Graphical Rainflow Counting - Complete Algorithm for single courses (SCRF)

Sven Füser
2023-04-22
HAW Hamburg, Germany

Contents

1 Introduction **3**

2 Fundamental points and definitions **10**
2.1 Preparation of the raw signal 10
2.2 Rules of rainflow counting 10
2.3 Pairing the courses into closed loops 11

3 Mode of operation of the SCRF algorithm **12**
3.1 General functioning of the course determination 12
3.2 Explanation of the course determination by means of an example 15
3.3 Explanation of the pairing algorithm 18

4 Verification **20**
4.1 Detailed comparison to other counting algorithms 21
4.1.1 Single cycle 21
4.1.2 Consideration of the residue vector 21
4.1.3 Reversing a signal sequence in index direction 23
4.1.4 Signal sequence with the same initial and final value 23
4.2 Comparison using random sequences 23
4.2.1 Comparison using a random *periodic* sequence 23
4.2.2 Comparison using a random *non-periodic* sequence 24
4.3 Conclusion from the comparisons to other counting algorithms 24
4.4 Comparison of SCRF algorithm variants regarding course order 25
4.5 Evaluation in terms of computer technology 25

5 Treatment of residues **26**

6 Conclusion **27**

Abstract

Rainflow counting can be considered the most important method in counting random stress- or strain histories with respect to time in the field of fatigue strength. The method is often explained by rules of a rain flow which gave the name to the method. The algorithmic implementation nevertheless is usually done in three-point or four-point algorithms which work by comparing the differences between the adjacent reversals and then delete the points of the identified intermediate cycle from the time history.

The SCRF-algorithm presented in this contribution determines the courses of the rain flow in terms of their start und stop classes as half-cycles. In a subsequent step the half-cycles are matched together to form a closed loop resp. kept as a residue if the counter-course is missing. The algorithm is explained in general und by means of an example. For verification the SCRF-algorithm is compared to a three-point and a four-point algorithm.

For periodic signals, there is complete agreement between the three methods. For non-periodic signals, there are small differences: The SCRF can combine some more half-cycles into closed cycles which remain as residues in the other methods. Furthermore the sense of direction of the hysteresis loops is captured more precisely with the SCRF due to its chronological approach.

Keywords

Cycle Counting Methods
Rainflow counting
Rainflow Residue
Random Loading
Load Histories
Hysteresis Loop
Algorithm
Fatigue analysis

Conflicts of Interest

None.

Contact to the author

Prof. Dr.-Ing. Sven Füser
Mail: `sven.fueser@haw-hamburg.de`
HAW Hamburg, Berliner Tor 9, 20099 Hamburg, Germany

1 Introduction

The rainflow counting method is known to have originated in the publication of MATSUISHI und ENDO, which appeared in 1968 in Japanese [1], also included in [2]. Rainflow counting is the preferred counting method in fatigue strength for the extraction of the closed loops from a history of load, stress or strain related to time. Its success is founded on an underlying interpretation of the material mechanics since the identified hysteresis loops can be regarded as damaging events [3]. In [3] we also find the graphical procedure of the rainflow algorithm, which has often been mentioned in the descriptions of the rainflow method. Fig. 1 shows the graphical method and its relation to the hysteresis loops in the stress-strain-diagram.

To perform the graphical procedure, in many contributions the time history is rotated so that the time axis points downwards. The edges of the stress-history then form nested roofs over which the rain flows happen. This feature gave the method its name. A rain flow starts from each reversal of the signal. The flow rules define when the course of a rain flow ends. The characteristic recorded for each course is the swept interval in the strain direction resp. stress direction. The algorithm presented is based on a concrete program-technical implementation of these rules. The flow rules are explained in section 2.2.

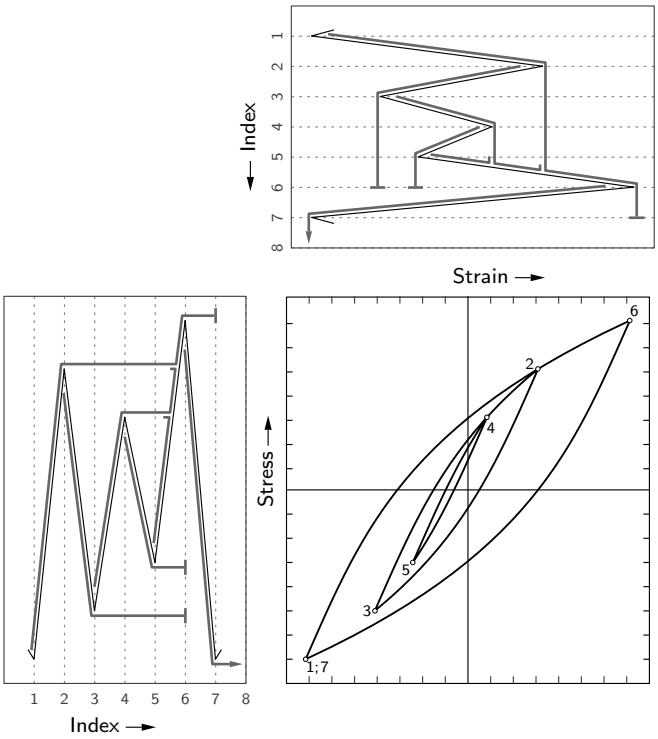

Fig. 1: Stress-strain-history as sequence of reversals

Regardless of the approach used, all rainflow methods reduce the signal to a sequence of the reversals y_k; intervening samples are deleted. In many cases, a classification is also performed, which assigns a class number to the signal value designating the interval. Typical quantities of classes are 64 or 100.

The literature on the subject of rainflow counting is very extensive, cf. [4]. The most important aspects are elaborated below.

Difference algorithms in general

In the literature on rainflow algorithms, the flow rules are rarely used for programmatic implementation. Given the former limitations of computer performance in terms of memory and computational speed, in most approaches these rules have been implemented in the form of algorithms that recognise closed hysteresis loops by taking the difference between the values of adjacent reversals of the signal.

In these difference algorithms, the aim is always to identify an intermediate oscillation of a smaller range along a larger edge. The difference conditions indicate the presence of such an intermediate oscillation. Identified oscillations are counted as closed cycles. Afterwards those reversals, that interrupt the larger edge, are deleted from the signal, which also has the advantage of reducing the amount of data to be held in the computer memory. The early approach of counting the loops as a list of mean value and range as in [5] has given way to a counting in the rainflow matrix, which is either spanned by mean value and range or by lower value and upper value.

Three-point and four-point algorithms

The difference algorithms can be divided into three-point and four-point algorithms. Three-point algorithms use the following condition:

$$X \geq Y \qquad \Leftrightarrow \qquad |\, y_{k+1} - y_k \,| \;\geq\; |\, y_k - y_{k-1} \,| \tag{1}$$

If condition (1) is fulfilled, the loop from y_{k-1} to y_k (and back) is counted. The two signal points y_{k-1} and y_k are deleted. The development of the three-point algorithm takes place along the contributions of Okamura et al., Downing and Socie [5, 6]. The principle of the three-point algorithm resulted in the American standard ASTM E1049, first published in 1985 [7].

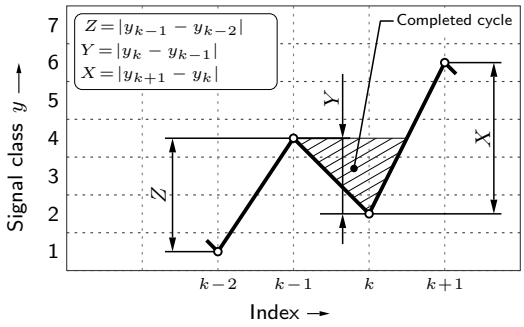

Fig. 2: Comparison of three point and four point criterion

Four-point algorithms recognise an intermediate hysteresis loop by the following criterion:

$$X \geq Y \leq Z \qquad \Leftrightarrow \qquad |\, y_{k+1} - y_k \,| \;\geq\; |\, y_k - y_{k-1} \,| \;\leq\; |\, y_{k-1} - y_{k-2} \,| \tag{2}$$

If condition (2) holds, the cycle between y_{k-1} and y_k is counted and the two signal points y_{k-1} and y_k are deleted. Four-point algorithms reach back into the same time as three-point algorithms. Already ENDO as the inventor of the Rainflow method redefined his counting method by evaluating absolute differences from the signal (peak-valley-differences, PV-differences for short) and by identifying an intermediate loop along a larger edge via a four-point criterion [8, 9].

DRESSLER, BROKATE et. al [10, 11] and AMZALLAG [12] establish the four-point algorithm as a sequence of reversals, from which, according to the four-point condition, intermediate loops are counted and afterwards deleted. This is continued until a residue vector remains, which cannot be further reduced by such deletions. If the sense of direction of the loops (*standing* or *hanging* loops) does not interest, the order in which the intermediate loops are deleted is irrelevant. Smaller distinctions within the four-point algorithms result from whether the four-point condition is formulated with strict or weak inequalities. In fig. 2 the three-point and four-point condition are shown for an ascending edge with an intermediate loop.

Some contributions are based on the idea of aligning the rainflow algorithm particularly close to the processes in the stress-strain diagram. CLORMANN [13] develops a three-point algorithm for this purpose, the HCM algorithm, which simulates the effects of the material memory. The third rule of material memory, which deals with the continuation of stress-strain paths on the initial load curve, is also implemented in this algorithm. However, this makes the counting result dependent on the zero point of the signal. Also the modification described by HONG [14] arrives at an extended four-point criterion by adding a condition of the absolute stress level to the flow rules; this leads to a zero point dependency as well. ANTHES extends the HCM. In order to overcome the problem that the loops are not counted in terms of their order until they reach their closing half-cycle, the opening half-cycle is already evaluated as a damaging event. The later closing of a cycle or non-closing (in the case of a residue) no longer makes a difference. Only a cycle that increases in terms of range in the later part of the signal has to be corrected [15].

Counting of periodic signals

The counting algorithms are often formulated in two sub-variants: one sub-variant for periodic signal sequences and another sub-variant for other sequences. Periodic sequences do not contain residues. In three-point algorithms this is achieved by rearranging the signal: the signal needs to start at the absolute maximum or at the absolute minimum. The part of the signal that is cut off at the front is added at the end; the signal ends at the same value at which it started. The choice of whether to take the absolute minimum or absolute maximum decides if the loop of the biggest range is standing or hanging. The ASTM E1049 [7] has accordingly, in addition to the normal (complete) *Rainflow Counting* its own algorithm for that purpose called *Simplified Rainflow Counting for Repeating Histories*.

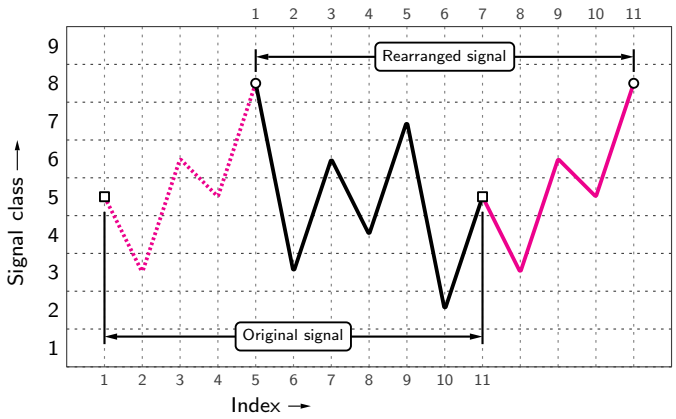

Fig. 3: Performing of a rearrangement

Looking at the four-point algorithm in the question of periodic signals, the residue vector resulting from the (first) counting is appended to itself and counted again. The result of the second counting is added to the one of the first counting; the new residue vector is the original residue vector because a periodic invariance is present [10]. The (old) residue vector resulting from the recalculation is then ignored. This procedure could be interpreted as a rearrangement, but it is done, after all closed cycles have been removed (cf. section 4.1.2). Three-point and four-point algorithms with their sub-variants are documented and compared in [16]. The equivalency of three-point and four-point algorithms shown in [17] refers only to the counting of signals which are considered periodic or which are treated according to the rearrangement described above.

Structure of the residue vector

Counting algorithms for non-periodic signals, on the other hand, can usually process residues. During the counting with difference algorithms the signal sequence is gradually reduced by the identified hysteresis loops. Since each deletion of identified loops deletes two neighbouring points, the resulting data remains a sequence of reversals. When all hysteresis loops have been deleted, the residue vector finally remains. The residue vector consists of the concatenated edges of the residues and is also a sequence of reversals.

The residue vector in the four-point counting has a typical structure, already described in [8] and discussed in [10, 11]. For its description the differences between the n_R individual reversals in the residue vector have to be calculated:

$$d_k \;=\; y_{k+1} - y_k \qquad \text{mit} \quad k = 1, \,\ldots\, , n_R - 1 \tag{3}$$

Then the structure formulated in equation (4) is valid for the absolute values of the differences.

$$0 \;<\; |d_1| \;<\; \ldots \;<\; |d_{J-1}| \;\leq\; |d_J| \;>\; \ldots \;>\; |d_{n_R-1}| \;>\; 0 \tag{4}$$

$$\text{for some } J \text{ with } 1 \leq J \leq n_R$$

The absolute values of the differences increase strictly monotonically up to the center J and its maximum $|d_J|$. The maximum difference can occur twice. Afterwards the absolute

differences decrease monotonically, cf. fig. 4. It is also possible that the increasing section and/or the decreasing section each is missing.

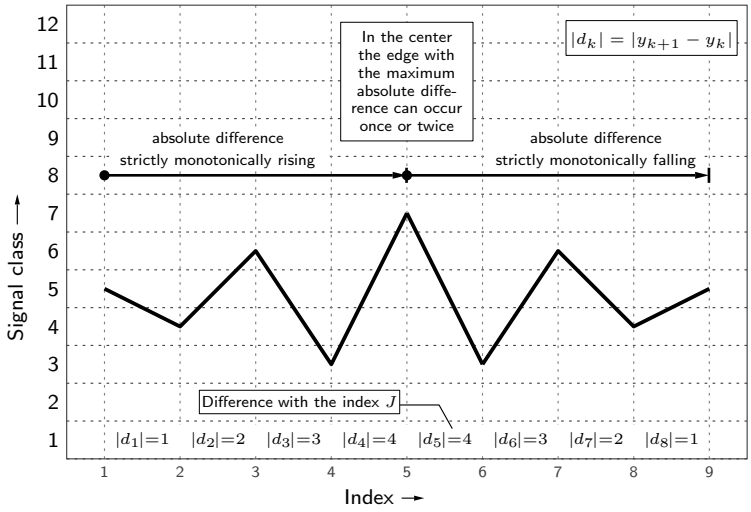

Fig. 4: Structure of the residue vector resulting from a four point counting

When a signal is divided into classes, the number of possible differences between the classes is significantly reduced and thus the maximum length of the residue vector is quite small: With K classes, $(K-1)$ distinguishable absolute difference amounts are possible, which in turn can lead to a maximum of $2(K-1)$ edges in the residue vector.

How the algorithms deal with residues

It is undisputed that residues for short signal sequences must be considered. One approach is to count residues as half-cycles, cf. [7, 13]. Another approach is to force the closure of the loops as in the case of [12, 14, 15, 17, 18].

By either

- a rearrangement,
- by the subsequent counting of the doubly concatenated residue vector or
- by immediately counting the opening half-cycle as a whole cycle

a forced closing of hysteresis loops is achieved. These approaches therefore aim at describing the signal as a periodic and thus a quasi-complete signal.

The residue vector can also be used to continue the counting [13, 18] or to recount it with another counting method [18].

Improvements regarding the computer-technical implementation

The development of the rainflow algorithms has also been driven by the state of the art in computer technology at that time. Especially the limitation of memory space had a strong impact. For example, the release of memory after the identification of closed loops [5, 10, 13] or the buffering of only one part of the signal [19] have been proposed. Related to this was the

possibility for on-board counting, i.e. a possibility to carry out the counting, without having already completed the measurement. In on-board counting the release of memory plays as well a role as the handling of data at the beginning of the measured sequence.

In [20] problems of graphical rainflow algorithms are evaluated in terms of its need for resources for programmatic application, especially for use in on-board measurements.

Further developments

However, the literature about the algorithmic implementation of rainflow counting is not finished with the discussion about three- and four-point algorithms. In [21] a recursive algorithm is described which regards the signal sequence as a structure in which elementary hysteresis loops (consisting of the sequence min - max - min) are nested into each other at any depth. The algorithm branches out deeper and deeper by calling the structuring function itself and divides the signal sequence into subsections. The breakdown stops, when it arrives at the elementary hysteresis loop. This creates a tree-like structure of the signal sequence, which is typical for recursive algorithms. The procedure only works for rearranged signals, because residues are not comprised in the approach of structuring the data into elementary cycles.

Following the idea that even more recent assessments such as [18] always mention the rainflow rules for the determination of cycles as an algorithm, there are also approaches to implement the native graphical flow rules as program code. The algorithm of Deng [22] is based on such an interpretation of the flow rules. However, the algorithm identifies whole loops at once: for example from a minimum, the position of the next smaller minimum is determined first. This is the longest possible extension of a course in the index direction. If this run length has more than three indices, a search for minima and maxima is carried out to identify closed loops. The algorithm also does not process residues.

The algorithm of Shinde et al. [23], on the other hand, calculates the courses of the rain flows. If the course starts from a minimum, the run length to the next smaller minimum is also determined. This path is then traced back and the maxima on the path are evaluated, to find a possible earlier end of the course. Thus, with this method should not be able to identify courses that run several times into each other (as, for example, between indices 5 and 6 in fig. 1): In such courses the maximum that determines the stop class is located before the start index of the course processed at the moment. This means the stop class is determined by a chronologically earlier course.

Further characteristics of the rainflow method

In general, the application of rainflow counting is not limited to mechanical fatigue strength. Rainflow counting is a method to decompose a complex signal whereby the order of extrema can be preserved, while the time reference is lost. The rainflow counting is also used for non-mechanical problems, for example in the fatigue analysis of semiconductor technology or the lifetime assessment of batteries.

Another advantage of the rainflow method is its inclusion of other counting methods. Since each identified hysteresis loop contains at least one original edge of the signal, other countings such as the level crossing counting or the peak counting can be derived from the rainflow result without a new counting process [4]. Provided that there are no residues, rainflow counting and range pair counting, which is also considered suitable for fatigue strength, come to the same result [3].

Conclusion from literature survey and reference to SCRF

The rainflow method can therefore be regarded as a particularly important counting method. Mostly rainflow counting codes to date implement the rainflow rules in three-point or four-point algorithms. Some approaches go in the direction of the graphical method, but do not cover all possible cases. The graphical method according to the flow rules may perhaps distract from the background of material mechanics [13], nevertheless it is not doubted, that the algorithm reliably identifies the hysteresis loops and residues.

The SCRF-algorithm presented here does not use difference criteria (and thus cannot be directly assigned to the three-point or four-point algorithms), but determines the rain flows as they are specified by the flow rules: For every starting point the algorithm calculates to which stop class a drop course extends and at which class and at which signal index it terminates. The latter is important for the further processing of the courses. In a subsequent step courses and counter-courses are matched together into cycles, optionally with the additional criterion, that the courses must also touch each other in the index direction. Since courses of the drops are determined individually from each reversal, the new algorithm is called SCRF (Single Course Rainflow) for a compact designation.

Preview of the content of the article

We start with some definitions. Then the pre-treatment of the raw signal required for the algorithm is described. The next step is the description of the flow rules that are then exactly implemented in the algorithm.

After that the procedure for the calculation of the courses is detailed. This is followed by an explanation by means of an example. In the next step the algorithm which pairs the courses and the counter-courses is explained. This also includes a description of how the residue vector is determined, if it is not empty. The procedure requires only one pass through the signal sequence chronologically in the index direction for the determination of the rainflow matrix and the residue vector.

In section 4 the verification of the algorithm is described. Some comparisons are also made to the three-point and four-point algorithms. In this context also an evaluation of the computational resources is done. Finally, recommendations for the treatment of residues are given.

2 Fundamental points and definitions

2.1 Preparation of the raw signal

A link to physical quantities is deliberately omitted in this description for greater generality. The time direction is called the index direction. Since the rainflow method can count arbitrary signals like load, stress or strain, the ordinate direction is termed the value direction or stress direction.

After a measurement the signal exists digitally in a sampled and thus discrete form with respect to time and value. The (unprepared) signal, which is usually sampled with a somewhat high resolution in value direction y, is divided into K classes. This classification coarsens the resolution, but at the same time also creates the probability of repetition of equal values and thus the possibility of counting. The classes are numbered with integers which are used as the new signal value. The result is now a sequence of class numbers.[1] According to [5], this sequence is also thought of as a vector.

In the next preparation step consecutive duplicates in the same class are removed. Afterwards in the final preparation step the reversals (local minima and maxima) are identified; signal values lying in between are deleted. With the deletions the indices move up in each case; no empty entries remain in the signal sequence.

Through the processing steps we obtain the (prepared) signal as a sequence of reversals in which relative minima and maxima alternate. The time axis is represented by the counting index as a natural number. The immediate reference to time is lost, but the chronological order of the reversals is still present.

Even if the reversals are discrete points in the representation of the data, we consider the connection between two reversals as an edge that can be occupied by rain flows. For the rain flows spreading over several edge sections, the term *drop course* or *course* is used. One could name the course of the rain flows also with the term *residue* or with the term *half cycle*. But the term residue is used in several meanings, which is why the previously unused word *course* is taken here. The half cycle is thus a downstream interpretation of the course.

2.2 Rules of rainflow counting

Courses begin each at a reversal which defines the start class and the start index. The courses run along edges or extend in the index direction. In this contribution, we do not use signal rotation. The *direction of fall* of a rain drop is the index direction and is to be considered here as a movement to the right. The following rules define when a course terminates.

[1] A class represents the samplings within that class. As a class is a range (and not a value), the number of classes are written at the center of the class at the diagram's ordinates. The lines represent the limits between the classes, i.e. a level crossing.

- If the course starts at a minimum, the course will stop as soon as a smaller or equally small minimum is reached.
- If the course starts at a maximum, the course will stop as soon as a greater or equally great maximum is reached.
- If a part of an edge is already occupied by an earlier course, the current course stops just before the occupied part.
- A courses ends when the signal sequence ends.

In addition to the start class and start index, the stop class and stop index are the further characteristic values of a course. Edges that start from a minimum are called *positive*. Correspondingly, edges that start at a maximum are termed *negative*.

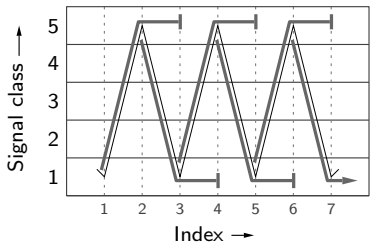

Fig. 5: Plain cycles with their rain flows

In most contributions the case of equality in the first two flow rules is not explicitly mentioned as a stopping condition of a course. Because of the classification of the signal the case of equality should be included. As justification a simple signal alternating between two values is given, cf. fig. 5. Here after every second reversal a loop is completed. Moreover, this approach is in agreement with the instructive example shown in [4].

2.3 Pairing the courses into closed loops

In general, the algorithm is that two opposite courses form a closed hysteresis loop [4]. In [18] an additional criterion is mentioned, namely that the courses must be adjacent to each other. This means that the stop index of the opening course must be greater than or equal to the start index of the closing course. Thus, if such two *adjacent* courses are opposite in start and stop class, they are paired to form a hysteresis loop.

The term *residue* is used in this contribution to denote a course, for which there is no closing counter-course in the rest of the signal. The vector containing all residues is called *residue vector*. The (continuous) residue vector can only be formed if the stop class of the preceding residue is equal to the start class of the succeeding residue.

The condition of adjacency is also implemented in the algorithm presented here. On the one hand it increases the algorithmic effort. But on the other hand it ensures that the resulting residues form a coherent residue vector at the end of the counting.

3 Mode of operation of the SCRF algorithm

The algorithm runs in two stages. In the first stage, the stop classes and the stop indices of the courses are calculated (course determination). In the second stage, courses belonging to a loop are paired to each other and counted into a matrix (pairing and counting), while courses without a counter-course are counted into a residue matrix. Finally the residue vector is composed.

3.1 General functioning of the course determination

In the prepared signal, positive and negative edges alternate. The courses that start on positive edges can only rise or remain constant. Accordingly the courses starting on negative edges can only decrease or remain constant. The terms *positive* and *negative* can consequently be applied to the courses. Positive and negative courses can be calculated separately, as they do not influence each other. This separation makes the algorithm clearer for explanation.

Any manipulations of the signal, e.g. a rearrangement for periodic signals, must be carried out before starting the calculation of the courses, as with other algorithms, cf. section 5.

In the following the algorithm is explained for the determination of *positive* courses. For each course start and stop class are calculated as well as the start and stop index. The jumps on the individual edges are not stored; they are not needed for pairing and counting.

For the set of all positive courses, an upper boundary indexed in coherence with the signal is carried along, which is the actual core idea of the algorithm. At the beginning, this boundary is set to a value above the largest possible class value, i.e. beyond the signal range.

In principle a positive course can climb along the edges over which it passes until it comes up against the mentioned boundary. By extending in the index direction a course pulls the upper boundary downwards and limits the rise of subsequent courses. Thus, within their way over the edges, subsequent courses can only rise up to the upper boundary. In this way, a course that flows into another course is stopped by the upper boundary. But it is also possible that a subsequent course spreads out in the index direction below the current boundary and in turn pulls the upper boundary further down for the courses following it.

The procedure uses the chronological order of the edges for processing. A variant in which the positive courses start from the minima of the smallest classes and then successively continue starting from the minima of the next smaller class can also be realised with the algorithm. This will be discussed in section 4.4.

The pulling down of the upper boundary is done only until the course terminates according to the flow rules. When the end of the course is detected, the stop class and the stop index are stored with reference to the index of the course's starting point in a vector. Start class and start index need not to be stored separately as they are available directly from the signal sequence.

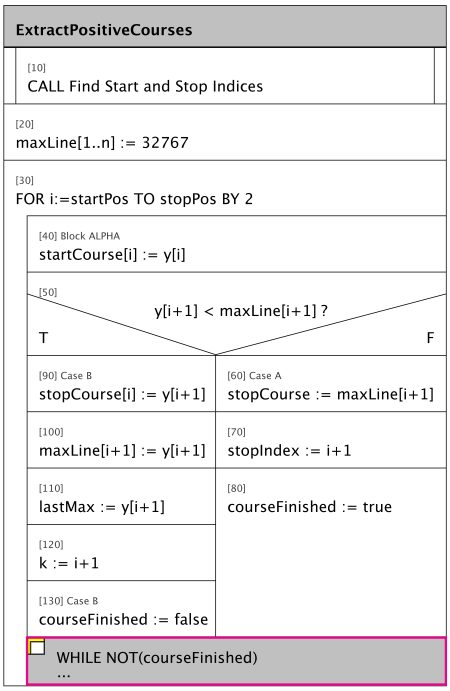

Fig. 6: Outer part of the structogram for the determination of courses

Since a course starts from the beginning of each edge, n values in the signal sequence result in $(n-1)$ calculated courses. The programmed implementation of the course determination is given in the form of structograms, i.e. in a form that can be directly transferred to a programming language with a structured paradigm such as Matlab, C or Delphi. For graphical reasons, the representation of the algorithm is divided into two figures. The last block in the fig. 6 contains the instructions of the second structogram in fig. 7.

For each course, a total of five cases are distinguished, which are designated from case A to case E. The cases A and B, which are shown in the structogram in fig. 6 refer to the initial edge, i.e. the first edge that a course passes through. Here it can happen in case A that the initial edge cannot be completely occupied, because an earlier course has seized part of its top before. This can be detected by a comparison with the boundary, which might have been pulled down by an antecedent course. If the initial edge cannot be passed through completely, the course determination is already finished; the stop class is then set to the value of the boundary. Otherwise, in case B, at least the end class of the initial edge is reached as stop class and the course calculation must be continued.

In the second structogram in fig. 7 the continuation of the course is treated, provided that the initial edge could be occupied completely. This part of the algorithm is a loop which we call the *course continuation loop*. During each pass of the loop a further negative and then a further positive edge are processed pairwise. Various conditions can cause the processing within the loop to finish earlier. The loop is then abandoned as a whole. This is always done via the boolean variable `courseFinished`.

First it is checked whether a negative edge follows at all or whether the signal ends. Other than these two cases are not possible for a negative edge along a positive course. If the negative edge exists, the stop class remains at its old value, because negative edges cannot influence the stop class for positive courses. The boundary is set to the memory value `lastMax`, that has been assigned during the treatment of the antecedent maximum.

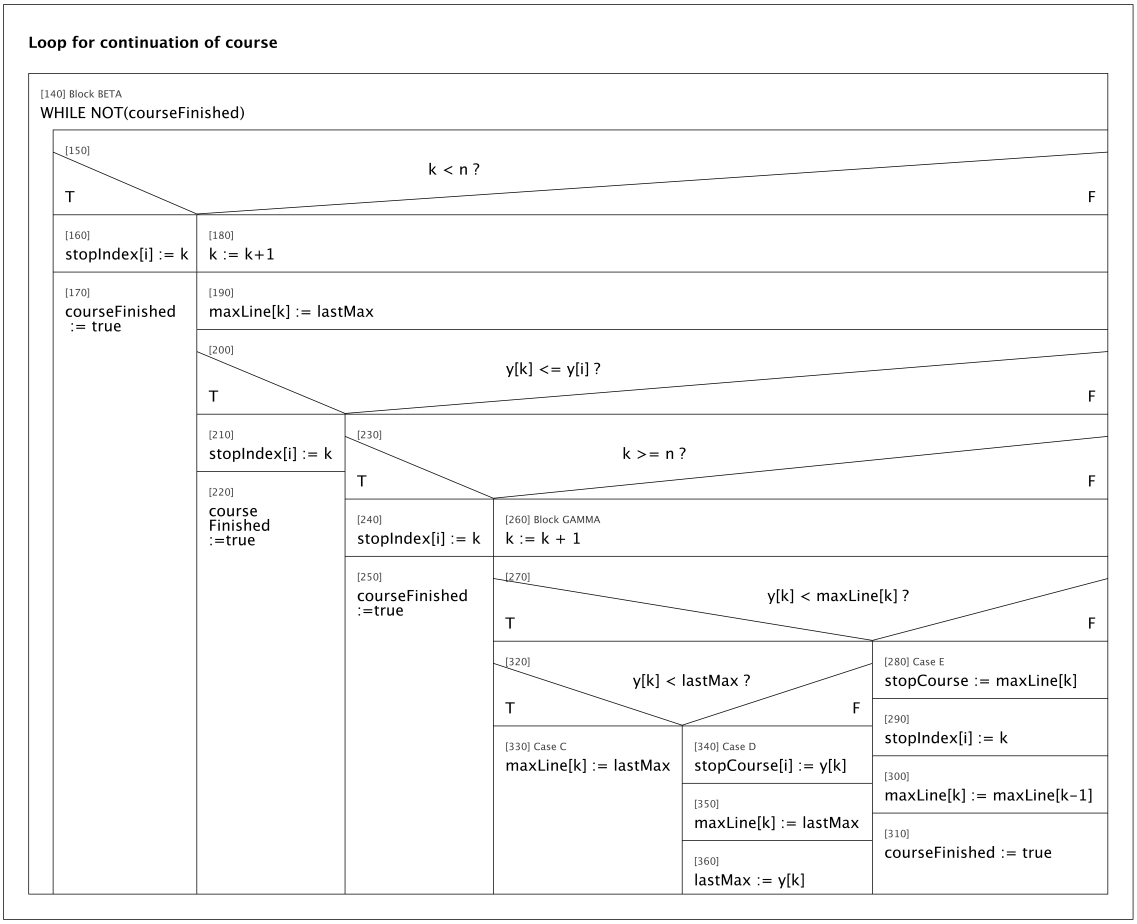

Fig. 7: Inner part of the structogram for the determination of courses: loop for the continuation of the course

The end of the negative edge is a minimum. If it is less than or equal to the minimum of the initial edge, the course terminates according to the flow rules and the course determination is completed; the stop class is already set from previous assignments. If the course is not completed yet the presence of the next positive edge is checked. If this positive edge exists in the signal, the detailed check takes place within the cases C to E.

If the next positive edge would reach the boundary, the case of running into an already existing course is present. A number of earlier courses may already have successively occupied the edge from top to bottom. In this case E the determination of the course is finished. It is important that the boundary line is set to the lower previous value of the antecedent edge rise in order to block the occupation for course to come next.

For the two remaining cases it must be clarified whether in case C the course is to be continued horizontally in index direction or whether case D occurs, which means that the full value of the forthcoming edge rise can be processed before the course continuation loop goes into the next pass. For this case D another special feature needs to be considered: the new course of the boundary results from the last valid class of the course, not from the final class of the treated positive edge. Even if in case D the full rise of the newly encountered edge can be done, the boundary is pulled down one index later. This is necessary so that chronologically later courses do enter this edge too far.

Reaching the end of the signal sequence terminates the course continuation loop as well as reaching an equal or smaller minimum. In addition, the course continuation loop can be stopped by case E, which handles running into an already partly occupied edge.

When the course is finished, the stop class is set unless the variable does not already contain the correct value from previous assignments. Furthermore, the stop index is set. The boolean variable `courseFinished` is used to stop further iterations of the course continuation loop.

3.2 Explanation of the course determination by means of an example

The mode of operation is described with reference to the signal shown in fig. 8. The hysteresis loops resulting from the individual stress values can be traced from the stress-strain curve shown on the right part of the figure.

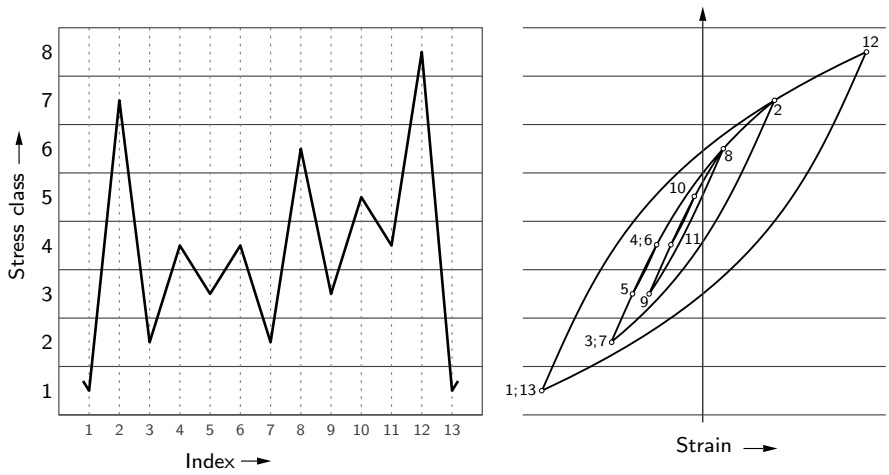

Fig. 8: Example of a stress history related to time for explanation

To simplify the explanation, again only the calculation of the positive courses, each starting at a minimum, is explicated. The current upper boundary is represented by full squares. Empty squares show the previous class value of the boundary. The initial value of the boundary is above the highest signal class to be reached by the signal.

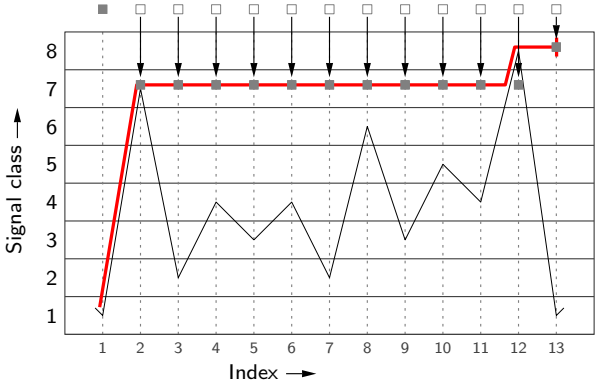

Fig. 9: Evolution of the course from index 1

In fig. 9 the first course in case B runs along the initial edge. Then it continues with the final class of the initial edge in case C. At index 12 case D occurs, since this edge can be executed to its top. For index 12, the boundary line must still remain at class 7 (coming from the ancedent edge rise, index 1 to 2), since a later course at this point can only rise to class 7. At index 13 the setting of the boundary is continued. The minimum at index 13, which is smaller or equal to the start minimum, stops the course continuation loop. The value 8 is stored as the stop class, the stop index is 13.

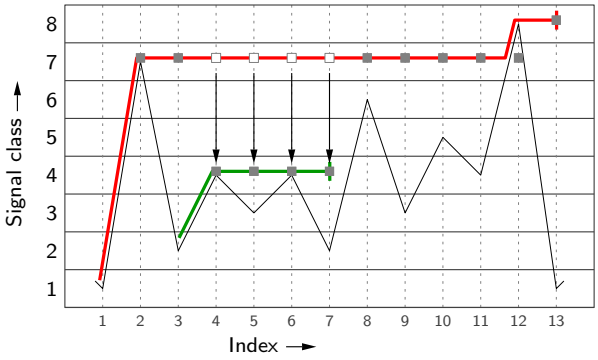

Fig. 10: Evolution of the course from index 3

Fig. 10 shows the positive course starting at index 3. It can propagate below the boundary that was pulled down by the first course. In doing so, it in turn pulls down the boundary further. The course takes way to the right in the case C and is stopped by reaching the equally small minimum in index 7.

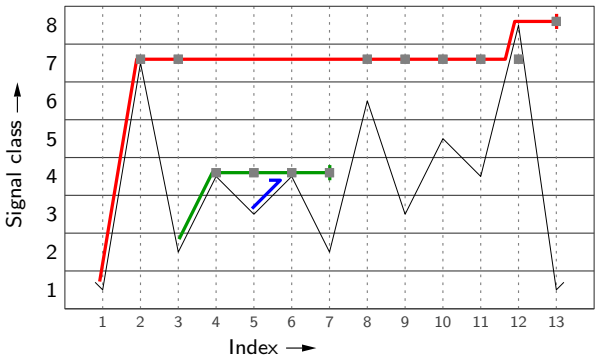

Fig. 11: Evolution of the course from index 5

Fig. 11 shows the positive course starting at index 5. It is stopped by the upper boundary at index 6 (case A) and terminates already on its initial edge.

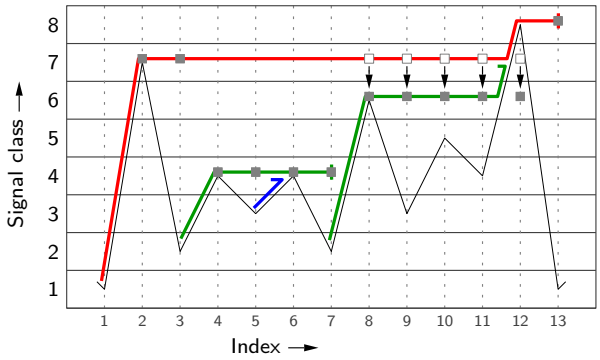

Fig. 12: Evolution of the course from index 7

The course starting at index 7 in fig. 12 can process the initial edge fully, then pulls down the boundary according to case C (keeping the class constant), and ultimately runs into the edge from index 11 to 12. The stop class finally reached in case E is controlled by the boundary at index 12. The boundary at index 12 is afterwards set to the value class 6 (from the antecedent edge rise, index 7 to 8) which is then reachable for subsequent courses.

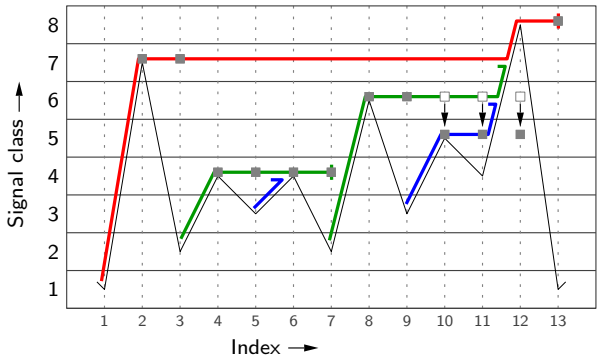

Fig. 13: Evolution of the course from index 9

Fig. 13 shows the positive course starting at index 9, which again is running into a partially occupied edge (case E) and adjusts the boundary for any subsequent courses.

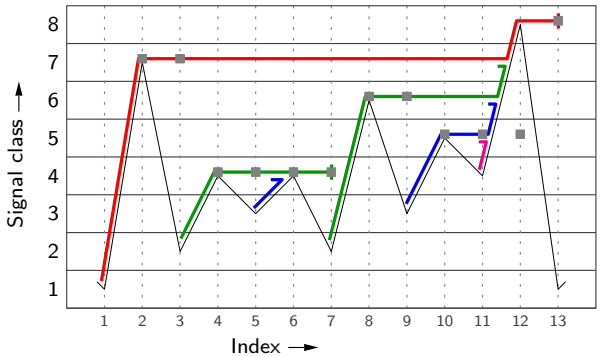

Fig. 14: Evolution of the course from index 11

In fig. 14, the course starting at index 11 cannot be executed completely; the rise is limited by the boundary at index 12. An adjustment of the boundary does not take place in this case A, because in case A invariably no further occupiable part of the edge remains.

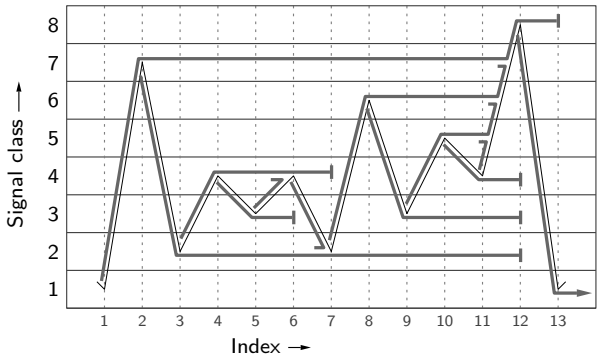

Fig. 15: Final result from the course determination

The determination of the negative courses, each starting from a maximum, could be performed in between the respective course determinations for the neighbouring positive courses. Alternatively the determination of the negative courses can be done after all positive courses are calculated. The result in both procedures will be identical. Fig. 15 shows the final result of the course determination for positive and negative courses.

3.3 Explanation of the pairing algorithm

The second section of the procedure is the pairing of courses with their *counter courses*, i.e. courses with opposite start and stop classes, in order to form a completed cycle. First, the simpler procedure without fulfilling the condition of adjacency is explained.

Matrices are used for counting: For closed loops this is the well-known rainflow matrix. The matrix is kept as a from-to matrix, storing the start class and stop class of the first half cycle of a hysteresis loop, as explained in [4].

The residues as remaining half-cycles also have a start and a stop class, so one can also count residues as well in a from-to matrix, the *residue matrix*. The entries in the residue matrix are not binary; it is possible that up to two residues of the same start and stop class can be counted. However, the number of duplicate residues is limited by the number of classes in the value direction.

Since some programming or scripting languages, such as Matlab, do not allow negative indices, an index shift is necessary: the value 1 is set as the index of the lowest class. Therefore the pairing routine additionally needs to return the signal value of the lowest class, so that the correct class value can be reproduced for both the rainflow and the residue matrix.

The algorithm runs through the list of start and stop classes from the course determination. Each course is at first counted into the residue matrix. Then it is checked whether the counter course has already been counted into the residue matrix. If this is the case, the indexed element in the rainflow matrix is incremented and the two entries in the residue matrix are decremented accordingly.

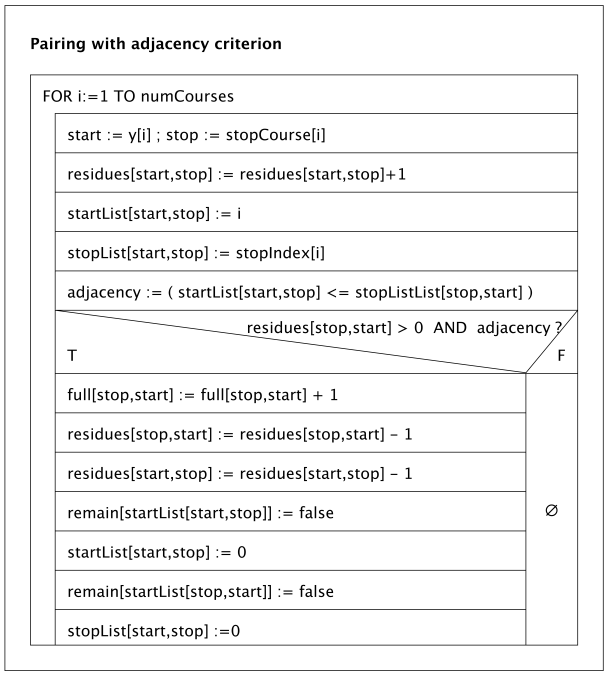

Fig. 16: Structogram of the pairing algorithm observing the rule of adjacency

After the end of the counting, the rainflow matrix and the residue matrix are returned. A residue vector is not calculated for this type of pairing because it is not necessarily coherent: it is possible that a residue stops in a class different from the class the next residue starts in.[2]

For a coherent residue vector, an additional condition must be fulfilled. This little-mentioned rule can be found in [18]. This *condition of adjacency* holds if the opening course reaches the start of the closing course.

In order to query the condition of adjacency, two further matrices indexed by the class numbers are kept, in which the last start index and the last stop index are stored. Only if the entry in the start list of the current course is equal or smaller than the entry in the stop list of the counter course and in fact there is already a counter course in the residue matrix, the above-mentioned increments in the rainflow matrix and decrements in the residue matrix are carried out. Fig. 16 shows the pairing algorithm with the condition of adjacency as a structogram.

The composition of the residue vector is prepared during the pairing algorithm. A vector of boolean values is created, where it is stored, which edges remain as residues. In this so so-called *remain vector* all values are initially set to *true*. For each identified cycle, the two positions where the constituting half-cycles are located are set to *false*. After completion, the remain vector is passed through. At each position where there is still a *true*, the corresponding edge from the original signal is assembled to build the residue vector.

[2] With the four-point algorithm the condition of adjacency is automatically met because both partial edges, from which the cycle is constituted, are deleted; the neighbouring edges move together. Consequently, coherent residue vectors result automatically there.

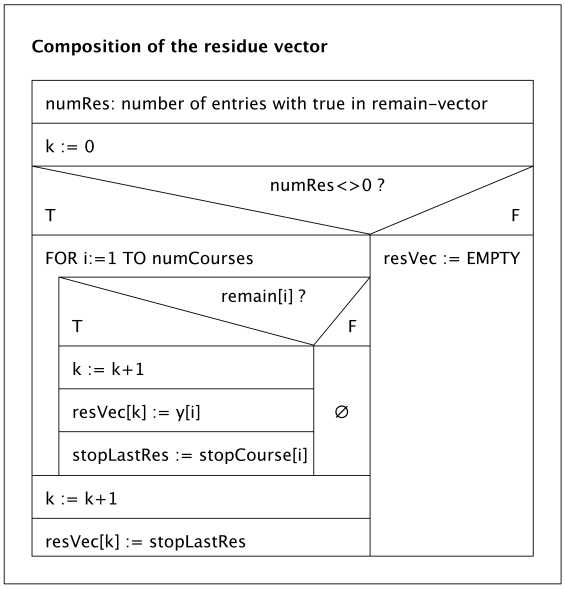

Fig. 17: Structogram of the reassembly of the residue vector

From the stress history in fig. 8 the algorithm finds in compliance with the condition of adjacency the cycles in table 1.

	Class		**Indices**	
No.	Start	Stop	Start	Stop
1	8	1	1	13
2	7	2	2	12
3	2	4	3	7
4	4	3	4	6
5	6	3	8	12
6	5	4	10	12

Tab. 1: Result of the counting of the sequence in fig. 15

4 Verification

The verification of the algorithmic implementation was carried out on the basis of many manually counted signal sequences. In all of these sequences the algorithm proved to be robust and correct with regard to the flow rules. Also an automated testing was programmed in order to test optimisations of the code quickly and extensively. The automated testing is additionally performed by negating the class numbers of the test sequences. This ensures that the processing of positive and negative courses is working symmetrically.

The comparisons are done with reference to the *from-to matrix*, which is also designated as the *full matrix*. If the sense of direction of the hysteresis loops is not taken into account and only the minimum and maximum of a cycle are used, one obtains the *half-matrix* [4] from the

full matrix. When hysteresis loops are mentioned down below by numbers in the following explanations, the numbers refer to the class values.

4.1 Detailed comparison to other counting algorithms

With the help of small examples it will be documented, how the SCRF algorithm works in comparison to the three-point algorithm according to [7] and the four-point algorithm according to [12]. For both algorithms also a reference is made to the special counting variants for periodic signals.

4.1.1 Single cycle

Fig. 18 shows the overall behavior of SCRF at an intermediate cycle on a larger edge. The assessment of the behavior at partial sequences is generally difficult, since all rainflow algorithms evaluate the preceding history of the signal. In addition the complete three-point algorithm according to [7] treats the edge containing the starting point in a special manner.

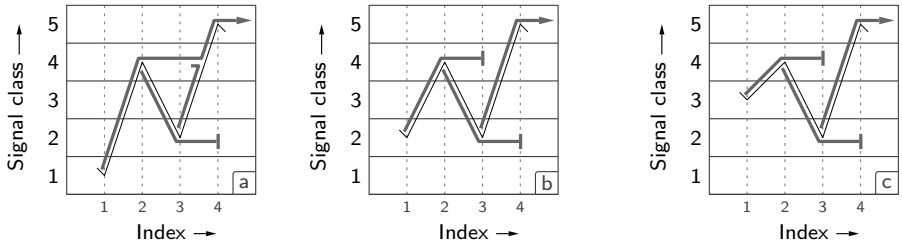

Fig. 18: Elementary signals for the comparison of the principle counting algorithms

In subfigure a, the cycle 4-2-4 is equally recognised by SCRF, by the three-point condition and by the four-point condition according to the equations (2) and (3), respectively. In subfigure b, the SCRF and the three-point condition recognise cycle 2-4-2, whereas the four-point condition identifies cycle 4-2-4. In the last subfigure c, the SCRF and the four-point condition do not detect any cycles, while the three-point condition identifies the cycle 4-2-4.

4.1.2 Consideration of the residue vector

As can be seen in fig. 19 the structure of the residue vector, given by equation (4), also applies to the SCRF counting. But there is one small difference: According to equation (4) the same absolute difference is possible twice in the middle of the residue vector. The SCRF combines such a constellation to a closed loop. Nonetheless the equation concerning the structure of the residue vector is the same for the four-point counting and the SCRF-counting. As shown in subfigure b of fig. 19, there remains another cycle in the middle. Such a residue vector with symmetrically arranged edges can be further reduced by repeated SCRF counting. If working without the condition of adjacency for the pairing of the courses according to section 3.3, such a symmetrical residue vector can even completely be transformed into closed loops.

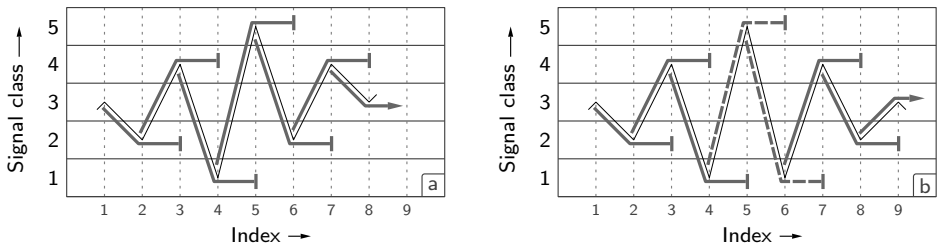

Fig. 19: SCRF-counting of a residue vector

The complete three-point algorithm according to [7] also does not count any cycles, although the three-point condition for the front part of the monotonically increasing absolute differences is met: In all cases the respective starting point is involved, because the edge just processed is deleted and thus creating a new starting point. Therefore the edges in the front part are only considered as half cycles and hence as residues. In the rear part (monotonically decreasing absolute differences) the three-point condition is not fulfilled. Therefore the algorithm counts only residues there as well.

As can be seen from fig. 20, the periodic invariance of the residue vector according to [10, 11] also holds for the SCRF method. That means, counting the multiplicated residue vector reproduces the residue vector again while the cycles in the interior are closed. Furthermore it can be noticed that doubling the residue vector for post-counting in four-point algorithms is a rearrangement, although it is carried out after the counting: the rearranged inner part can be counted as closed cycles, while the outer edges contain the self-reproducing residue vector.

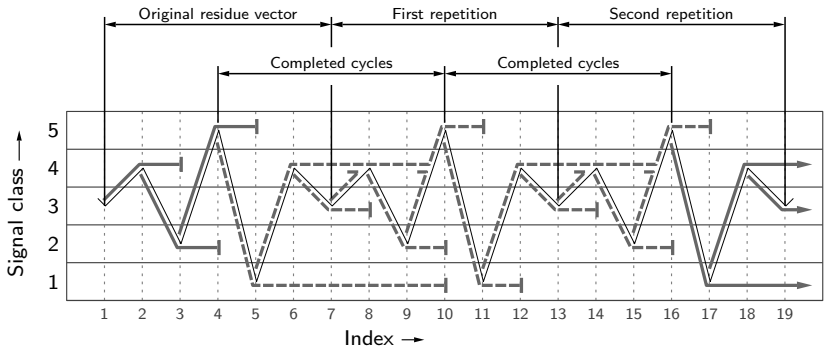

Fig. 20: SCRF-counting of a multiply appended residue vector: periodic invariance

The periodic invariance also applies for residue vectors with a symmetric arrangement as in fig. 19b, which contain a closable cycle in the middle, but there is a minor difference: if the residue vector is multiplicated m-fold in index direction, the same residue vector results as from counting the simple residue vector. The middle cycle (dashed lines) is detected by the SCRF method in contrast to the four-point condition and m-fold counted, the other cycles are counted only $(m-1)$-fold which is the same for the four-point algorithm.

4.1.3 Reversing a signal sequence in index direction

For the four-point counting it is described in [10], that the signal sequence flipped in the index direction is equivalent to the one formulated forward: The reversed sequence leads to the same half-matrix and to the same, but flipped residue vector. This property also applies to the SCRF algorithm.

Although the three-point condition in its definition does not have the symmetry of the four-point condition (cf. fig. 2), the same result is obtained here for the counting in forward and backward direction in terms of the half-matrix. So the counting is symmetrical with respect to the index direction for SCRF, four-point counting and three-point counting.

4.1.4 Signal sequence with the same initial and final value

This deals with signal sequences that begin and end with the same absolute extremum. Such signals arise in the case of periodic processes or by an (additional) rearrangement. These special sequences are counted here both with the general algorithm variants as well as with the special algorithm variants for the periodic counting.

As an example, the signal according to fig. 15 is used. The SCRF finds all cycles in the example, even under compliance with the condition of adjacency. The SCRF does not need to distinguish between several algorithm variants. The *simplified* three-point algorithm also finds the same closed cycles as the SCRF, but identifies the largest loop 1-8-1 with a reversed sense of direction. For the *complete* three-point method according to [7], the largest loop 1-8-1 results in two opposite half cycles. In the half-matrix the two residues counted as half cycles are arithmetically added up to a full cycle, although no closed loop is really detected. Hence concerning the half-matrix the SCRF on the one hand and the two three-point methods on ther other hand achieve the same results.

The four-point algorithm finds the same closed loops as the SCRF except for the largest loop 1-8-1, which is output as the residue vector [1 8 1]. The transition to the four-point method for a periodic signal and the associated recalculation of the doubly concatenated residue vector [1 8 1 8 1] finds the cycle missing from the initial counting, though with a reversed sense of direction. The residue vector which comes out again in the recalculation due to the periodic invariance [1 8 1] is then to be dropped. After the application of the periodic four-point counting, the same result for the SCRF and the periodic four-point method is obtained related to the half-matrix.

4.2 Comparison using random sequences

4.2.1 Comparison using a random *periodic* sequence

For the comparison, a normally distributed random signal with 999 reversals was generated. This signal was rearranged and then counted. For better comparability the number of classes of the signal was limited to eight classes. The comparison was carried out for several random sequences.

The SCRF algorithm counts the 499 cycles as expected, the residue vector is empty. With the simplified ASTM algorithm for periodic signals a few cycles in the full matrix differ in their sense of direction. Such cycles are found in the area of large cycle ranges. With respect to the half-matrix, the ASTM algorithm for periodic signals counts the same cycles.

In a four-point counting for a periodic signal it can be seen, that in some cases the doubly concatenated residue vector has to be post-processed as described in section 2.1 in order to obtain a sequence of reversals: If there is an even number of reversals in the residue vector, the number of edges is odd and thus the sense of direction of the first and last edge are the same. After the addition of this post-counting, the full matrix differs from case to case for some loops concerning the sense of direction. The deviations are located in the area of large cycle ranges. The proportion of these cases is clearly less than 1% of the cycles and is rapidly decreasing for longer sequences. Related to the half-matrix the SCRF and the four-point algorithm for periodic signals come to the same result.

4.2.2 Comparison using a random *non-periodic* sequence

Here a normally distributed random signal with 1000 reversals was generated, again the number of classes was limited to eight classes for better comparability. The comparison was also carried out for several random sequences.

The SCRF method produced residue vectors with few reversals and the structure of the equation (4), whereby no closable cycles were in the center. The maximum length of $(2K - 2)$ edges according to the equation (4) was reached at most by half. In order to make a comparison with the three-point counting according to [7], the residues were subsequently evaluated as half-cycles, cf. section 4.1.2. After that step there result small differences in the comparison at a few positions of the from-to rainflow matrix, which again are located in the areas with greater ranges in the cycles. After the addition of residues, each counted by 0.5, no differences can be observed with reference to the half-matrix.

The four-point counting yields residue vectors that in some cases contain cycles in their center, which are closed by the SCRF. If those cycles are counted manually and added to the four-point counting, both four-point counting and SCRF obtain the same half-matrix. The part of cycles detected by the four-point algorithm in the opposite sense of direction was about 1% of the total cycles in the signal, again decreasing with longer signals. This proportion is about twice as large as in the counting of the three-point counting, in each case related to the same signal.

4.3 Conclusion from the comparisons to three- and four-point algorithms

For rearranged signals the examined algorithms find the same cycles, if the sub-variants for the periodic signals are used for the three- and four-point counting. For the SCRF, no formulation in two variants is necessary. Furthermore a few cycles (mostly with large ranges) are identified by the three-point and four-point algorithm with a reversed sense of direction. So the equality between the methods is confined to the half matrix.

Also for non-periodic signals all three types of algorithms come to approximately the same result. There are differences in the identification of the sense of direction. The insensitivity of the four-point algorithm concerning the sense of direction is described in [10]. If the comparison is restricted to the half-matrix, the differences are minimal. With the four-point algorithm, one pass of counting is not sufficient; a recalculation with the doubly concatenated residue vector is necessary. Most striking is the effect that the residue vector according to the four-point method in its *center* often contains a cycle that the SCRF can close. Here lies a problem of the four-point algorithm: the algorithm cannot evaluate an (obvious) single elementary cycle consisting of three points, since always a group of four reversal points is needed to apply the four-point condition.

The ASTM algorithm for non-periodic signals in most cases gives the same result in the half matrix, even if a loop is not detected as closed: As the ASTM algorithm evaluates two opposite half cycles each by 0.5 the arithmetic addition in the half-matrix appears as the counting of a closed cycle.

4.4 Comparison of SCRF algorithm variants regarding course order

For the algorithms according to the flow rules in section 2.2, one can basically distinguish between several variants of the procedure. The question can be posed in which order the courses are drawn. It has already been stated above that positive and negative courses can be determined separately. Most references such as [3, 18, 20] suggest a chronological processing. LEE [16] describes an approach of starting the courses from the outer signal classes. The negative courses are first drawn from maxima in the largest class number, followed by the maxima in the second largest class number, and so on. For the positive courses, one starts with the minima in the smallest class number and then proceeds in the opposite way.

The presented SCRF-algorithm is suitable for both procedures. A comparative calculation has been performed with 10^6 random values and with $4, 8, \ldots, 64$ classes in the signal. No difference could be found between the two procedure variants in all calculation runs: All courses were the same in terms of class numbers as well as in terms of the run lengths. Therefore we prefer chronological processing for a clearer presentation.

4.5 Evaluation in terms of computer technology

The SCRF algorithm has undoubtedly higher requirements in terms of computer memory and processing time than the referenced three-point and four-point algorithms. But nowadays the memory concerns of earlier contributions are no longer relevant, since memory capacity has developed exponentially over the years, appositely predicted by the law of MOORE [24]. The use of the SCRF-algorithm is even unproblematic for long time series in the range of 10^7 and more reversals, which allows multi-channel measurements with long durations even at higher sampling rates.

The SCRF algorithm works on an integer basis with simple operations and is therefore by no means slow. The computing time is proportional to the length of the signal sequence. In

addition, the computing time is approximately proportional to the third root of the number of classes. This is because an increasing number of classes induces an increase in the length of the courses.

The number of classes for the signal can be arbitrarily large, but the use of a 16 bit integer resulting in a range from -32768 to $+32767$ should be more than sufficient. A positive integer is used for indexing. Here a length of 16 bit is not enough; a 32 bit unsigned integer is used, which allows more than 4 billions entries in the signal. By deploying these data types, the memory requirements can be optimised.

A disadvantage of the SCRF is that the signal must be kept in the computer memory as a whole, for a course can extend to the last index or may be closed with the last edge. Since the SCRF is not intended as a replacement for the proven and tested algorithms, a memory optimisation, e.g. forcing the end of a course as proposed in [20], seems unviable.

The real-time capability is repeatedly discussed in publications such as [5, 11, 19, 20]. The SCRF is not real-time capable, because the pairing of the courses to closed cycles is done after the calculation of the courses. But the deletion of data already being counted and the release of memory space are not really necessary nowadays. As a substitute for a real-time evaluation, a preliminary counting can be made during a measurement with the data collected up to that moment.

5 Treatment of residues

Typical techniques for reducing or avoiding residues, as described in [18] or [6], can also be realised with the SCRF. The algorithm for determining the courses is anyway upstream to the question of the treatment of residues: SCRF only deals with half-cycles before the courses are going to be paired. Not until that second stage the decision about closed cycles or residues is made. The recent proposals for treating residues are commented below together with a reference to the SCRF algorithm.

1) Rearrangement of the signal sequence, so that the sequence starts with an absolute maximum or an absolute minimum and ends with the same value. This means that the sequence treated as a periodic signal which has no residue by its nature. An application of the method to non-periodic signal means to forcibly close residues to closed cycles.

   In four-point counting, this result is obtained by doubling the residue vector, its post-counting and the addition to the first counting result.

   Rearrangement, especially for the treatment of periodic signals, is of course also possible with the SCRF method; this kind of modification of the signal is independent of the counting algorithm as it is done before the counting.

2) Prepending the residue vector to a continuation of the signal. This is done in the expectation that the residues of the vector can be closed with counter-half-cycles in the continuation.

This procedure proposed in [13, 18] could be performed easily possible with the SCRF. Of course, it must be ensured that the concatenated sequence is a reversal point sequence; if necessary, the signal must be cleaned at the concatenation point according to section 2.1.

3) Recounting of the residue vector using a different counting algorithm.

   This procedure proposed in [18] does not really represent a further *method of the treatment of residues. For the four-point algorithm and SCRF, the residue vector according to equation (4) does not contain any further cycles. The three-point algorithm does not identify any further cycles either, but counts the remaining edges as half cycles. So this proposal is with the recent counting methods equivalent to proposal no. 4.*

4) Counting residues as half cycle, what makes an implicit statement towards the assessment of the damage by residues.

   This method is possible for the SCRF, regardless of whether the courses are paired with or without the condition of adjacency. The simplest way to perform this operation is by adding the residue matrix multiplied by the factor 0.5 *to the rainflow matrix.*

The SCRF provides another method of closing some of the cycles by eliminating the condition of adjacency and pairing each half-cycle into a loop immediately after the opposite half-cycle occurs.

6 Conclusion

The presented SCRF algorithm counts physical quantities according to the graphical rainflow method. The algorithm works with a boundary against which the courses can strive. The course currently being calculated sets the boundary for chronologically subsequent courses.

From the course determination result the start and stop class of each course as well as the start and stop index. In a second step the courses with opposite start and stop classes (course and counter-course) are paired into closed loops. The criterion that the courses must touch each other for pairing is also implemented.

The SCRF algorithm works with signal classes and independently of a zero point position. The algorithm has no implicit evaluation of damage for the unclosed loops. These are output as residues. The output is available as residue matrix or as residue vector.

The SCRF produces almost the same results as the three-point counting and the four-point counting. In particular with regard to the counting results without evaluation of the sense of direction of the loops (in the half-matrix), nearly no differences can be observed. Conversely it is also confirmed that the known three-point and four-point algorithms do in fact represent the graphical Rainflow algorithm. The SCRF algorithm is better at correctly recognising the sense of direction of hysteresis loops because of its chronological approach.

A distinction in algorithm variants for periodic and non-periodic signals is not necessary for the SCRF. A rearrangement of the signal is dispensable unless periodic signals are expressly evaluated.

A further advantage of the SCRF is that the order of the hysteresis loops can be reconstructed if requested, as the start and stop index of each course are known after their determination.

The SCRF needs more computer memory than three- and four-point algorithms, but this should not be a problem anymore with the actual computer hardware. The processing time is somewhat greater. The algorithm is not real-time capable, as the signal must be held completely in the computer memory at one time. Nevertheless, a preliminary counting of the history accumulated up to that moment can be made during a measurement.

The algorithm shall and will not replace the proven and tested algorithms, but can be used as a supplement if required.

References

[1] Matsuishi, M.; T. Endo: *Fatigue of metal subjected to varying stress.* Fukuoka Japan Society of Mechanical Engineers (1968).

[2] Murakami,Y.: *The rainflow method in fatigue. The Tatsuo Endo Memorial Volume.* Butterworth-Heinemann (1991). https://doi.org/10.1016/C2013-0-04533-1

[3] Dowling, N. E.: *Fatigue Failure Predictions for Complicated Stress-Strain Histories.* Journal of Material, JMLSA, Vol. 7, No. 1 (1971), 71-87.

[4] FVA: *Zählverfahren zur Bildung von Kollektiven und Matrizen aus Zeitfunktionen.* FVA-Richtlinie 131 IV, Forschungsvereinigung Antriebstechnik (2010).

[5] Okamura, H.; S. Sakai; I. Susuki: Cumulative Fatigue Damage under Random Loads. Fatigue of Engineering Materials und Structures, Vol. I (1979), 409-419. https://doi.org/10.1111/j.1460-2695.1979.tb01328.x

[6] Downing, S.D.; D.F. Socie: *Simple rainflow counting algorithms.* International Journal of Fatigue 4 (1982), 31-40. https://doi.org/10.1016/0142-1123(82)90018-4

[7] ASTM E1049-85: *Standard practices for cycle counting in fatigue analysis* (1985), reapproved 2011. https://doi.org/10.1520/E1049-85R11E01

[8] Endo, T.: *Review on life prediction for complex load versus time histories. Critical review on rainflow algorithm.* Transactions of the Japan Society of Mechanical Engineers. A 54 (1988), 869-874.

[9] Anzai, H.: *Algorithm of the rainflow method.* In: [2], 11-20. https://doi.org/10.1016/B978-0-7506-0504-5.50010-2

[10] Brokate, M.; K. Dreßler; P. Krejcí: *Rainflow counting and energy dissipation for hysteresis models in elastoplasticity.* European Journal of Mechanics, A/Solids 15 (1996), 705-737.

[11] Beste, A.; M. Brokate; K. Dreßler: *Kann man berechnen, wie lange ein Auto hält?* In: Bachem, A.; M. Jünger ; R. Schrader, R. (eds): Mathematik in der Praxis. Springer, Berlin, Heidelberg (1995), 3-24. https://doi.org/10.1007/978-3-642-79763-7_1

[12] Amzallag, C.; J. P. Gerey; J. L. Robert; J. Bahuaud: *Standardization of the rainflow counting method for fatigue analysis.* International Journal of Fatigue 16 (1994), 287-293. https://doi.org/10.1016/0142-1123(94)90343-3

[13] Clormann, U. H.; T. Seeger: *Rainflow-HCM. Ein Zählverfahren für Betriebsfestigkeitsnachweise auf werkstoffmechanischer Grundlage.* Stahlbau 55, Heft 3 (1986), 65-71.

[14] Hong, N.: *A modified rainflow counting method.* International Journal of Fatigue 13 (1991), 465-469. https://doi.org/10.1016/0142-1123(91)90481-D

[15] Anthes, R. J.: *Modified rainflow counting keeping the load sequence.* International Journal of Fatigue 19 (1997), 529-535. https://doi.org/10.1016/S0142-1123(97)00078-9

[16] Lee, Y.; M. E. Barkey; H. Kang: *Metal fatigue analysis handbook.* Butterworth-Heinemann (2012), 89-113. https://doi.org/10.1016/C2010-0-66376-0

[17] McInnes, C.; P. A. Meehan: *Equivalence of four-point and three-point rainflow cycle counting algorithms.* International Journal of Fatigue 30 (2008), 547-559. https://doi.org/10.1016/j.ijfatigue.2007.03.006

[18] Marsh, G.; G. Wignall; P. R. Thies; N. Barltrop; A. Incecik; V. Venugopal; L. Johanning: *Review and application of Rainflow residue processing techniques for accurate fatigue damage estimation.* International Journal of Fatigue 82 (2016), 757-765. https://doi.org/10.1016/j.ijfatigue.2015.10.007

[19] Glinka, G.; J. C. P. Kam: *Rainflow counting algorithm for very long stress histories.* International Journal of Fatigue 9, Nr. 3 (1987), 223-228. https://doi.org/10.1016/0142-1123(87)90025-9

[20] Antonopoulos, A.; S. D'arco; M. Hernes; D. Peftitsis: *Challenges and Strategies for a Real-Time Implementation of a Rainflow-Counting Algorithm for Fatigue Assessment of Power Modules.* 2019 IEEE Applied Power Electronics Conference and Exposition (APEC) (2019), 2708-2713. https://doi.org/10.1109/APEC.2019.8722284

[21] Jiang, D.: *A Sequence Retainable Iterative Algorithm for Rainflow Cycle Counting.* SAE International Journal of Materials and Manufacturing 7 (2014), 108-114. https://doi.org/10.4271/2013-01-9091

[22] Deng, Q.; H. J. Yuan: *The Algorithm for Graphic Method of Rain-Flow Counting in Programming.* Advanced Materials Research 225-226 (2011), 1157 - 1161. https://doi.org/10.4028/www.scientific.net/AMR.225-226.1157

[23] Shinde, V.; J. S. Jha; A. Tewari; S. Miashra: *Modified Rainflow Counting Algorithm for Fatigue Life Calculation.* Proceedings of Fatigue, Durability and Fracture Mechanics, Springer Singapore (2018), 381-387. https://doi.org/10.1007/978-981-10-6002-1_30

[24] Moore, G.E.: *Cramming more components onto integrated circuits.* Electronics 38 Nr. 8 (1965).

All figures created by the author.