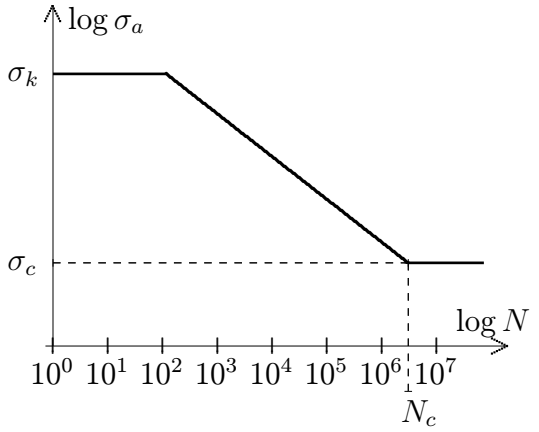
VÍCEKANÁLOVÉ SLEDOVÁNÍ KUMULACE POŠKOZENÍ V REÁLNÉM ČASE

Miroslav Balda [1]

1 Kumulace poškození

Závislost amplitudy harmonické odezvy – napětí σ_a – na počtu cyklů N do lomu součásti vynesená do diagramu obvykle s logaritmickými stupnicemi se nazývá Wöhlerovou křivkou anebo S-N křivkou v anglicky psané literatuře. Šikmá větev křivky, shora smluvně omezená mezí kluzu σ_k a zdola mezí únavy σ_c, je popsaná rovnicí



$$\frac{N_2}{N_1} = \left(\frac{\sigma_{a1}}{\sigma_{a2}}\right)^w \qquad (1)$$

Wöhlerova křivka představená na prvním obrázku se běžně konstruuje pro harmonické napětí s nulovou střední hodnotou. Není překvapující, že nenulové střední napětí σ_m ovlivňuje počet harmonických cyklů do porušení. Označíme-li nyní počet cyklů do porušení při amplitudě σ_a a středním napětí σ_m jako $N_m = N(\sigma_m)$, lze střední úsek (přímkový v logaritmických souřadnicích) vyjádřit vztahem vyplývajícím z podobnosti k rov. (1):

$$N_m\, \sigma_a^{w_m} = konst \qquad (2)$$

Exponent $w_m = w\,(\sigma_m)$ je charakteristický pro daný materiál, jeho technologické zpracování, konstrukční vruby a obecně i pro střední napětí σ_m. Lze očekávat, že budou jak mez únavy $\sigma_{cm} = \sigma_c\,(\sigma_m)$, tak i jí odpovídající počet cyklů $N_{cm} = N_c(\sigma_m)$ do porušení funkcemi středního napětí. Pokud bychom za tyto funkce zvolili polynomy v σ_m, mohli bychom materiálové vlastnosti shrnuté do vektoru $\boldsymbol{p}_m = [w_m, N_{cm}, \sigma_{cm}]^T$ vyjádřit jako

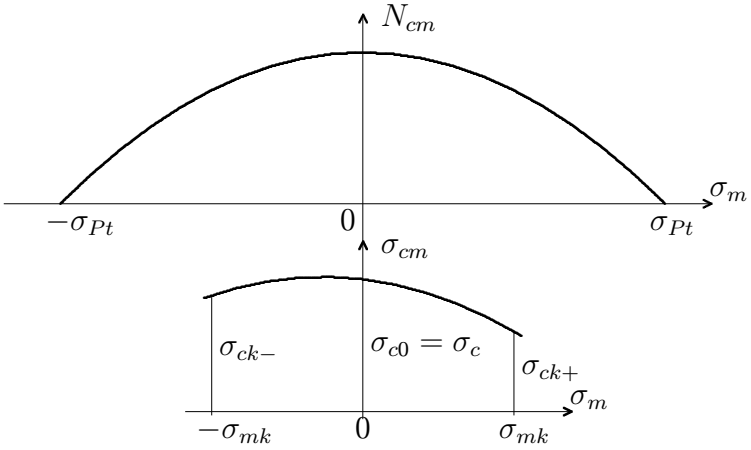
$$\boldsymbol{p}_m = \boldsymbol{C}_m\, \boldsymbol{\sigma}_m\ , \qquad (3)$$

kde

$$\boldsymbol{C}_m = \begin{bmatrix} c_{w0} & c_{w1} & c_{w2} & \dots \\ c_{N0} & c_{N1} & c_{N2} & \dots \\ c_{\sigma 0} & c_{\sigma 1} & c_{\sigma 2} & \dots \end{bmatrix} \qquad \text{resp.} \qquad \boldsymbol{\sigma}_m = \begin{bmatrix} 1 \\ \sigma_m \\ \sigma_m^2 \\ \vdots \end{bmatrix} \qquad (4)$$

[1]Doc. Ing. Miroslav BALDA, DrSc., Západočeská univerzita - ÚFY, Veleslavínova 11, 301 14 Plzeň

Prvky matice $\boldsymbol{C}_m$ jsou materiálové konstanty. Pro praktické účely nemůže být stupeň polynomu příliš vysoký, aby byl počet materiálových konstant (prvků v $\boldsymbol{C}_m$) únosný. Experimenty napovídají, že se w_m se středním napětím σ_m příliš nemění. Naproti tomu N_{cm} a σ_{cm} jsou na středním napětí σ_m závislé výrazněji. Počet cyklů do lomu $N_c(\sigma_m)$ je znázorněn v horním obrázku.



První závislost je patrně symetrická, neboť napětí na mezi pevnosti v tahu i tlaku se považují za stejné, a to při počtu cyklů $N_{Pt} = 0.5$, protože k porušení dojde v obou případech v polovině prvního zatěžování (bez odlehčovací fáze). Druhá závislost σ_{cm} je obvykle nesymetrická, protože mez únavy pro tlakové střední napětí je vyšší než při tahovém napětí. Vyjádřena graficky nese název Haighův diagram. Ať lze obě závislosti aproximovat kvadratickými parabolami. Potom matice $\boldsymbol{C}_m$ bude mít při aproximaci $N_{Pt} \approx 0$ tvar

$$\boldsymbol{C}_m = \begin{bmatrix} w & 0 & 0 \\ N_c & 0 & -N_c/\sigma_{Pt}^2 \\ \sigma_c\ , & (\sigma_{ck+} - \sigma_{ck-})/(2\sigma_{mk})\ , & (\sigma_{ck+} + \sigma_{ck-} - 2\,\sigma_c)/(2\sigma_{mk}^2) \end{bmatrix} \tag{5}$$

V matici $\boldsymbol{C}_m$ se objevují jen dvě nové materiálové konstanty, totiž napětí na mezi únavy $\sigma_{ck+} = \sigma_c(\sigma_{mk})$ při špičkovém tahovém napětí $\sigma_{mk} + \sigma_{ck+} = \sigma_k$ tj. napětí na mezi kluzu v tahu σ_k a $\sigma_{ck-} = \sigma_c(-\sigma_{mk}) - \sigma_k$ a při stejném špičkovém tlakovém napětí.

Za míru relativního poškození při amplitudě napětí σ_{ai}, středním napětí σ_{mj} a počtu provedených zatěžovacích cyklů n_{ij} slouží veličina

$$d_{ij} = n_{ij}/N_{ij}\ , \tag{6}$$

kde N_{ij} je mezní počet cyklů do lomu při zmíněných napětích. Pokud zatěžovací proces není prostý harmonický, ale generuje cykly o různých napětích σ_{ai} a σ_{mj}, je mírou relativního poškození veličina

$$D = \sum_i \sum_j d_{ij} \tag{7}$$

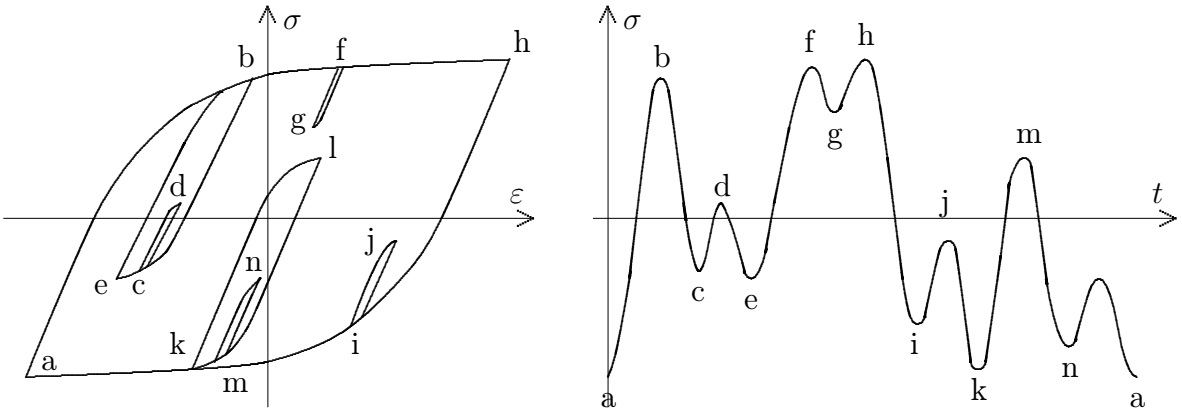
Zde je zapotřebí dát důraz na slovo "cykly", protože jen uzavřený (úplný) zatěžovací cyklus vytváří uzavřenou hysterezní smyčku na čele trhliny, v níž se předá definované množství energie do plastické deformace – poškození. Předpis rov. (7) se nazývá zákonem lineární kumulace poškození a je spojen se jmény jeho autorů – Palmgrena a Minera.

Je zřejmé, že pokud by tento tzv. zákon lineární kumulace poškození skutečně platil, došlo by k lomu součásti při $D = 1$. Experimenty ukazují, že k poruše dochází i pro $D \neq 1$. Odchylky lze vysvětlit skutečností, že proces porušování je podstatně složitější než předpokládá model. Přesto však tento zákon stojí v pozadí většiny hypotéz kumulace poškození vypracovaných později řadou jiných autorů, kteří se snažili o dosažení lepší shody modelu se skutečností.

Důvodů pro někdy i značné odchylky D od jedničky je celá řada. Především sám jev poškozování není lineární. Po relativně dlouhou dobu zatěžování probíhají v materiálu latentní změny, během kterých se vyčerpává jeho odolnost ke vzniku trhliny, která se potom šíří průřezem součásti. Dalším zdrojem odchylek od linearity je vlastní proces porušování. Trhlina se totiž šíří materiálem, který byl lokálně zplastizován v důsledku silné koncentrace napjatosti v okolí jejího čela. Zplastizovaný objem závisí opět nelineárně na zatížení. A právě tato skutečnost byla důvodem proč řada přístupů pro hodnocení únavové životnosti částí namáhaných náhodnými zatíženími selhávala, a to vinou různého výkladu pojmu "cykl" nebo "rozkmit".

2 Metoda stékání deště (rain-flow)

V roce 1968 na jakémsi oblastním zasedání japonské společnosti strojních inženýrů odezněl referát pánů Matsuishi a Endo [1], který s ohledem na jazykovou bariéru se stal známý až po několika letech, když pan Dowling opublikoval jejich metodu nazývanou poeticky "rain-flow method" v angličtině [2]. Takřka současně s ní se objevila v Rusku metoda nazývaná "metoda plných cyklů". Cílem obou metod je dekompozice složitého procesu na do sebe vnořené ukončené cykly odpovídající uzavřeným hysterezním smyčkám zatěžovaného dílu. Plocha uzavřená každou i dílčí hysterezní smyčkou v diagramu σ-ε představuje energii, která se spotřebovala na plastické deformace materiálu a tedy k jeho poškození.



Obě metody mnohem lépe interpretují fyzikální podstatu děje než metody jiné. Z tohoto důvodu také využití uzavřených napěťových cyklů získaných dekompozicí zatěžovacího procesu k odhadu poškození podle zákona lineární kumulace dává podstatně lepší výsledky než s použitím jiných metod. Problémem ovšem bylo programové zvládnutí metody.

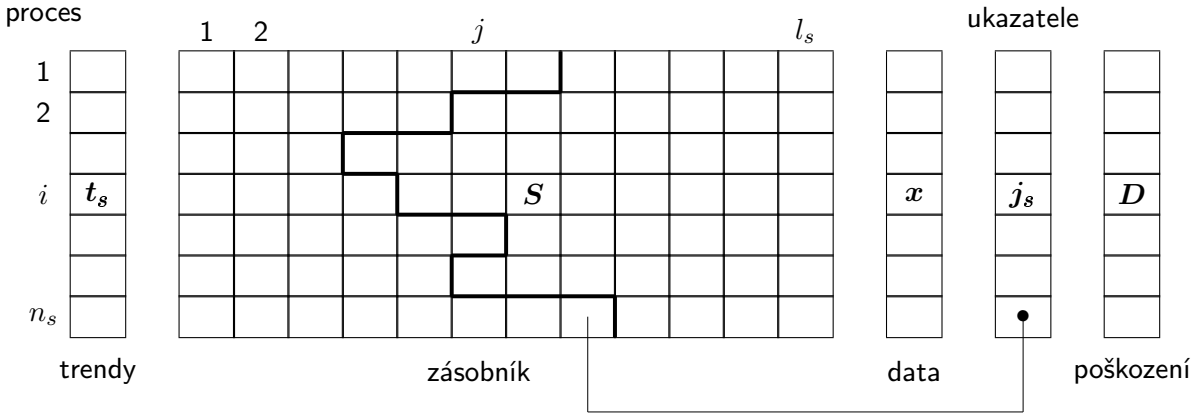
Před 20 lety se u nás znal pouze slovní popis metody říkající kdy voda na střeše pagody (reprezentované grafem zatěžovacího procesu otočeným o 90°) teče, kdy ukapává a kdy se zaráží [3]. Není proto divu že se i pochybovalo, zda je tento postup vůbec algoritmizovatelný [4]. Autor tohoto příspěvku se tehdy úspěšně pokusil o dekompozici signálu v úplné cykly zcela novou metodou, která na rozdíl od ruské byla jednoprůchodová. Navíc však ji bylo možno použít i v reálném čase, aniž bylo zapotřebí zpracovávat jen

záznam zatěžovacího procesu mnohonásobným průchodem programem, který započítával a odstraňoval uzavřené cykly s nejmenším rozkmitem, jako tomu bylo u ruské metody. Jaké pak bylo jeho překvapení, když zjistil, že jím realizovaná metoda [5] započítává cykly a půlcykly přesně stejně jako metoda japonská.

Metoda byla na rozdíl od originálu rozšířena ještě o frekvenční analýzu cyklů. Jejím výsledkem byly tříparametrické histogramy půlcyklů vytvořené v reálném čase, z nichž se následně vyhodnocovalo poškození. Výhodou frekvenční analýzy byla možnost posuzování účinků rezonančních kmitů na proces poškozování. Uvedený algoritmus používal na minipočítači S90 pro jednokanálové zpracování náhodných provozních napětí buď z digitalizovaných záznamů v režimu off–line, anebo i v reálném čase v průběhu jejich vzorkování. Při tom maximální vzorkovací frekvence mohla být až 2000 Hz. Program s tímto algoritmem sloužil po celou životnost počítače S90. Od jeho zrušení se využívá jeho jednodušší varianta pro osobní počítače.

3 Mnohokanálová dekompozice procesů

V poslední době se vynořila potřeba vyhodnocovat v reálném čase poškození v mnoha místech současně. Bylo proto zapotřebí upravit stávající postup pro možnost současného zpracování řady signálů. Při tom nebyl vznesen požadavek na frekvenční analýzu poškozovacího procesu. S ohledem na ohromné paměťové nároky histogramů pro serii signálů, upustilo se i od jejich načítání. Potom ihned po nalezení plného (uzavřeného) cyklu se vypočítá jím vyvolané poškození. Situace je však proti jednokanálové verzi komplikovanější o skutečnost, že uzavřené cykly vznikají v různých místech a tím i procesech v nestejných časech a že je tedy třeba analyzovat každý proces samostatně, i když paralelně s ostatními procesy. Nový algoritmus vícekanálového zpracování procesů metodou rain–flow využívá vektory a matici, které jsou přehledně uvedeny v obrázku.



Všechny vektory jsou dimenze n_s, kde n_s je počet signálů – procesů ke zpracování. Vektor $\boldsymbol{t_s}$ obsahuje trendy procesů indexovaných $1 \leq i \leq n_s$. Vektor $\boldsymbol{x}$ obsahuje vždy poslední vektor vzorků odebraných z měřených procesů, kdežto vektor $\boldsymbol{D}$ velikosti průběžně kumulovaných poškození. Matice $\boldsymbol{S}$ typu (n_s, l_s) slouží jako zásobník extrémů, které dosud nevytvořily úplné cykly. Vektor $\boldsymbol{j_s}$ indexů posledně obsazených sloupců matice $\boldsymbol{S}$ pro jednotlivé procesy. Počet l_s sloupců matice $\boldsymbol{S}$ musí být dostatečný pro uložení nejdelší posloupnosti nezpracovaných extrémů. Vlastní analýza extrémů procesů se pak dá provést podle následujícího **algoritmu**:

1. **Příprava zpracování**:
   Odeberou se dva datové vektory z n_s procesů do prvních dvou sloupců zásobníkové paměti $\boldsymbol{S}$, do vektoru $\boldsymbol{j_s}$ se dosadí do všech prvků index 2 a do i-tého prvku vektoru trendů $\boldsymbol{t_s}$ se dosadí pro $1 \leq i \leq n_s$

   $\boldsymbol{t_s}$(i) = 1 (true), je-li $S_{i2} > S_{i1}$, pro následné hledání maxima, anebo
   $\boldsymbol{t_s}$(i) = 0 (false), je-li $S_{i2} <= S_{i1}$, pro následné hledání minima.

2. **Cykl $\boldsymbol{n}$ − vzorkování** pro $1 \leq n \leq N$
   Odvzorkuje se n-tý datový vektor $\boldsymbol{x}$ z n_s procesů.

3. **Cykl $\boldsymbol{i}$ − signálů** o indexech $1 \leq i \leq n_s$:
   $j = \boldsymbol{j_s}(i)$

   a) Je-li $\boldsymbol{t_s}(i) > 0$, hledej MAXIMUM:
      Je-li $S_{ij} > \boldsymbol{x}(i)$, je maximum nalezeno a pokračuj:
      α) $\boldsymbol{t_s}(i) = NOT\ \boldsymbol{t_s}(i)$ pro přípravu hledání minima
      β) **Cykl $\boldsymbol{j}$ − extrémů** procesu i
      - Pokud jsou v zásobníku pro i–tý signál alespoň 4 extrémy, (t.j pro $\boldsymbol{j_s}(i) > 3$), hledej plný cyklus:
      - Je-li $S_{ij} < S_{ij-2}$, přeruš cykl $\boldsymbol{j}$ a pokračuj od γ). Jinak
      - vypočti poškození vyvolané plným cyklem s extrémy S_{ij-2}, S_{ij-1}.
      - Sniž $\boldsymbol{j_s}(i)$ o 2 (tj. vypusť nalezený plný cykl)

      Obrátka cyklu $\boldsymbol{j}$
      γ) Zvětši j o 1, dosaď $j_{si} = j$ a pokračuj od c).

   b) Jinak hledej MINIMUM:
      Je-li $S_{ij} < \boldsymbol{x}(i)$, je minimum nalezeno a pokračuj:
      α) $\boldsymbol{t_s}(i) = NOT\ \boldsymbol{t_s}(i)$ pro přípravu hledání minima
      β) **Cykl $\boldsymbol{j}$ − extrémů** procesu i
      - Pokud jsou v zásobníku pro i–tý signál alespoň 4 extrémy, (t.j pro $\boldsymbol{j_s}(i) > 3$), hledej plný cyklus:
      - Je-li $S_{ij} > S_{ij-2}$, přeruš cykl $\boldsymbol{j}$ a pokračuj od γ). Jinak
      - vypočti poškození vyvolané plným cyklem s extrémy S_{ij-2}, S_{ij-1}.
      - Sniž $\boldsymbol{j_s}(i)$ o 2 (tj. vypusť nalezený plný cykl)

      Obrátka cyklu $\boldsymbol{j}$
      γ) Zvětši j o 1 a dosaď $j_{si} = j$

   c) Ulož prvek datového vektoru do zásobníku $S_{ij} = \boldsymbol{x}_i$

   d) Zvětši index i o 1.

   Obrátka cyklu $\boldsymbol{i}$ − signálů

4. Inkrementuj počet n odvzorkovaných datových vektorů $\boldsymbol{x}$.

   Obrátka cyklu $\boldsymbol{n}$ − vzorkování

5. Nakumuluj poškození od klesajících cyklů uložených v zásobníku po skončení vzorkování.

Tento algoritmus podstatně zjednodušuje započítávání cyklů, protože na rozdíl od klasického způsobu nezapočítává jednotlivé půlcykly [5]), ale sdružuje sousední do úplných cyklů, i když s jistou malou chybou. Rozdíly však nejsou významné. Na konci zpracování zůstanou v zásobníku od každého signálu buď jen první vzorek, případně i poslední vzorek, o nichž stejně nelze prohlásit, že byly extrémy.

Algoritmus byl naprogramován v jazyku `MATLAB` a odladěn na PC Pentium/90 MHz, kde se dosahuje rychlostí zpracování (mezních vzorkovacích frekvencí f_s) uvedených v tabulce:

n_s	1	5	10	20
$f_c[\frac{vzork}{s}]$	317	442	599	620
$f_s[\frac{vektor}{s}]$	317	88	60	31

Z tabulky vyplývá, že jazyk `MATLAB`, v němž je algoritmus naprogramován do tvaru tzv. m-fumkce `mchrfl.m`, není vhodný pro zpracování rychlejších jevů, které je zapotřebí vzorkovat cca desetinásobkem nejvyšší frekvenční složky ve spektru, protože je pro skalární výpočty příliš pomalý. Pro rychlé děje by bylo zapotřebí funkci `mchrfl.m` naprogramovat v jazyku `C`, přeložit a vytvořit z ní tzv. `MEX`-file, totiž soubor `mchrfl.mex`, který je volatelný z `MATLAB`u jako každá jiná funkce, avšak jehož rychlost je řádově vyšší.

4 Závěr

Předložená verze vícekanálové dekompozice procesů metodou rain-flow může pracovat v reálném čase, a tedy není vázána na zpracování již v minulosti odvzorkované serie reálných napětí. Zpracovává vektory vzorků odebraných současně na množině procesů a z uzavřených cyklů počítá poškození. Protože v paměti uchovává pouze minimum informací, totiž jen extrémy, které dosud nevytvořily plný cykl, je paměťově nenáročná a umožňuje zpracovávat i značně dlouhé realizace provozních namáhání v libovolném počtu měřících míst současně. Nevýhodou této metody je stejně jako u jejího vzoru neschopnost analýzovat procesy odpovídající víceosé napjatosti.

```
%  ================================================================
    % RFLOW.M               Rain-flow damage cummulation
    % ~~~~~~~
    close all
    clear all
    global C_ file id scx

    format short e
    C_=cmtx;                          %  General coefficient matrix
    format short
    ns=inp('nsig',1);              %  number of signals
    ls=inp('lstk',50);            %  length of a stack
    N =inp('N',100);                %  number of samples per signal
    file=inp('file','proces.dat');       %  File name
    scx =inp(['Scale x'],[ones(ns,1)]); %  Meritko procesu x
    T   =inp('T',0.1);            %  Sampling period
    format short e

    [D,smax]=mchrfl(ns,N,ls)     %  Relative damage
    %        ~~~~~~~~~~~~~~~
    L=T*N/(D*24*3600)              %  Longevity in years
```

```
%================================================================
% MCHRFL.M            Multichannel Rain-Flow            v.3.0   Dec. 1995
% ~~~~~~~~
function [D,maxs] = mchrfl(ns,N,ls)
%         ~~~~~~~~~~~~~~~~~~~~~~~~
%  ns   = number of signals
%  N     = number of samples per signal
%  ls   = length of stack per signal
%  D     = vector of total damages
%  maxs = maximum used length of stack
%
%  ================================================================
%
D = zeros(ns,1);
S = zeros(ns,ls);   %            Stack matrix
S(:,1) = getvec(ns);
S(:,2) = getvec(ns);
tr = S(:,2)>S(:,1); %            trend vector
js = ones(ns,1)*2;  %            pointer vector
n  = 2;              %           data vector counter

while n<N             %     **** Cycle of sampled data vectors ****
  x = getvec(ns);   %                sample data
  n=n+1;

  for i=1:ns          %              Cycle of single samples in data vector
     j=js(i);
     if tr(i)         %                 Test trend
        if S(i,j)>x(i)   %              *** MAXIMUM ***
          tr(i) = ~tr(i); %             Maximum found
          while j>3      %              test for full cycle
             k=j-2;
             if S(i,j)<S(i,k), break, end
             D(i)=damage(S(i,k:k+1))+D(i); % Damage cummulation
             S(i,k)=S(i,j);
             j=k;            %              delete cummulated full cycle
          end
          j=j+1;
          js(i)=j;
        end % if maximum
     else
        if S(i,j)<x(i)   %              *** MINIMUM ***
          tr(i) = ~tr(i); %             Minimum found
          while j>3      %              test for full cycle
             k=j-2;
             if S(i,j)>S(i,k), break, end
             D(i)=damage(S(i,k:k+1))+D(i); % Damage cumulation
             S(i,k)=S(i,j);
             j=k;            %              delete cummulated full cycle
          end
          j=j+1;
          js(i)=j;
        end % if minimum
     end    % if tr(i)
     S(i,j)=x(i);    %                Put data to TOS
  end                   % i           Cycle of items in a data vector
end                    % n    **** Cycle of data vectors ****

for i=1:ns           %            Rest stack cummulation
  j=2; k=js(i);
  while j<k
     D(i)=damage(S(i,j:j+1))+D(i);
     j=j+2;
  end
end

maxs = max(sum(S'~=0)); %        Max. used length of stack
%================================================================
```

```
    % CMTX                          Matrix C_
    % ~~~~
    function C_ = cmtx
    %         ~~~~~~~~~
    w  = inp('exponent w',8);
    Nc = inp('N_c',1e7);
    sPt= inp('sigma_Pt',700);
    sc = inp('sigma_c',225);
    smk= inp('sigma_mk',180);       s=2*smk;
    scp= inp('sigma_ck+',180);
    scm= inp('sigma_ck-',230);
    C_=[ w,      0,              0
         Nc,       0,          -Nc/sPt^2
         sc, (scp-scm)/s, (scp+scm-2*sc)/s^2 ];
%================================================================
% GETVEC.M                  Get vector of data
% ~~~~~~~~
function d = getvec(n)
%         ~~~~~~~~~~~~~
global file id x scx

if isempty(id)
  eval(['load ' file]) %    load matrix of processes into "x":
  eval(['x=' eval('file(1:find(file==''.'')-1)') ';']);
  id=1;
end
d=x(id:id+n-1).*scx;
id=id+n;
%================================================================
% DAMAGE.M                   Full-cycle processing
% ~~~~~~~~
function D = damage(s)
%         ~~~~~~~~~~~~~
global C_

sm=(s(1)+s(2))/2;
p=C_*[1;sm;sm^2];           % p(1)=w_cm,  p(2)=N_cm,  p(3)=sigma_cm
D=(abs(s(1)-s(2))/p(3)*.5)^p(1)/p(2);
%================================================================
```

Poděkování: Příspěvek je jedním z výsledků řešení grantu GAČR 101/95/0087.

Reference

[1] Matsuishi M., Endo T.: Fatique of metals subjected to varying stress. Kyushu District meeting of Japan Society of Mechnical Engineers, Japan, March 1968

[2] Dowling N. E.: Fatigue failure predictions for complicated stress-strain histories. J. Mater. 7, (1972), č. 1, pp 71 – 87

[3] Klesnil M., Lukáš P.: Únava kovových materiálů při mechnickém namáhání. Academia, Praha, 1975

[4] Jelínek F.: Výpočty životnosti při proměnlivém kmitavém namáhání. IN: Únavová pevnost a životnost strojních částí. Sborník DT ČVTS, Praha, 1975

[5] Balda M.: Softwarové zabezpečení analýzy měření provozních zatížení. IN: Prevádzkové zaťaženie strujných a stavebných konštrukcií. Sborník ze semináře SVTS - DT Košice, 1976