Documentation for the fatigue dataset in composites

Manuel Chiachio Juan Chiachio Abhinav Saxena and Kai Goebel

E-mail: { mchiachio,jchiachio}@ugr.es

E-mail: {abhinav.saxena,kai.goebel}@nasa.gov

20 September 2013

(a) Placemente of actuators (1 to 6) and sensors (7 to 12).

(b) Specimen prepared to be tested.

Figure 1: SMART® Layers location and coupon specimen

1. Introduction

The data reported in this document correspond to the test of fatigue aging on CFRP composites carry out by the Stanford Structures and Composites Laboratory (SACL) in collaboration with the Prognostic Center of Excellence (PCoE) of NASA Ames Research Center. In this experiment, a set of coupons were subject to a tension-tension fatigue test conducted under a load controlled loop of cyclical loadings with a frequency of 5.0 [Hz] and a stress ratio of $R \simeq 0.14$. The coupons were laminated with ***three symmetric layup configurations*** to account for the effect of ply orientation: *a*) Layup 1: $[0_2/90_4]$; *b*) Layup 2: $[0/90_2/45/-45/90]$; and *c*) Layup 3: $[90_2/45/-45]_2$.

Torayca T700G unidirectional carbon-prepreg material was used for 15.24 [cm] $\times$ 25.4 [cm] coupons with dogbone geometry and a notch (5.08 [mm] $\times$ 19.3 [mm]) to induce stress concentration. The fatigue damage was monitored by two sets of six-PZT-sensors SMART Layer® from Acellent Technologies, Inc., which were attached to the surface of each sample. This configuration allows six actuators and six sensors to monitor Lamb wave propagation through the samples (see Figure 1). Each of the 36 actuator-sensor trajectories were actuated at 7 different interrogation frequencies in the range of $150 - 450$ [KHz], with an average input voltage of 50 volts and a gain of 20dB. Further details about the Lamb signal configuration are provided in Section 2.1.

All tests were performed on an MTS machine following ASTM Standards D3039 and D3479 [1, 2]. The fatigue cycling tests were stopped every $50,000$ cycles to collect PZT sensor data for all trajectories and interrogation frequencies. X-rays of the samples were taken using a dye-penetrant to enhance X-ray absorption. See further details in [3].

1.1. Data Acquisition

The system used to generate and collect Lamb waves information is the Scan Genie from Acellent Technologies, Inc. The Lamb wave is generated in one of the actuators, and is further received by one of the sensors, once the wave has travelled across the trajectory (path) joining actuator-sensor. Piezoelectric sensors (PZTs) have the capability of changing mechanical energy into electrical energy and vice versa. So an electrical voltage acting as an input in the i^{th} actuator, generates a mechanical excitation traveling across the specimen. When this mechanical excitation is sensed in the j^{th} sensor, it is changed into an electrical signal that can be recorded. The fatigue data for each coupon were collected considering ***three different boundary conditions:*** *a*) Type 1, specimen loaded with the mean load; *b*) Type 2, specimen unloaded but clamped; and *c*) Type 3, coupon removed from the testing machine (absolute 0 loads) .

In addition to this, and for the task of establishing a reference to identify the progression of damage in the coupon, the first step was to collect data when the specimen was undamaged and without the presence of loads. This gives the baseline that serves as reference to detect the damage propagation. Notice that each of the folders (corresponding to each of the coupons) contains one baseline data file.

2. Original data structure

This section describes the data structure as it is originally received from our Stanford partners. Notice that in Section 3.1, a new data structure is proposed and explained. However, previously to explain the new data structure, a detailed description of the former data format from Stanford seems convenient.

Table 1: Short extract of a path txt file

Path	Actuator	Sensor	Input Voltage	Frequency	Gain
1	1	7	50	150	30
2	1	8	65	150	30
3	1	9	65	150	30
4	1	10	60	150	25
5	1	11	65	150	27

Table 2: Short extract of the LogBook Excel® Sheet contained in each folder.

Date Responsible	Time	Record Information	Cycles	Load [Kips]	Boundary Conditions	Data File Name	MTS File Name	Remarks
24-may	13 : 40	3	1000000	0	Traction Free	L1_S17_NEW_baseline1.dat		–
24-may	02 : 28	2	1000000	0.003	Clamped	L1_S17_NEW_2_F000		–
24-may	05 : 30	1	1050000	3.505	Loaded	L1_S17_NEW_1_F002	L1_S17_NEW_1_F002	–

Table 3: Struct field names, types and description

Field name	Description	Field type
setup		struct var
a_0	Lamb signal (path 1)	matrix
a_1	Lamb signal (path 2)	matrix
$\vdots$	$\vdots$	$\vdots$
a_{251}	Lamb signal (path 252)	matrix
s_0	Lamb signal (path 1)	matrix
s_1	Lamb signal (path 2)	matrix
$\vdots$	$\vdots$	$\vdots$
s_{251}	Lamb signal (path 252)	matrix

Table 4: Setup struct field structure. See the structure of "signal_definition" field in Table 5.

Field name	Field type
sampling_rate	scalar
number_data_points	scalar
averaging_number	scalar
number_pretrigger	scalar
signal_definition	struct

The full set of fatigue data for each specific coupon is recorded in a specific folder that is called with the coupon name. Inside each folder, a LogBook Excel® Sheet can be found describing the data collection in a schematic fashion.
Table 2 shows a portion of a LogBook Excel® Sheet structure to serve as an example for better understanding the data collection.
For example, the file L1_S17_NEW_2_F000 in the second row of Table 2, contains the Lamb signals data corresponding to the cycle 1000000 when the specimen was unloaded but clamped (it appears an insignificant load of 0.003 [Kips] due to the zero error of the testing machine).
The information in each data file (for example in the file L1_S17_NEW_2_F000) is organized in a Matlab® struct array, with fields as shown in Table 3. The fields a_n and s_n, $n = \{0, \ldots, 251\}$, contains the Lamb signal information corresponding to the path $n + 1$. See Section 2.1 for a detailed definition of the concept of path for this work. Figure 2 shows an example of actuator and sensor Lamb signal for the path number 42. The "setup" field in Table 3 is again a nested struct variable containing the details related with the Lamb signal definition and acquisition. See Table 4 for a description of this field structure.

2.1. Path definition

The likelihood of detecting damage increases by choosing a set of trajectories sweeping the maximum material surface as possible. Based on this, all the actuators are connected to all the sensors defining a total number of 36 trajectories (see Figure 1). By other hand, due to the dispersive nature of Lamb waves (group and phase velocity vary with frequency), each trajectory is interrogated at 7 different frequencies (from 150 [Hz] to 450 [Hz]). Therefore, each data file corresponding to a specific fatigue cycle, contains a

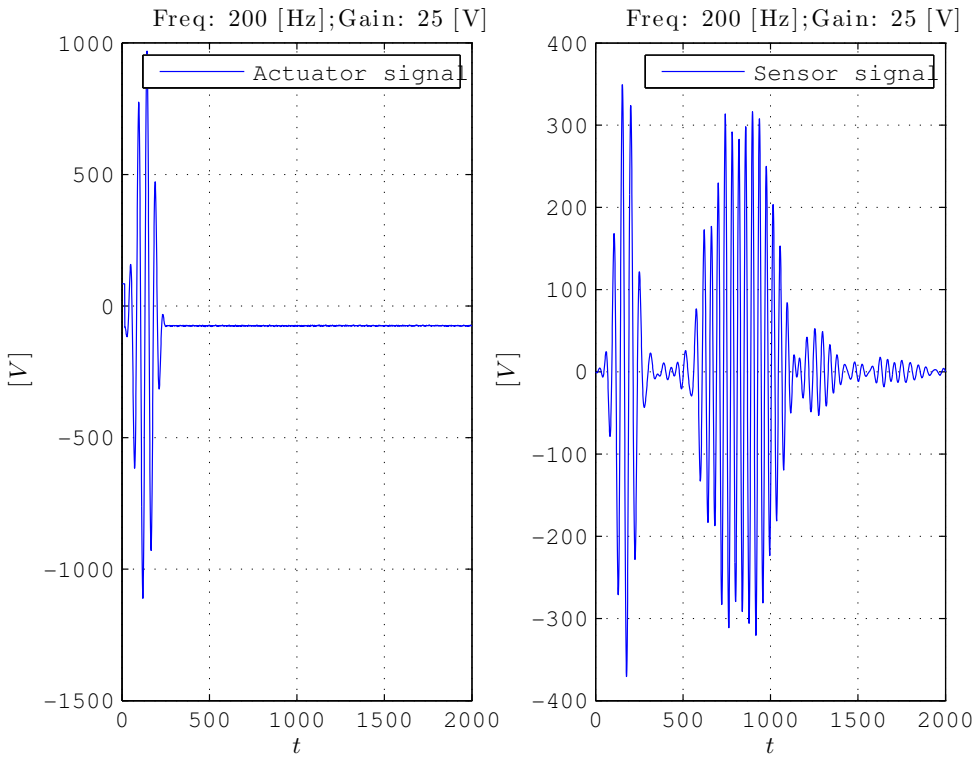

Figure 2: Plot of Lamb signals corresponding to the path 42 (matrices a_{41} and s_{41}): actuator 1, sensor 12, frequency 400 [Hz] and Gain 25 [V]. This data corresponds to the cycle $1.2 \cdot 10^6$ of a laminate with Layup type 1.

Table 5: Signal definition field structure. In bold text are highlighted the variables that define the path definition.

Field name	Field type
actuator	scalar
sensor	scalar
amplitude	scalar
signal_type	scalar
signal_type_text	string
frequency1	scalar
frequency2	scalar
act_nbr_points	scalar
act_sampling_freq	scalar
user_signal_filename	scalar
gain	scalar

total number of $36 \times 7 = 252$ signals for the actuators and another 252 signal for the sensors. We call path to any of these 252 signal data (see also Table 5). The path definition file includes the information about the path denomination (from 1 to 252), the pair actuator-sensor activated, input voltage, frequency and gain. See Table 1 for a short extract of 5 paths from a path txt file.

3. Matlab® Scripts

This section is dedicated to introduce the Matlab scripts created to run and modify (when it is necessary) the data for each of the coupons. Notice that each of this scripts are though to work inside the folder corresponding to each coupon.

3.1. Script to change the original structure to our desired final format

To upload all the test information to the PcoE webpage, a new data-structure was proposed in order to make the information more intelligible and intuitive for outsiders. See Figure 3 for a schematic description of the new data structure. For this task, we use a Matlab script called **NEWFILEDEF.m** that automatically converts and rename each of the files that are provided by our Stanford partners. These are instructions to run the Matlab file:

1. Open the Logbook and modify its structure until coinciding with one of the Logbooks that are already prepared. See the next paragraph where a basic guideline is provided.
2. Save the Excel sheet Logbook as a comma separated value file (csv), with the following filename: "couponname"import.csv. Then change the extension to *txt*. This is the final name of the input file that uses Matlab ‡.
3. Copy the file *Newfiledef.m* into the folder.
4. Open *Newfiledef.m* with Matlab, and change the next things:
   4.1 fid1=fopen('L1S11import.txt','r');
   4.2 Change couponname='L1S11';
   4.3 Uncomment only the STM matrix corresponding to the our corresponding coupon. If no stiffness info is provided, write a STM full of zeros (see the folders $L217$, $L2S18$, as example).
   4.4 Write the vector of Xray cycles. §

Through this, the Logbook is transformed into a *txt* that serves as a master-file to be read by Matlab. The following items are basic recommendations that help Matlab to correctly read the *txt* file:

- Open the Logbook and clean it of possible merged cells. Delete all rows that do not contain information.
- Delete the rows corresponding to the Strain and Xray info.
- Delete the columns "S. N^o", "Frequency" and "Responsible".
- Change the values of the cells in the "Record Information" column to appear only with 0 (when baseline), 1 (when loaded) 2 (when clamped) and 3 (when traction free). For the case of baseline data, the "Record Information" cell is assigned a 0 value.
- Fix any possible typos in the "Remarks" column.
- Only for those cases where the strain data for the three positions (S: sensors, M: middle, A: actuators) are provided, write the three strain files that were previously deleted in the item (2) (positions A, M, S) just in the row before them. Write it horizontally, without commas, only separated by a blank space.

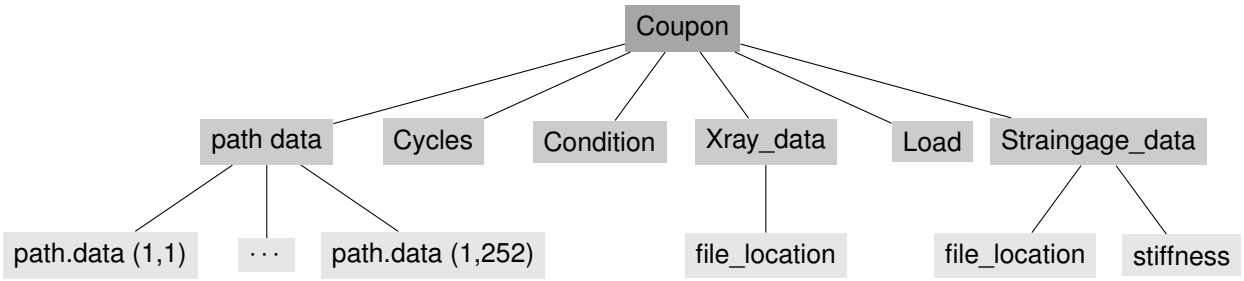

Figure 3: Schematical view of the new data structure. See Table 6, for more detailed information about the variable "path.data(1,j)", j: $\{1, \ldots, 252\}$.

‡ In case of using a single comma "," or a tab as separator, instead of ";" try to modify the delimiter specification on the $C\{1,i\}$ matrix definition: i.e., $C\{1,i\}$=textscan(fid1,'$\%s\%s\%s\%s\%s\%s\%s\%s\%s\%s$', 1, 'delimiter', ';') to $C\{1,i\}$ =textscan(fid1,'$\%s\%s\%s\%s\%s\%s\%s\%s\%s\%s$', 1, 'delimiter', '\t')

§ This vector contains the list of cycles where X Rays were taken for the coupon. The folder must contain the *jpg* files with the following naming convention: "couponname_cycle.jpg"

Table 6: Structure of the "path.data(i,j)" variable. Notice that Lamb waves signals are stored in "signal_sensor" and "signal_actuator" fields.

Field name	Field type
actuator	scalar
sensor	scalar
amplitude	scalar
sampling_rate	scalar
frequency	scalar
gain	scalar
signal_sensor	matrix
signal_actuator	matrix

3.2. To change the path definition

In some cases, the path convention for the Lamb wave data acquisition do not corresponds with what is reflected in 2.1. For these cases ($L1S11, L1S17, L2S17, L2S18, L2S20, L3S17,$ $L3S18, L3S19, L3S20$), a Matlab script called **CHANGEPATH.m** was created to automatically transform the internal path structure to the correct one. The procedure to run this Matlab file is analogous to the one used in the rest of Matlab files used here (i.e.,: *Newfiledef.m*), which has been previously explained in this document. *changepath.m* also needs a numerical matrix indicating the final format of the Lamb wave data structure. We called this matrix *path1mastermatrix.mat* and must be located in the same folder where *changepath.m* runs. This matrix can be taken from the folders corresponding to the coupons cited on the last parenthesis.

3.3. Data management

In addition to the LogBook Excel® Sheet that gives the information to select the desired file name between all those contained in the same folder, several Matlab ® codes are provided to help the user to navigate around the data set. The following list provides a description of the Matlab codes provided for data management:

DATA_MANAGEMENT.m This is a Matlab user interface to operate with the set of fatigue data of a coupon. This code allows the user to import and/or plot the data corresponding to: a) a specific cycle number for a specific path or b) for specific sensor-actuator pair. This code also gives information about the folder contents and also information referred to a specific number of cycles.

PIECE1.m This is a piece of code to serve as example for users that need to import the Lamb wave signals corresponding to a specific pair actuator-sensor, for a given frequency and load state (boundary condition).

PIECE2.m This piece of code is an example to import the signal data corresponding to each cycle of the experiment. The user might indicate the actuator-sensor pair, frequency and boundary condition; and the script will return a matrix containing the signals for each cycle value.

3.4. General instructions to run the Matlab scripts

All the scripts cited above are based on a data structure that is taken from a master txt file. This txt file is specific for each folder (and so specific for each coupon). The only rule to run these scripts is that the txt file, the Matlab script and the data files must be located in the same folder (the current directory of Matlab). For example, suppose the case that we are going to run the DATA_MANAGEMENT.m script for the L2S17 coupon (L2_S17_F folder). We have to see inside this folder the following files:

L2_S17_F:
- L2S17import.txt
- DATA_MANAGEMENT.m
- all the data files: "L2_S17_1_F002.dat", "L2_S17_1_F003", etc.

There maybe cases in which more that the last general requirements are needed. For example, to run the *changepath.m*, the *path1mastermatrix.mat* matrix is also required. Those details are explicitly highlighted in each section.

Appendix A. Matlab codes

This appendix provides the Matlab scripts developed for the work presented herein. A detailed description of each code is provided below, together with some comments that may help to understand them better.

Appendix A.1. NEWDEFFILE.m

```
%This script is to change the data structure for the NASA-Stanford dataset
%of composites
%Manuel & Juan Chiachio & Abhinav Saxena (NASA ARC, Nov 2012)


clear all

fid1=fopen('L1S11import.txt','r');% Opens the data file
heading = textscan(fid1,'%s',1,'delimiter','\n');% It discards the heading
couponname='L1S11';
i=1; % counter initialization
% It reads the first row of the txt file
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
FILENAMES=[];
FILENAMES=[FILENAMES;C{1,i}{1,7}];
%Successively reads the rows until it find a row without a Data File
%(Column 7)
while isempty(C{1,i}{1,7})== false
i=i+1;
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
FILENAMES=[FILENAMES;C{1,i}{1,7}];
end
%
fclose(fid1);
%%%%%%Stiffness information %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%T19
% STM=[0,1,10,100,1000,10000,20000,30000,40000,50000,75000,100000;...
%      0,0.044201246,0.047862216,0.045450695,0.050135543,0.055153717,...
%      0.070914577,0.078797313,0.080422812,0.087165429,0.092774474,0.093123452];
%%%T21
%  STM=[0,1,10,100,1000,10000,20000,30000,40000,50000,75000,100000;...
%       0,0.033365108,0.038699259,0.039564571,0.044786464,0.060346674,...
%       0.091761864,0.098345182,0.105576807,0.110909529,0.114961755,0.119143486];
%%%T22
%  STM=[0,1,10,100,1000,10000,20000,30000,40000,50000,60000,70000,80000,...
%       90000,100000,0,0.021536202,0.028654042,0.030159738,0.031429189,...
%       0.069976451,0.084637842,0.088982321,0.094290387,0.091580501,...
%       0.11,0.110606883,0.11799544,0.247092307,0.250330479];
%%%L1S11
STM=[0,1,10,100,1000,10000,20000,30000,40000,50000,60000,70000,80000,...
    90000,100000;0,0.055036381,0.045813245,0.060735707,0.069766198,...
    0.076005931,0.098380137,0.10051699,0.11226211,0.118553783,...
    0.10426744,0.128221136,0.12236086,0.11445585,0.120155531];
%%%L1S12
%  STM=[0,1,10,100,1000,10000,30000,40000,50000,60000,70000,80000,90000,...
%       100000;0,0.043288345,0.051679586,0.047985147,0.059726484,0.084112497,...
%       0.233175942,0.239071597,0.236134709,0.245338988,0.244799668,...
%       0.237792304,0.245462847,0.245730332];
%%%L2S11
% STM=[0,1,10,100,1000,10000,20000,30000,40000,50000,60000;...
%      0,-0.012213572,0.012154654,0.005044283,0.008583815,...
%      0.010634185,0.031656763,0.042811126,0.055037786,0.044292129,0.084180661];
%%% L2S12
%  STM=[0,1,10,100,1000,10000,20000,30000;...
%       0,-0.015854422,0.003176221,-0.001135796,0.011828933,...
%       0.033564358,0.045858964,0.053385839];

%%%%%%XRay information %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
XRM=[0,1,10,100,1000,10000,20000,30000,40000,50000,60000,70000,80000,90000,100000];

%Heading details and global information
NEWFILENAME={};
%%
matrixreptcont=[];
Checkmatrix2={};
% File to load
```

```
k=0;

fid3=fopen('filestable.txt','wt');
fprintf(fid3, 'Filename;  Cycle;  Load;  Bound.Cond  \n\n');

for j=1:numel(FILENAMES)

%%%this is to call the actual data from the "dat" file
kk=importdata(strcat(FILENAMES{j},'.dat'));
signal_definition=kk.setup.signal_definition;


for i=1:size(signal_definition,2)

    %This gets and orders the information from the actual "dat" file
    act=signal_definition(i).actuator;% Actuator
    sen=signal_definition(i).sensor;  % Sensor
    amp=signal_definition(i).amplitude; % Input Voltage
    %sig_type=signal_definition(i).signal_type; % Signal type
    freq=signal_definition(i).frequency1;% Frequency
    gain=signal_definition(i).gain;% Gain

    lambsignalact=(getfield(kk,strcat('a',num2str(i-1))));
    lambsignalsens=(getfield(kk,strcat('s',num2str(i-1))));

    %%%Here I select some info from the "dat" file and plug into my struct

    path_data(i).actuator=act;
    path_data(i).sensor=sen;
    path_data(i).amplitude=amp;
    path_data(i).frequency=freq;
    path_data(i).gain=gain;
    path_data(i).signal_actuator=lambsignalact;
    path_data(i).signal_sensor=lambsignalsens;
    path_data(i).sampling_rate=1.2e6;
    clear act sen amp freq gain

end

%%%%Stiffness degradation
ind2=find(STM(1,:)==str2num(C{1,j}{1,4}{1,1}));%Finds in the stiff. matrix.
ind3=find(XRM==str2num(C{1,j}{1,4}{1,1}));     %Finds in the XRay matrix.
s=pwd;%%%This is to read the current directory
switch isempty(num2str(C{1,j}{1,8}{1,1}))
case true
    straingage_data.file_location=[];
case false
    B=textscan(C{1,j}{1,8}{1,1}, '%s %s %s', 1); %Find the filename

for m=1:numel(B)
    %This is to avoid the case in which we only have one file for
    %strain position.
    if isempty(B{1,m})==true continue
    end
    straingage_data.file_location{1,m}{1,1}=strcat(B{1,m}{1,1},'_DAT');


%%%%This is to select the strain data files cited in the Logbook and put
%%%%all together in a new folder to be uploaded. StrainData is the
%%%%folder with the selected strain files.
   copyfile(strcat('rawstrain/',straingage_data.file_location{1,m}{1,1})...
   ,strcat(s,'/StrainData'));
    end

end
straingage_data.stiffness_degradation=STM(2,ind2);
if isempty(ind3)==true
    XRay_data.file_location=[];
else
XRay_data.file_location=strcat('\XRay\',couponname,'_',num2str(XRM(ind3)),'.jpg');

end

%%%% Now here, the new coupon struc for the current filename
coupon=struct('cycles', str2num(C{1,j}{1,4}{1,1}), 'load',...
    str2num(C{1,j}{1,5}{1,1}),'comment',C{1,j}{1,9}{1,1},'condition',...
    C{1,j}{1,6}{1,1},'path_data',path_data,'straingage_data',...
```

```
    straingage_data,'XRay_data',XRay_data);

matrixreptcont=[matrixreptcont;coupon.cycles,str2num(C{1,j}{1,3}{1,1})];
%This is to control the cycles and bound. cond. repetitions in the serie.

%%%This is to rename the file with the new convention
if j>1
    %Finds repetitions
    ind1=find(matrixreptcont(1:end-1,1)==matrixreptcont(end,1));
    ind2=find(matrixreptcont(1:end-1,2)==matrixreptcont(end,2));


    if (isempty(ind2)==false && isempty(ind1)==false)

switch str2num(C{1,j}{1,3}{1,1})

case 0
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(0)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(0),'.mat');
case 1
  k1=min(numel(ind2),numel(ind1))+1;
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(1),'_',num2str(k1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(1),'_',num2str(k1),'.mat');
case 2
  k2=min(numel(ind2),numel(ind1))+1;
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(2),'_',num2str(k2)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(2),'_',num2str(k2),'.mat');
case 3
  k3=min(numel(ind2),numel(ind1))+1;
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(3),'_',num2str(k3)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(3),'_',num2str(k3),'.mat');

end %endswitch

elseif isempty(ind1)==false %else of no change of cycle

switch str2num(C{1,j}{1,3}{1,1})
case 0
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(0)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(0),'.mat');
case 1
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(1),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(1),'_',num2str(1),'.mat');
case 2
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(2),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(2),'_',num2str(1),'.mat');
case 3
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(3),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
        '_',num2str(3),'_',num2str(1),'.mat');

        end %endswitch

    else %else of no change of cycle and no repetitions

matrixreptcont=[];%To seek only within a cycle set
matrixreptcont=[matrixreptcont;coupon.cycles,str2num(C{1,j}{1,3}{1,1})];
switch str2num(C{1,j}{1,3}{1,1})
case 0
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(0)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(0),'.mat');
case 1
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(1),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(1),'_',num2str(1),'.mat');
case 2
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(2),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(2),'_',num2str(1),'.mat');
```

```
case 3
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(3),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(3),'_',num2str(1),'.mat');

end %endswitch

    end %Endif No change of cycle
else %this is just for the first file

switch str2num(C{1,j}{1,3}{1,1})
case 0
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(0)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(0),'.mat');
case 1
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(1),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(1),'_',num2str(1),'.mat');
case 2
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(2),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(2),'_',num2str(1),'.mat');
case 3
  save([couponname,'_',C{1,j}{1,4}{1,1},'_',num2str(3),'_',num2str(1)],'coupon');
  NEWFILENAME{j,1}= strcat(couponname,'_',C{1,j}{1,4}{1,1},...
      '_',num2str(3),'_',num2str(1),'.mat');

   end %endswitch

end%endif


fprintf(fid3, ' %s.mat; %8i; %4.5f; %s \n\n',NEWFILENAME{j,1},...
    coupon.cycles,coupon.load,coupon.condition);

end


fprintf(fid3, '\n\n');
fclose(fid3);
```

Appendix A.2. CHANGEPATH.m

```
%This script is to change the path denomination is the NASA-Stanford data
%Manuel & Juan Chiachio & Abhinav Saxena (NASA ARC, Nov 2012)


clear all

fid1=fopen('L2S20import.txt','r');% Opens the data file
heading = textscan(fid1,'%s',1,'delimiter','\n');% It discards the heading

i=1; % counter initialization
% It reads the first row of the txt file
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
FILENAMES=[];
%Successively reads the rows until it find a row without a Data File
%(Column 7)
while isempty(C{1,i}{1,7})== false
    i=i+1;
    C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
    FILENAMES=[FILENAMES;C{1,i}{1,7}];
end
%
fclose(fid1);

%Heading details and global information

Checkmatrix1={};
% File to load
%%%This a matrix with the path information corresponding to the path 1 name
load path1mastermatrix %It offers a matrix called P


for j=1:numel(FILENAMES)
    kk=importdata(strcat(FILENAMES{j},'.dat'));

    signal_definition=kk.setup.signal_definition;
    P2=[];

    % Name the output file and write it into the folder Paths
    signal_definition=kk.setup.signal_definition;

    for i=1:size(signal_definition,2)

act=signal_definition(i).actuator;% Actuator
sen=signal_definition(i).sensor;  % Sensor
amp=signal_definition(i).amplitude; % Input Voltage
%sig_type=signal_definition(i).signal_type; % Signal type
freq=signal_definition(i).frequency1;% Frequency
gain=signal_definition(i).gain;% Gain

lambsignalact=(getfield(kk,strcat('a',num2str(i-1))));
%This copies the signal to an auxiliary var. to ensure that we do not
%replace any file
kk=setfield(kk,strcat('auxi',num2str(i-1)),lambsignalact);

lambsignalsens=(getfield(kk,strcat('s',num2str(i-1))));
kk=setfield(kk,strcat('s_auxi',num2str(i-1)),lambsignalsens);


[r]=find (P(:,2)==act & P(:,3)==sen & P(:,5)==freq,1,'first');%Search
lambsignalact2=(getfield(kk,strcat('auxi',num2str(i-1))));%Changing
kk=setfield(kk,strcat('ac',num2str(r-1)),lambsignalact2);%Changing

lambsignalsen2=(getfield(kk,strcat('s_auxi',num2str(i-1))));%Changing
kk=setfield(kk,strcat('sc',num2str(r-1)),lambsignalsen2);%Changing
%i-1 because of the way in which the "ai" matrices are called

%Managing with the signal_definition matrix. In this step, the signal
%definition matrix is also modified in correspondence to the path
%definition type (1)
%This is to get the signal_definition information from Path type 2
SDMAN=getfield(kk,'setup','signal_definition');

%%%Now the the structure of the signal_definition is changed (the order) to
```

```
%%%fullfil the Path name convention (1)
kk=setfield(kk,'setup','signal_definition_mod',{1,r},SDMAN(1,i));

    end


    %%%*************************************************************************
    %%%This is to rename the "ac" and "sc" matrices to their initial names
for i=1:size(signal_definition,2)
lambsignalact3=(getfield(kk,strcat('ac',num2str(i-1))));
kk=setfield(kk,strcat('a',num2str(i-1)),lambsignalact3);

lambsignalsens3=(getfield(kk,strcat('sc',num2str(i-1))));
kk=setfield(kk,strcat('s',num2str(i-1)),lambsignalsens3);
end



    %%%*************************************************************************
    %%%This is to write the new path txt file (This also serves to check)

    % Name the output file and write it into the folder Paths
    %Creates and opens an "txt" object into the current folder
fid3=fopen(strcat('Paths/',FILENAMES{j},'shortFIXEDpath.txt'),'wt');
fprintf(fid3, 'Path  Actuator  Sensor  InputVoltage    Frequency    Gain\n\n');
signal_definition_mod=kk.setup.signal_definition_mod;

    for i=1:size(signal_definition_mod,2)

act=kk.setup.signal_definition_mod(i).actuator;% Actuator
sen=kk.setup.signal_definition_mod(i).sensor;  % Sensor
amp=kk.setup.signal_definition_mod(i).amplitude; % Input Voltage
%sig_type=signal_definition_mod(i).signal_type; % Signal type
freq=kk.setup.signal_definition_mod(1,i).frequency1;% Frequency
gain=kk.setup.signal_definition_mod(i).gain;% Gain
fprintf(fid3, ' %d   %d   %d   %d   %d   %d \n\n',i, act,sen,amp,freq,gain);
clear act sen amp freq gain
    end

    fprintf(fid3, '\n\n');
    fclose(fid3);

    %%%Now the variable signal_definition_mod is renamed to its initial name
    %(mandatory for the rest of codes which takes this matrix information)
    %This is to get the signal_definition information from Path type 2
    signal_definition=getfield(kk,'setup','signal_definition_mod');


    clear kk % It clears the kk struct variable
end
```

Appendix A.3. DATA_MANAGEMENT.m

```
%Data Management script
%M & J Chiachio & Abhinav Saxena, October 2012
%NASA, ARC
clear all

fid1=fopen('L1S11import.txt','r');% Opens the data file
heading = textscan(fid1,'%s',1,'delimiter','\n');% It discards the heading

i=1; % counter initialization. It reads the first row of the txt file
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');

    i=i+1;
%Successively reads the rows until it find a row without a Data File (Column 7)
while isempty(C{1,i}{1,7})== false
    C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
end
%
fclose(fid1);

%Heading details and global information
Heading1='NASA-STANFORD FATIGUE TEST IN CFRP. \n';
Heading2=' This is the information for the fatigue test of the L1S17 coupon';
reportname='L1S17_test_information.doc';
frecuency=5;
maxload=6;
minload=2;
Date1=C{1,1}{1,1}{1,1};
Date2=C{1,end-1}{1,1}{1,1};
cycleini=C{1,1}{1,4}{1,1}; %This is a string. See the reading mode below.
cyclefin=C{1,end-1}{1,4}{1,1};%Idem.

%%%%%Display information in the Command Window
mainuserexit=1; %Enables to operates this case until the user decides to exit
while mainuserexit==1



clc, format short e

disp('  ____________________________________________________________________ ');
disp(' |    Welcome to database of fatigue in CFRP from NASA-STANFORD       |');
disp(' | More information, visit: http://ti.arc.nasa.gov/tech/dash/pcoe//   |');
disp(' | Note: All the information below assumes that the user is           |');
disp(' |       familiar with the typical terms of a fatigue test.           |');
disp('  ____________________________________________________________________');
disp(' ');
disp('   0: Exit');
disp('   1: Help');
disp('   2: Import the database for a given number of cycles');
disp('   3: Obtain information for a specific cycle');

disp(' ');


informtype = input(' CHOOSE OPTION FROM THE LIST ABOVE: ');


while (isnumeric(informtype)==false || informtype < 0 || informtype > 3 )
disp(' ');
disp(['Sorry, the available options are from ',num2str(0),' to ',num2str(3)]);
informtype = input('Please try again with a allowed numeric value:');
end

switch informtype

case 0 % ---- EXIT ----------------------------------------------------
mainuserexit=0;
disp(' ');
disp('  Bye, bye.');
disp(' ');


```

```
74
75  case 1 % ---- HELP -----------------------------------------------------
76
77  mainuserexit=1;% It refresh the user exit variable.
78  userexit=1; %This a variable that enables to operates this
79  %case until the user decides to exit
80
81  clc
82  disp(' ');
83  disp('  HELP');
84  disp(' ');
85  disp('If you are new to this database, you are recommended to visit the web page:');
86  disp('http://ti.arc.nasa.gov/tech/dash/pcoe//');
87  disp(' ');
88  disp(['Check your current directory for a file called ',reportname]);
89
90  disp(' ');
91
92  This is the script that allows to start again.
93  mainuserexit=input('Pulse 1 to go the main menu. Pulse 0 to exit: ');
94
95
96  while (isnumeric(mainuserexit)==false || mainuserexit>1 )
97  mainuserexit=input('Please introduce again a correct option: ( "0,1") : ');
98  end
99
100 if mainuserexit==0;
101
102
103     disp(' ');
104     disp('  Bye, bye.');
105     disp(' ');
106
107 end
108
109 case 2 % ---- IMPORT DATA -------------------------------------------------
110 userexit=1; %This variable enables the user to decide to exit
111 while userexit==1
112 %%%Previous checking
113
114 cycles=[];
115 for i=1:max(size(C))-1
116
117 cycles=[cycles;str2num(C{1,i}{1,4}{1,1})];
118 end
119 %%%%%%%%%%%%%%%%
120 clc
121 disp('  This allows you to select and import the data for a specific cycle');
122 disp('   ');
123 inpcycle = input(' INTRODUCE THE CYCLE TO IMPORT THE DATA: ');
124 %%%Check the user input to be according with our available data.
125 while (isnumeric(inpcycle)==false||inpcycle<cycles(1)||inpcycle>cycles(end) )
126 disp(' ');
127 disp (['Sorry, we only have data for the range of cycles ',...
128     num2str(cycles(1)),' to ',num2str(cycles(end))]);
129 inpcycle = input('Please try again with a new value between the cited range:');
130 end
131
132 indsrch=find(cycles==inpcycle, 1, 'first');
133 lastposit= find(cycles>inpcycle, 1, 'first');
134 initposit= find(cycles<inpcycle, 1, 'last');
135
136
137
138 while isempty(indsrch)==true;
139 disp(' ');
140 disp('_____________________________________________________________');
141 disp('Warning:');
142 disp(['This database does not contain information for the cycle: ',...
143     num2str(inpcycle)]);
144 disp(['The best match for your entry corresponds to the cycle: ',...
145     num2str(cycles(initposit))]);
146 disp([' as lower bound and ',num2str(cycles(lastposit)),' as upper bound']);
147 optn=input('Please, pulse 1 to consider this lower bound or 2 for the upper bound.');
148 disp('Pulse 0 to try again with other cycle: ');
149 if optn==1;
150 inpcycle= cycles(initposit);
```

```
indsrch=find(cycles==inpcycle, 1, 'first');% allows to scape from the while loop
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');

elseif optn==2;
inpcycle= cycles(lastposit);
indsrch=find(cycles==inpcycle, 1, 'first');% allows to scape from the while loop
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');
else optn==0;
inpcycle = input('Try again with a new cycle value : ');

%Check
while (isnumeric(inpcycle)==false|| inpcycle < cycles(1) || inpcycle > cycles(end) )
disp(' ');
disp('_____________________________________________________________');
disp('Warning:');
disp (['Sorry, we only have data in the range',num2str(cycles(1)),...
    ' to ',num2str(cycles(end))]);
inpcycle = input('Try again with a new value between the available range of data: ');
end
%Compute again the data bounds and the availability
indsrch=find(cycles==inpcycle, 1, 'first');
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');
end

end

FILENAMES={};

if isempty(initposit)==true %The init. position is 1
    initposit=0;
    for i=1:(lastposit-initposit-1)
        FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};

    end
elseif isempty(lastposit)==true %The last position is the final

    lastposit=max(size(C));
    for i=1:(lastposit-initposit-1)
        FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
    end

else
    for i=1:(lastposit-initposit-1)
        FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
    end
end

disp(['These ',num2str(lastposit-initposit-1),...
    ' data files are found containing information:']);
disp(FILENAMES);


optn2=input('Pulse: 1 to import all; 2 to import a selected file; 3 to start again: ');
while (isnumeric(optn2)==false || optn2>3 )
optn2=input('Introduce again a correct option between the allowed list: "1,2,3" :');
end

if optn2==1;

    for j=1:numel(FILENAMES)
        kk(j)=importdata(strcat(FILENAMES{j},'.dat'));
    end

    disp('****DATA FILE SUCCESSFULLY LOADED****');
    disp('----Check your Matlab workspace for the struct variable kk----');

elseif optn2==2;
disp('');
disp('Note: the files are numbered from top to down.');
disp('');
posit=input('Please, select the position of the file from the previous list:');
while (isnumeric(posit)==false || posit ≤0 || posit > numel(FILENAMES)  )
disp(['There are ',num2str(numel(FILENAMES)),...
    ' files numbered from 1 (top) to ',num2str(numel(FILENAMES)), ' (down).']);
```

```
posit=input('Please introduce again a correct position input');
input(['an integer number between 1 to ',num2str(numel(FILENAMES)) '):']);
end


kk=importdata(strcat(FILENAMES{posit},'.dat'));

disp('');
disp('');
disp('Now you can obtain the information for this/these file/s');

optn3= input('Pulse 1 to obtain information from a specific path');
input('Pulse 2 if you know the sensor config. Pulse 3 to exit:');

while (isnumeric(optn3)==false || optn3>3 )
optn3=input('Please introduce again a correct option');
input('between the allowed list ( "1,2,3") : ');
end

if optn3==1
pathinp=input('Please, introduce the path number:');
while (pathinp < 0 || pathinp > 252 || isnumeric(optn3)==false)
    pathinp=input('Paths are named from 1 to 252. Please try again:');
end

signal_definition=kk.setup.signal_definition;

act=signal_definition(1,pathinp).actuator;% Actuator
sen=signal_definition(1,pathinp).sensor;  % Sensor
% amp=signal_definition(i).amplitude; % Input Voltage
% %sig_type=signal_definition(i).signal_type; % Signal type
freq=signal_definition(1,pathinp).frequency1;% Frequency
gain=signal_definition(1,pathinp).gain;% Gain
disp('-------------------------------------------------------------------------')
disp(['Actuator: ',num2str(act),' Sensor: ',num2str(sen),...
    ' Frequency[Hz]: ',num2str(freq),' Gain[V]: ',num2str(gain)]);
disp('');
lambsignalact=(getfield(kk,strcat('a',num2str(pathinp))));
lambsignalsens=(getfield(kk,strcat('s',num2str(pathinp))));

%%%%%%PLOTTING TOOL
subplot(1,2,1), plot(lambsignalact);
set(gcf,'PaperUnits','centimeters','PaperPosition',[0.5 0.5 5 5])

%%%%%%%Axis properties%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ax1=gca;
set(Ax1,'Color','w',...
    'YGrid','on','XGrid','on');

set(gca,'FontName','Latex');
xlabel('$t$','Interpreter','latex'); ylabel('$[V]$','Interpreter','latex');
title(['Freq: ',num2str(freq),' [Hz];','Gain: ',num2str(gain),' [V]']);
h=legend('Actuator signal',1);

subplot(1,2,2), plot(lambsignalsens);



set(gcf,'PaperUnits','centimeters','PaperPosition',[0.5 0.5 5 5])

%%%%%%%Axis properties%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ax1=gca;
set(Ax1,'Color','w',...
    'YGrid','on','XGrid','on');

set(gca,'FontName','Latex');
xlabel('$t$','Interpreter','latex'); ylabel('$[V]$','Interpreter','latex');
title(['Freq: ',num2str(freq),' [Hz];','Gain: ',num2str(gain),' [V]']);
h=legend('Sensor signal',1);



disp('');
disp('-------------------------------------------------------------------------');
disp('Check your Matlab Workspace for a matrix called:');
disp('"lambsignalact" for the actuator signal');
disp('and "lambsignalsens" for the sensor signal');
```

```
disp('----------------------------------------------------------------------');

elseif optn3==2
signal_definition=kk.setup.signal_definition;
actuat=input('Please introduce the actuator number:');
while (actuat < 1 || actuat > 6 || isnumeric(actuat)==false)
actuat=input('Actuators are called with numbers from 1 to 6. Please try again:');
end
disp('');
sensnam=input('Now introduce the sensor number:');
while (sensnam < 7 || sensnam > 12 || isnumeric(sensnam)==false)
sensnam=input('Sensors are called with numbers from 7 to 12. Please try again:');
end

%freq=input('Choose a frequency. Pulse enter if you want to bypass the frequency');
selected_cells=[];
for i=1:size(signal_definition,2)
    %
if (signal_definition(1,i).actuator==actuat...
        && signal_definition(1,i).sensor==sensnam);
    freq=signal_definition(1,i).frequency1;% It takes the frequency
    gain=signal_definition(1,i).gain;%It takes the gain
    selected_cells=[selected_cells;i-1,freq,gain];
    %i-1 because in the data, the numbering is done in this way.
end

end
signalactmatrix=[];
signalsensmatrix=[];

for j=1:max(size((selected_cells(:,1))))

    lambsignalact=(getfield(kk,strcat('a',num2str(selected_cells(j,1)))));
    lambsignalsens=(getfield(kk,strcat('s',num2str(selected_cells(j,1)))));


    subplot(7,2,(2*j-1)), plot(lambsignalact);
    %%%%%PLOT SIZE FOR ADJUSTING TO PAPER SIZING REQUIREMENT
    set(gcf,'PaperUnits','centimeters','PaperPosition',[0.5 0.5 14 14])

    %%%%%%%AXIS PROPERTIES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Ax1=gca;
    set(Ax1,'Color','w',...
        'YGrid','on','XGrid','on');

    set(gca,'FontName','Latex');
    xlabel('$t$','Interpreter','latex'); ylabel('$[V]$','Interpreter','latex');
    title(['Freq: ',num2str(selected_cells(j,2)),' [Hz];',' Gain: ',...
        num2str(selected_cells(j,3)), ' [V]']);
    h=legend('Actuator signal',1);
    %hgsave('.fig');
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    subplot(7,2,2*j), plot(lambsignalsens);
    %%%%%PLOT SIZE FOR ADJUSTING TO PAPER SIZING REQUIREMENT
    set(gcf,'PaperUnits','centimeters','PaperPosition',[0.5 0.5 14 14])

    %%%%%%%AXIS PROPERTIES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Ax1=gca;
    set(Ax1,'Color','w',...
        'YGrid','on','XGrid','on');

    set(gca,'FontName','Latex');
    xlabel('$t$','Interpreter','latex'); ylabel('$[V]$','Interpreter','latex');
    title(['Freq: ',num2str(selected_cells(j,2)),' [Hz];',' Gain: ',...
        num2str(selected_cells(j,3)), ' [V]']);
    h=legend('Sensor signal',1);
    %hgsave('.fig');

    signalactmatrix=[signalactmatrix,lambsignalact];
    signalsensmatrix=[signalsensmatrix,lambsignalsens];


end
disp('');
disp('----------------------------------------------------------------------');
disp('Check your Matlab Workspace for a matrix called: "signalactmatrix"');
disp('for the actuator signals and "signalsensmatrix" for the sensor signals.');
```

```
disp('(Each column corresponds to a different frequency. See also the figure.');
disp('-------------------------------------------------------------------');


elseif optn3==3
userexit=1;
end

elseif optn2==3;
    userexit=1;
end

%%%This is to return to the main menu or to exit

userexit=input('Pulse 1 to obtain information for a different cycle.');
disp('Pulse 2 to go to the main menu. Pulse 0 to exit: ');
while (isnumeric(userexit)==false || userexit>2 )
userexit=input('Introduce again a correct option between the list ( "0,1,2") : ');
end

end

if userexit==0;
mainuserexit=0;% It refresh the user exit variable.

disp(' ');
disp('  Bye, bye.');
disp(' ');

elseif userexit==2
mainuserexit=1;

end
case 3 % ---- OBTAIN INFORMATION ----------------------------------------
%This a variable that enables to operates this case until the user decides
%to exit
userexit=1;
while userexit==1
cycles=[];
for i=1:max(size(C))-1
cycles=[cycles;str2num(C{1,i}{1,4}{1,1})];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
disp('This option allows you to select and import the data for a given cycle');
disp('   ');
inpcycle = input(' INTRODUCE THE CYCLE TO OBTAIN THE INFORMATION: ');
%%%Check the user input to be according with our available data.
while (inpcycle < cycles(1) || inpcycle > cycles(end)|| isnumeric(inpcycle)==false)
disp (['Sorry, we only have data for the range of cycles ',...
    num2str(cycles(1)),' to ',num2str(cycles(end))]);
inpcycle = input('Please try again with a new value between the cited range:');
end

indsrch=find(cycles==inpcycle, 1, 'first');
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');



while isempty(indsrch)==true;
disp(' ');
disp('_____________________________________________________________');
disp('Warning:');
disp(['This database does not contain information for the cycle: ',...
    num2str(inpcycle)]);
disp(['The best match for your entry corresponds to the cycle: ',...
    num2str(cycles(initposit))]);
disp([' as lower bound and ',num2str(cycles(lastposit)),' as upper bound']);
disp(' ');
optn=input('Please, pulse 1 to consider this lower bound or 2 for the upper bound.');
dis('Pulse 0 to try again with other cycle: ');
if optn==1;
    inpcycle= cycles(initposit);
    indsrch=find(cycles==inpcycle, 1, 'first');% scape from the while loop
    lastposit= find(cycles>inpcycle, 1, 'first');
    initposit= find(cycles<inpcycle, 1, 'last');
```

```
459
460 elseif optn==2;
461     inpcycle= cycles(lastposit);
462     indsrch=find(cycles==inpcycle, 1, 'first');% scape from the while loop
463     lastposit= find(cycles>inpcycle, 1, 'first');
464     initposit= find(cycles<inpcycle, 1, 'last');
465 else optn==0;
466 inpcycle = input('Try again with a new cycle value : ');
467
468 %Check
469 while inpcycle < cycles(1) || inpcycle > cycles(end)
470 disp(' ');
471 disp('_____________________________________________________________');
472 disp('Warning:');
473 disp (['Sorry, we only have data in the range ',...
474 num2str(cycles(1)),' to ',num2str(cycles(end))]);
475 inpcycle=input('Please try again with a new value between the available range: ');
476     end
477     %Compute again the data bounds and the availability
478     indsrch=find(cycles==inpcycle, 1, 'first');
479     lastposit= find(cycles>inpcycle, 1, 'first');
480     initposit= find(cycles<inpcycle, 1, 'last');
481
482     end
483
484
485
486 end
487
488 %Matrix containing the file names
489 FILENAMES={};
490 Recordinf=[];
491 cycleinf=[];
492 Loadinf=[];
493 if isempty(initposit)==true %The init. position is 1
494     initposit=0;
495     for i=1:(lastposit-initposit-1)
496         FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
497         Recordinf=[Recordinf;str2num(C{1,initposit+i}{1,3}{1,1})];
498         Loadinf=[Loadinf;str2num(C{1,initposit+i}{1,5}{1,1})];
499         cycleinf=[cycleinf;str2num(C{1,initposit+i}{1,4}{1,1})];
500
501     end
502 elseif isempty(lastposit)==true %The last position is the final
503
504     lastposit=max(size(C));
505     for i=1:(lastposit-initposit-1)
506         FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
507         Recordinf=[Recordinf;str2num(C{1,initposit+i}{1,3}{1,1})];
508         Loadinf=[Loadinf;str2num(C{1,initposit+i}{1,5}{1,1})];
509         cycleinf=[cycleinf;str2num(C{1,initposit+i}{1,4}{1,1})];
510
511     end
512 else
513     for i=1:(lastposit-initposit-1)
514         FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
515         Recordinf=[Recordinf;str2num(C{1,initposit+i}{1,3}{1,1})];
516         Loadinf=[Loadinf;str2num(C{1,initposit+i}{1,5}{1,1})];
517         cycleinf=[cycleinf;str2num(C{1,initposit+i}{1,4}{1,1})];
518
519     end
520 end
521
522
523 %%%Record information for the test: 1 Loaded, 2: Unloaded but
524
525
526 %IMPORTANT:Please do not alter this order. It is important for the
527 %identification of types of recorded information
528
529 numtrac= find(Recordinf==1);hmtrac=numel(numtrac);%how many trac
530 numclamp= find(Recordinf==2);hmclamp=numel(numclamp);%how many clamped
531 numremov= find(Recordinf==3);hmremov=numel(numremov);%how many removed
532 howmanymatrix=[hmtrac,hmclamp,hmremov]; %It corresponds to 1,2,3 types
533
534 nodata=find(howmanymatrix);% It account the number of non-zero elements
535
```

```
disp(' ');
disp(['For this cycle, the avaiable information was obtained in ',...
    num2str(numel(nodata)),' different states: ']);

for j=1:numel(nodata)
switch nodata(j)
    case 1
        disp ('*---Data while the coupon was stressed.');
    case 2
        disp ('*---Data while the coupon was clampled without stress.');
    case 3
        disp ('*---Data when the coupon was removed.');
end
end

disp('See further information in the table below: ');
disp(' ');
disp('   ================================================================');
disp('      STATE    LOAD (Kips)  CYCLES         FILE                    ');
disp('   ________________________________________________________________');
for i=1:numel(FILENAMES)
    switch Recordinf(i)
case 1
    disp(['      Stressed   ',num2str(Loadinf(i)),...
        '    ',num2str(cycleinf(i)),'    ',FILENAMES{i,1},'           ']);
case 2
    disp(['      Clamped   ' ,num2str(Loadinf(i)),...
        '    ',num2str(cycleinf(i)),'    ',FILENAMES{i,1},'']);
case 3
    disp(['      Removed   ' ,num2str(Loadinf(i)),...
        '    ',num2str(cycleinf(i)),'    ',FILENAMES{i,1},'           ']);
    end
end
disp('   ________________________________________________________________');
disp(' ');
disp(' ');
disp(' ');

userexit=input('Pulse 1 to obtain information for a different cycle.')
disp('Pulse 2 to go to the main menu. Pulse 0 to exit: ');
while (isnumeric(userexit)==false || userexit>2 )
userexit=input('Introduce again a correct option between the list ( "0,1,2") : ');
end

end

if userexit==0;
mainuserexit=0;% It refresh the user exit variable.

disp(' ');
disp('  Bye, bye.');
disp(' ');

elseif userexit==2
mainuserexit=1;

end

    end

end %Ending of the main while: mainuserexit
```

Appendix A.4. PIECE1.m

```
%%%%Manuel & Juan Chiachio & Abhinav Saxena NASA ARC. Nov, 2012.
%%%%Piece of code to import the signal data from a especific cycle, given
%%%%an user defined configuration of actuator-sensor-frequency.
%%%*************************************************************************
%%%%% This is to load the information contained within the master sheet
clear all

fid1=fopen('L1S17importNEW.txt','r');% Opens the data file
heading = textscan(fid1,'%s',1,'delimiter','\n');% It discards the heading

i=1; % counter initialization
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
%Successively reads the rows until it find a row without a Data File
%(Column 7)
while isempty(C{1,i}{1,7})== false
    i=i+1;
    C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
end
%
fclose(fid1);


%%%*************************************************************************
cycles=[]; %This storages the cycle list (the full cycles in the fatigue test)
for i=1:max(size(C))-1 % Dim of Matrix C is always 1 plus the actual dimension
cycles=[cycles;str2num(C{1,i}{1,4}{1,1})];
end
%Select the configuration for the signal
actuatorname=1;
sensorname=11;
freq=150;%Selected frequency of the signal: {150,200,250,300,350,400,450} [Hz]
%Select the state (boundary conditions)
bound=1; %1: loaded, 2: Clamped by no load; 3: No stresses.

%*************************************************************************

inpcycle=1200000;% The current cycle is called inpcycle

indsrch=find(cycles==inpcycle, 1, 'first');
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');


FILENAMES={};%This matrix storage the filenames corresponding to the current cycle.

if isempty(initposit)==true %The init. position is 1
initposit=0;
indvect=[];
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end


elseif isempty(lastposit)==true %The last position is the final in the list
indvect=[];
lastposit=max(size(C));
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});ą
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end

else %This is the regular and more frequent case.
indvect=[];
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end
end

```

```
%%%The next script is to select the filenames corresponding to the
%%%specified boundary cond.;

SELCTFILENAMES={};
for k=1:numel(indvect)
SELCTFILENAMES{k}=FILENAMES{indvect(k),1};
end


%%%*********************************************************************
%%%It loads the selected file names to our workspace and save in a struct
%%%variable calle kk
signalactmatrix=[];
signalsensmatrix=[];
for z=1:max(size(SELCTFILENAMES))
kk=importdata(strcat(SELCTFILENAMES{z},'.dat'));

signal_definition=kk.setup.signal_definition;

selected_cell=[];
for i=1:size(signal_definition,2)

if (signal_definition(1,i).actuator==actuatorname && ...
signal_definition(1,i).sensor==sensorname &&...
signal_definition(1,i).frequency1==freq);

gain=signal_definition(1,i).gain;%It takes the gain
selected_cell=[selected_cell;i-1,freq,gain];
end
end

lambsignalact=(getfield(kk,strcat('a',num2str(selected_cell(1,1)))));
lambsignalsens=(getfield(kk,strcat('s',num2str(selected_cell(1,1)))));

signalactmatrix=[signalactmatrix,lambsignalact];
signalsensmatrix=[signalsensmatrix,lambsignalsens];

end

%%%*********************************************************************
%%%This is to display the information by the Matlab Command Window
disp(' ');
disp('   ================================================================');
disp('Check your Matlab Workspace for a matrix called "signalsensmatrix" ');
disp('and "signalactmatrix" for the actuator signals.');
disp('See the correspondence with the files in the matrix "SELCTFILENAMES".');
```

Appendix A.5. PIECE2.m

```
%%%%Manuel & Juan Chiachio & Abhinav Saxena NASA ARC. Nov, 2012.
%%%%Piece of code to import one signal for each different cycle of the
%%%%experiment. The user must introduce the actuator-sensor and frequency
%%%%configuration, together with the boundary condition
%%%*************************************************************************
%%%%% This is to load the information contained within the master sheet
clear all

fid1=fopen('L1S17importNEW.txt','r');% Opens the data file
heading = textscan(fid1,'%s',1,'delimiter','\n');% It discards the heading

i=1; % counter initialization
C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
%Successively reads the rows until it find a row without a Data File
%(Column 7)
while isempty(C{1,i}{1,7})== false
    i=i+1;
    C{1,i} = textscan(fid1, '%s %s %s %s %s %s %s %s %s %s',1,'delimiter',';');
end
%
fclose(fid1);

%%%*************************************************************************
cycles=[]; %This matrix storages the cycle list (the full cycles in the fatigue test)
for i=1:max(size(C))-1 % Dim of Matrix C is always 1 plus the actual dimension
    cycles=[cycles;str2num(C{1,i}{1,4}{1,1})];
end
%Select the configuration for the signal
actuatorname=1;
sensorname=11;
freq=150;%Selected frequency of the signal: {150,200,250,300,350,400,450} [Hz]
%Select the state (boundary conditions)
bound=1; %1: loaded, 2: Clamped by no load; 3: No stresses.

diffcycles=[];
diffcycles(1)=cycles(1);
for i=2:numel(cycles)
    if cycles(i)≠cycles(i-1);diffcycles=[diffcycles;cycles(i)]; end
end

Accactmatrix=[];
Accsensmatrix=[];

for j=1:numel(diffcycles)

inpcycle=diffcycles(j);% The current cycle is called inpcycle
indsrch=find(cycles==inpcycle, 1, 'first');
lastposit= find(cycles>inpcycle, 1, 'first');
initposit= find(cycles<inpcycle, 1, 'last');

FILENAMES={};%This matrix storage the filenames corresponding to the current cycle.
if isempty(initposit)==true %The init. position is 1
initposit=0;
indvect=[];
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end

elseif isempty(lastposit)==true %The last position is the final in the list
indvect=[];
lastposit=max(size(C));
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end

else %This is the regular and more frequent case.
indvect=[];
for i=1:(lastposit-initposit-1)
FILENAMES{i,1}=C{1,initposit+i}{1,7}{1,1};
```

```
FILENAMES{i,2}=str2num(C{1,initposit+i}{1,3}{1,1});
if FILENAMES{i,2}==bound; indvect=[indvect;i]; end
end
end


%%%The next script is to select the filenames corresponding to the
%%%specified boundary cond.

SELCTFILENAMES={};
for k=1:numel(indvect)
SELCTFILENAMES{k}=FILENAMES{indvect(k),1};
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%It loads the selected file names to our workspace and save in a struct
%%%variable calle kk
signalactmatrix=[];
signalsensmatrix=[];
for z=1:max(size(SELCTFILENAMES))
kk=importdata(strcat(SELCTFILENAMES{z},'.dat'));

signal_definition=kk.setup.signal_definition;

selected_cell=[];
for i=1:size(signal_definition,2)

if (signal_definition(1,i).actuator==actuatorname && ...
signal_definition(1,i).sensor==sensorname && signal_definition(1,i).frequency1==freq);

gain=signal_definition(1,i).gain;%It takes the gain
selected_cell=[selected_cell;i-1,freq,gain];
end
end

lambsignalact=(getfield(kk,strcat('a',num2str(selected_cell(1,1)))));
lambsignalsens=(getfield(kk,strcat('s',num2str(selected_cell(1,1)))));

signalactmatrix=[signalactmatrix,lambsignalact];
signalsensmatrix=[signalsensmatrix,lambsignalsens];
end

Accactmatrix=[Accactmatrix,signalactmatrix(:,1)];
Accsensmatrix=[Accsensmatrix,signalsensmatrix(:,1)];

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%This is to display the information by the Matlab Command Window
disp(' ');
disp('==================================================================');
disp('Check your Matlab Workspace for a matrix called "Accsensmatrix"  ');
disp('and "Accactmatrix" for the actuator signals.');
disp('See also the correspondence with the matrix "diffcycles".');
```

References

[1] *ASTM, Standard Test Method for Tension-Tension Fatigue of Polymer Matrix Composite Materials, in D3479/D3479M 96*. ASTM International, 2007.

[2] *ASTM, Standard Test Method for Tensile Properties of Polymer Matrix Composite Materials, in D3039/D3039M 08*. ASTM International, 2008.

[3] Abhinav Saxena, Kai Goebel, Cecilia C. Larrosa, Vishnuvardhan Janapati, Surajit Roy, and Fu-Kuo Chang. Accelerated aging experiments for prognostics of damage growth in composite materials. *International journal for Prognostics and Health Management*, 1(1):1–1, 2008.