 100
      SLOPE=1.
      IF(E(1).GT.E(2)) SLOPE=-1.
    1 CALL DATA(P,K)
      IF(K.EQ.1) GO TO 6
      N=N+1
      SLOPE=SLOPE*(-1.)
      E(N)=P
```

```
    2 IF(N.LT.ISTART+1) GO TO 1
      X=SLOPE*(E(N)-E(N-1))
      IF(X.LE.0.) GO TO 200
      IF(N.LT.ISTART+2) GO TO 1
      Y=SLOPE*(E(N-2)-E(N-1))
    3 IF(X.LT.Y) GO TO 1
      IF(X.EQ.Y.AND.ISTART.EQ.N-2) GO TO 1
      IF(X.GT.Y.AND.ISTART.EQ.N-2) GO TO 4
      IF(X.GE.Y.AND.ISTART.NE.N-2) GO TO 5
    4 ISTART=ISTART+1
      GO TO 1
    5 RANGE=Y
      XMEAN=(E(N-1)+E(N-2))/2.
      N=N-2
      E(N)=E(N+2)
      GO TO 2
    6 J=J+1
      IF(J.GT.ISTART) STOP
      N=N+1
      SLOPE=SLOPE*(-1.)
      E(N)=E(J)
    7 IF(N.LT.ISTART+1) GO TO 6
      X=SLOPE*(E(N)-E(N-1))
      IF(X.LE.0.) GO TO 300
      IF(N.LT.ISTART+2) GO TO 6
      Y=SLOPE*(E(N-2)-E(N-1))
    8 IF(X.LT.Y) GO TO 6
      IF(X.GE.Y) GO TO 9
    9 RANGE=Y
      XMEAN=(E(N-1)+E(N-2))/2.
      N=N-2
      E(N)=E(N+2)
      GO TO 7
  200 N=N-1
      E(N)=E(N+1)
      SLOPE=SLOPE*(-1.)
      GO TO 2
  300 N=N-1
      E(N)=E(N+1)
      SLOPE=SLOPE*(-1.)
      GO TO 7
      END
```