Technical note

Exporting Ansys FEM models into Matlab mass and stiffness matrices

How to extract structural matrices (mass,stiffness...) from Ansys and import them into Matlab.
Obtention of elementary cyclic symmetric matrices from an elementary sector in Ansys.

Alain Batailly

McGill University

Structural Dynamics and Vibration Laboratory

May 2008, updated: May 2012

Introduction

The purpose of this note is to explain how to extract stiffness and mass matrices from Ansys. This can be useful for specific calculations with other codes such as Matlab. Section one focuses on structures featuring cyclic symmetric where the stiffness and mass matrices of only one elementary sector allow to obtain the global matrices. Section two presents the Ansys commands to obtain the binary files (***.emat*** and ***.full*** files) containing the matrices. Section three is devoted to the interpretation of these files.

1 Structures featuring cyclic symmetry

Structures featuring cyclic symmetry require to modify some parameters in Ansys before the extraction of the matrices. That is the reason why the case of these structures is given in the first chapter. The particularity of cyclic symmetry is the definition of "boundaries" allowing the expansion of the structure around the axis of symmetry.

1.1 Defining the boundaries : manual procedure

This section explains how to manually define the boundaries of cyclic symmetric structures. However, for most of the cyclic symmetric structures, Ansys will be able to automatically define these boundaries. In that case, you only need to read the second section of this chapter. Each boundary of the structure must have exactly the same number of nodes. The matrices

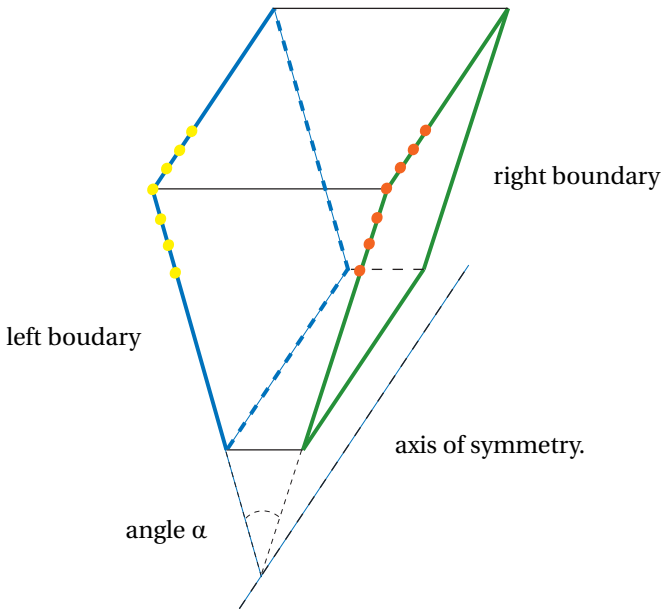

Figure 1 – General representation of cyclic symmetric structures with $\frac{2\pi}{\alpha}$ sectors

given by Ansys correspond to a full sector and so are different from the elementary matrices

required for a cyclic symmetry study. A reorganization of the dofs must be done depending on their belonging to a boundary or not. In our case we consider cylindrical coordinates which actually makes the study easier because the elementary blocks will be obtained directly from the reorganization the dofs. Any other coordinates systems will require modifications of the boundary dofs so that the right boundary of sector i is compatible with the left boundary of sector $i+1$. Ansys can automatically generate the left and right boundaries. But with complex (for instance non planar) boundaries, this option might not include all the nodes. Depending on the orientation of the structure, the terms "CYCLIC_M01H" and "CYCLIC_M01L" are alternate if an error message appears when the analysis starts.

Right boundary

Here are the steps to follow :

❶ click on ***Select*>*Component Manager...*>*Create Component***

❷ quote the line ***Pick entities***

❸ enter the name ***CYCLIC_M01H*** before validation by clicking on ***OK***

❹ select the nodes on the left boundary (see A and the procedure of ***nodes selection***)

❺ validate the selection by clicking on ***Apply***

❻ a new component should appear in the ***Component Manage*** Window named ***CYCLIC_M01H***

Left boundary

Follow the procedure of the previous subsection exchanging ***CYCLIC_M01H*** by ***CYCLIC_M01L***. The boundaries of the cyclic symmetry are now set. The other parameters of the cyclic symmetry study can be entered using the option ***user defined***.

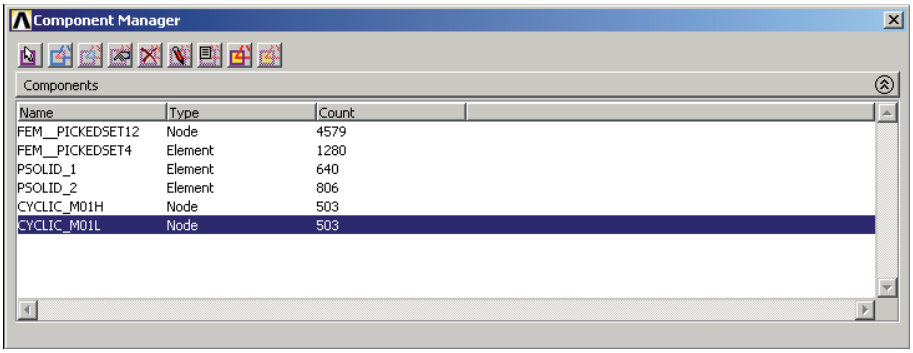

Figure 2 – Window “Component Manager”

1.2 Defining the boundaries : automatic procedure

With most of the cyclic symmetric structures, the ***auto-defined*** option allows to obtain automatically the boundaries. If not, the first thing to do before using the manual procedure is to

modify the ***tolerance*** for detecting the nodes on the boundary.

1.3 Linking the boundaries

Obtaining the list of the nodes on each boundary is quite easy with Ansys as explained in appendix A. However it is critical to check the correspondancy of the nodes obtained in each list such as shown in figure 3. The list of nodes can be written in a file by Ansys sorting them among different criterias (X, Y or Z position, number of nodes...). Instead of complex modifications of

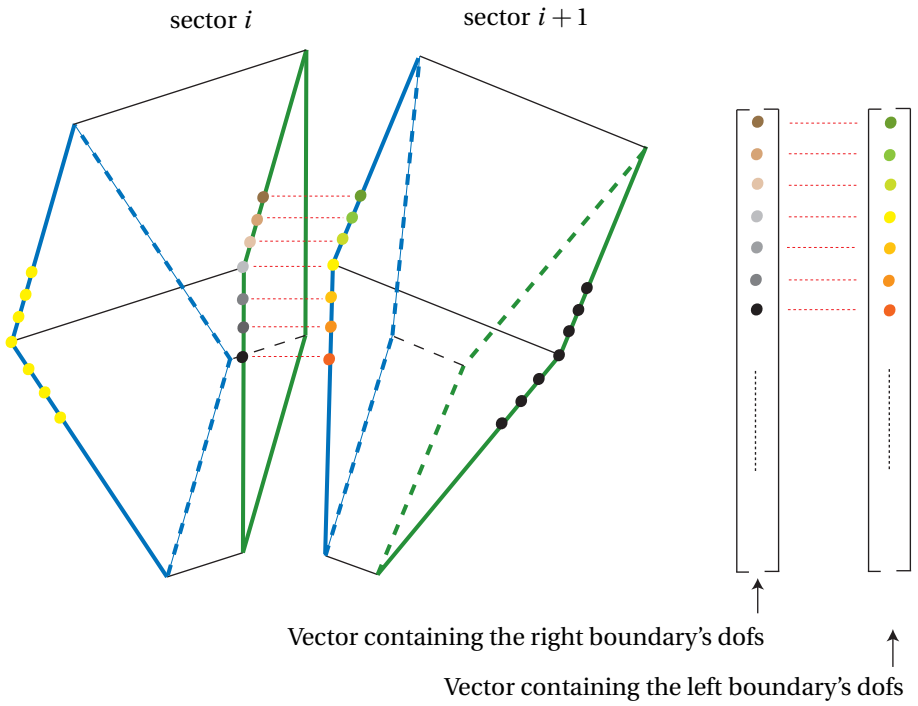

Figure 3 – Representation of the correspondancy of the nodes between the two boundaries

these lists, the use of the ***STATUS*** command in Ansys will directly give the correspondance between the two boundaries. After defining all the parameters for the cyclic symmetry, the command :

$$\underbrace{\textbf{\textit{*STATUS,CYCLIC}}}_{\text{variable}}\text{\_xref\_n}$$

will create the status.lis file containing the correspondance table between the nodes of the two boundaries. After deleting all the blank lines and non-numerical lines in this file, it can be saved in a ***status.txt*** file. The program ***status.m*** given in B is the Matlab program modifying this file and creating from it two vectors : cycg and cycd respectively the nodes on the left and right boundaries (considering the correspondance table : the node at the i^{th} line in vector cycg is facing the node at the i^{th} line in vector cycd.

2 Obtaining the matrices from Ansys

2.1 Writing *.emat* and *.full* files

The procedure for writing ***.emat*** and ***.full*** files containing the mass and stiffness matrices (Harwell-Boeing format) has been detailed in Cedric Rouault's report available in french.

First step Right before launching the analysis the following command must be entered in Ansys : ***EMATWRITE***, ***YES***
the effect of this command is to force Ansys to create a .emat file. Then, a modal analysis must be launched if mass matrix is needed otherwise, a simple static analysis is enough for getting the stiffness matrix. The command ***WRFULL*** makes the analysis stop after writing the files meaning that Ansys won't actually do the analysis. This command can be useful to avoid long calculations with Ansys.

Obtaining the matrices In the "menu" bar, use ***file>list>binary files*** then choose ***matrix***. Type in the window ***File to be read*** the link to the ***.full*** file that has been generated in the previous step. The format must be *ascii*. ***right hand side*** is not useful for getting the matrices. The only thing to do now for each matrix is to give the name wanted for each matrix created in the Ansys work directory.

Getting the node equivalence table In the menus bar : ***file>list>binary files***. then choose ***other***. Select ***Records to be list*** on ***All records*** and ***amt of output per record*** on ***Entire record***. Type in the window ***file to be read*** the link to *.emat* file. The window appearing contains all the information of the analysis on the structure. The nodal equivalence table is in the Record 5. Please refer to the *'Guide to interfacing with Ansys'* for more details on the content of the *.emat* file.

Transferring the matrices in Matlab The matrices written by Ansys are compatible with the Harwell Boeing (HB) format. They must be converted in Matlab format. The program ***sparse matrix converter*** available here.

3 Converting the matrices from HB to Matlab format

3.1 The Cygwin method

As mentionned previously, this step requires a program working under Linux. The following procedure is only necessary if you work under windows environment. A Linux emulator under windows is used : Cygwin (version 3.2.15(13)-release i686-pc-cygwin). The matrices given by Ansys must be transferred in the Cygwin working folder "sparse_matrix_converter". the required commands are detailed on the picture 4. The result is a dat file containing a three column table compatible with "spconvert" function in Matlab.

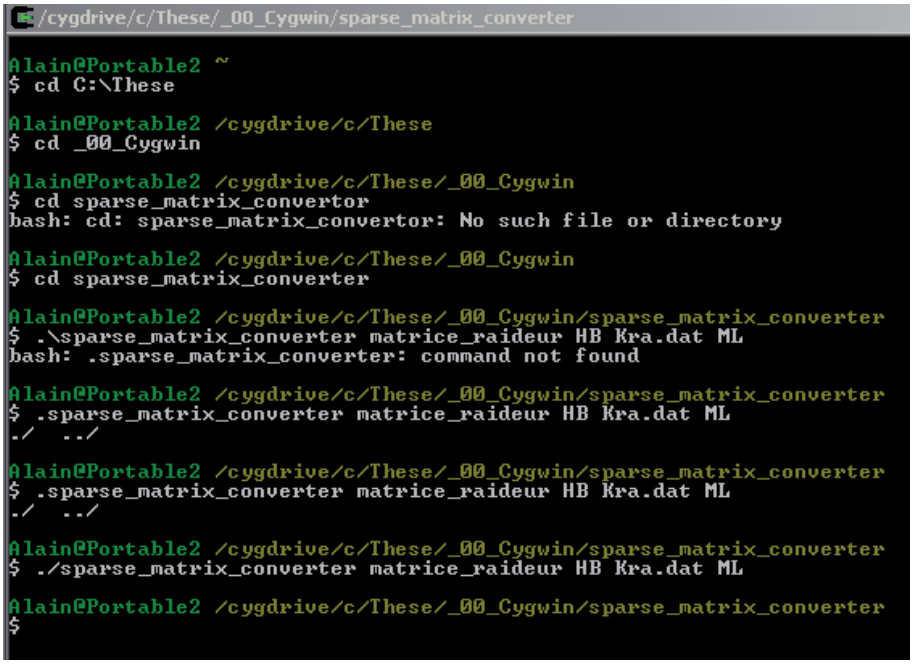

Figure 4 – Commandes Cygwin.

3.2 The RBio Matlab routine

Using the Matlab RBio routine (available here or here) is definitely the fastest way to deal with Harwell-Boeing (HB) or Rutherford-Boeing (RB) format matrices under Matlab. However, the conversion of the matrices from HB/RB format to Matlab format is slightly different from the one obtain in the previous subsection. **Results with matrices from this routine have not been checked.**

4 Description of the matrices

4.1 What Ansys gives

The three matrices which can be obtained from Ansys (stiffness, mass and damping) might have less lines and columns than the total number of dofs of the structure. This comes from the fact that Ansys provides the matrices after considering the boundary conditions. For instance, if some nodes of the structure are built in, then they won't appear in the matrices given by Ansys. However, it is important to notice that the other tables such as the nodal equivalence table contain ***all*** the dofs. Consequently, it is not possible to directly link the matrices with this table.

4.2 Reading the *.emat* file

The ***.emat*** file is a binary file that can be read with Ansys using the following procedure :

❶ Click on ***File*>*List...*>*Binary files...***[1]

❷ Choose ***Other*>*ok***

❸ In ***Records to be listed***, choose ***All records***. This field allows to choose the data to be plotted among all the available ones in the ***.emat*** file. Fields 1 to 5 are the ones to focus on

❹ In order to plot specific lists or data, choose ***Specified records*** then type the number of the first list to plot in the ***NSTRT Starting record*** field and the number of the last list to plot in the ***Nstop ending record*** field

❺ In the ***FORM*** field, choose the option ***entire records***. Without this option, Ansys will only write the header of each list without the content

❻ in ***[FILEAUX2] Binary file to list***, point at the folder containing the ***.emat*** file and click ***ok***

❼ The file should now appears in a separate windows

4.3 Interpreting the *.emat* file

Figure 5 shows the Ansys window where is written the ***.emat*** file. Only the 5 first records of this file will be detailed in this section.

Record 1 This list contains general information on the ***.emat*** file such as creation time of the analysis...

Record 2 this list is actually a table containing data linked with the analysis. In our case, here is the meaning of some of the number appearing :
- 1446 : number of elements
- 3 : number of dofs per node
- 24417 : total number of dofs...

More details about this record can be found in the "ANSYS Interfacing Guide" (chapter 1.6.2 page 1-29).

Record 3 This list is relative to time as a parameter of a dynamic analysis.

Record 4 Only a few details are given about this record. It is mainly a code linked with the nature of the dofs for each node. In our case, (1,2,3) means three displacements in a cartesian basis.

Record 5 **This list is essential for the analysis of the matrices given by Ansys** The list is composed of 5 columns of numbers plus a sixth column containing indices of position. It must be seen as a classical single column vector that would be spread on 5 columns to save some space. In the case of figure 5, the list represents a vector "**V**" which dimensions are $[8139 \times 1]$. This must be read as follow : ***node*** $\mathrm{V}(i,1)$ ***(number of the node that would appear in Ansys while selected) is actually at the*** i^{th} ***line/column of the matrices.*** This information is linked with the nodes, in order to have the position of the dofs we must know that node $\mathrm{V}(i,1)$ is linked to the dofs $3\mathrm{V}(i)-2$, $3\mathrm{V}(i)-1$ and $3\mathrm{V}(i)$. This dofs will

1. Do not pay attention to the error message *This function requires that files currently open be closed. Executing the function will exit the preprocessor* that might appear

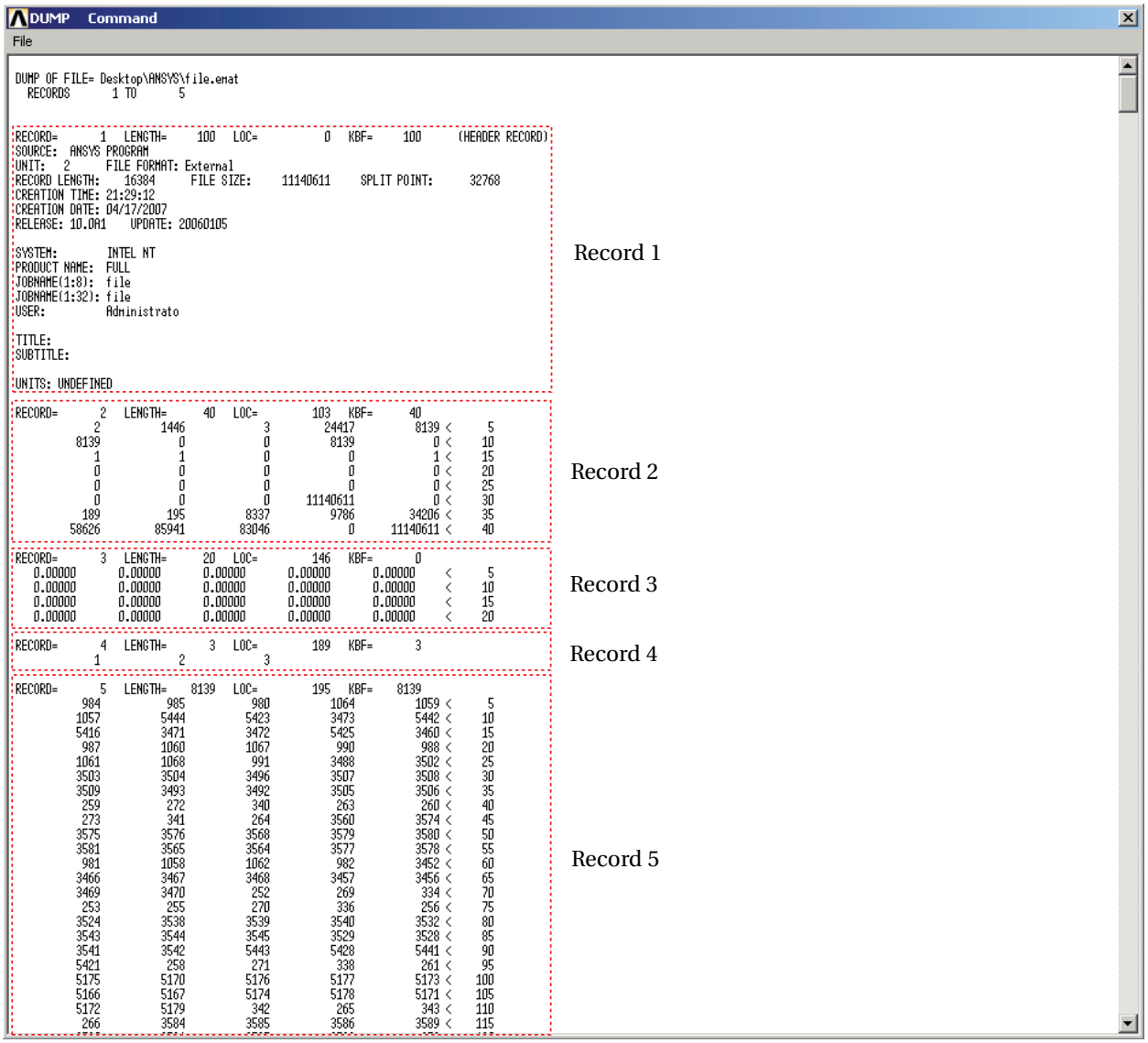

Figure 5 – Description of the 5 first "record" of the ".emat" file.

be found on lines/columns $3i-2$, $3i-1$ and $3i$ of the **full** matrices (meaning that there is a shift in these numbers if some nodes are built in for instance).

5 From Ansys to Matlab : general program

This chapter briefly describes the Matlab program used to calculate the eigenfrequencies of a cyclic symmetric structure created with Ansys. The inputs of this program are :

- the mass and stiffness matrices of the structure's elementary sector (see 4),
- the list of built-in nodes : encas.txt (see A).
- the status.txt file (see B),

– the record5 vector modified as a column vector in a ".txt" file (see C),

```
clear;
    clc;
    % Reading status.lis file which non numerical lines have been removed
    % The extension of the file is now ".dat"
load status.txt;
    A=size(status);
    a=A(1,1);
    b=A(1,2);
    % We only keep the columns giving the node numbers
    Cor1=status(:,3:4);
    % The lines containing virtual negative values are deleted
    k=1;
    for i=1:a
        if (Cor1(i,2)<=0)
            if (floor(i/2)==i/2)
                Cor1(i,2)=0;
                Cor1(i-1,2)=0;
            else
                Cor1(i,2)=0;
                Cor1(i+1,2)=2;
            end
        end
    end
    % Fitting in two columns
    for i=1:a/2
        Cor2(i,1)=Cor1(2*i-1,2);
        Cor2(i,2)=Cor1(2*i,2);
    end
    % Deleting the "0" lines
    for i=1:a/2
        if Cor2(i,1)>0
            Correspondance(k,1)=Cor2(i,1);
            Correspondance(k,2)=Cor2(i,2);
            k=k+1;
        end
    end
    % Checking
    size(Correspondance)
```

```
    % The "Correspondance" vector actually contains the two boundaries:
    % cyc_g and cyc_d
    % The boundary "M01H" is the left column, the boundary "M01L" is the
    % right column.
    cycg=Correspondance(:,2);
    cycd=Correspondance(:,1);
    save cycg.txt cycg -ascii
    save cycd.txt cycd -ascii
    save Correspondance.txt Correspondance -ascii
    % Memory clear
    clear;clc;
%
% Creating the cyclic symmetry elementary blocks: K0, K1, M0 and M1
%

load encas.txt
    encas_ddl=zeros(3*length(encas),1);
    for i=1:length(encas)
        encas_ddl(3*(i-1)+1,1)=3*encas(i,1)-2;
        encas_ddl(3*(i-1)+2,1)=3*encas(i,1)-1;
        encas_ddl(3*(i-1)+3,1)=3*encas(i,1);
    end;
    % Considering built-in nodes : modifications of the numbers
load record5.txt % loading Record5.txt from Ansys
    col=record5;
    col_ddl=zeros(3*length(col),1);
    for i=1:length(col)
        col_ddl(3*(i-1)+1,1)=3*col(i,1)-2;
        col_ddl(3*(i-1)+2,1)=3*col(i,1)-1;
        col_ddl(3*(i-1)+3,1)=3*col(i,1);
    end;
    kk=1;
    for i=1:length(col_ddl)
        test=0;
        % is dof "i" built in ?
        for j=1:length(encas_ddl)
            if (col_ddl(i,1)==encas_ddl(j,1))
                test=1;
            end;
        end;
        % if dof "i" is built in then here is its actual position in the
```

```
    % matrices given by Ansys
    if (test==0)
        col2_ddl(kk,1)=col_ddl(i,1);
        kk=kk+1;
    end;
end;
% Modifying CYCD
load cycd.txt

% Deleting nodes appearing twice (Ansys bug)
k=1;
cycd_corrige(1,1)=cycd(1,1);
for i=2:length(cycd)
    testb=0;
    testb=size(find(cycd(1:i-1,1)==cycd(i,1)));
    if testb(1,1)==0
        cycd_corrige(k,1)=cycd(i,1);
        k=k+1;
    end
    test=0;
end
clear cycd
cycd=cycd_corrige;
clear cycd_corrige

cycd_ddl=zeros(3*length(cycd),1);
for i=1:length(cycd)
    cycd_ddl(3*(i-1)+1,1)=3*cycd(i,1)-2;
    cycd_ddl(3*(i-1)+2,1)=3*cycd(i,1)-1;
    cycd_ddl(3*(i-1)+3,1)=3*cycd(i,1);
end;
jj=1;
for i=1:length(cycd_ddl)
    for j=1:length(col2_ddl(:,1))
        if (col2_ddl(j,1)==cycd_ddl(i,1))
            cycd2_ddl(jj,1)=j;
            jj=jj+1;
        end;
    end;
end;
% CYCD2_DDL vector contains the position of the right
```

```
% boundary dofs in the matrices given by Ansys
%
% Modifying CYCG
load cycg.txt

% Deleting nodes appearing twice (Ansys bug)
k=1;
cycg_corrige(1,1)=cycg(1,1);
for i=2:length(cycg)
    testb=0;
    testb=size(find(cycg(1:i-1,1)==cycg(i,1)));
    if testb(1,1)==0
        cycg_corrige(k,1)=cycg(i,1);
        k=k+1;
    end
    test=0;
end
clear cycg
cycg=cycg_corrige;
clear cycg_corrige

cycg_ddl=zeros(3*length(cycg),1);
for i=1:length(cycg)
    cycg_ddl(3*(i-1)+1,1)=3*cycg(i,1)-2;
    cycg_ddl(3*(i-1)+2,1)=3*cycg(i,1)-1;
    cycg_ddl(3*(i-1)+3,1)=3*cycg(i,1);
end;
jj=1;
for i=1:length(cycg_ddl)
    for j=1:length(col2_ddl(:,1))
        if (col2_ddl(j,1)==cycg_ddl(i,1))
            cycg2_ddl(jj,1)=j;
            jj=jj+1;
        end;
    end;
end;
% CYCG2_DDL vector contains the position of the left
% boundary dofs in the matrices given by Ansys
%
% Creating the inversion matrix (reorganizing lines and columns)
k1=0;
```

```
u=1;
compteur=0;
clear vect_int
vect_int=zeros(length(col2_ddl)-length(cycd2_ddl)-length(cycg2_ddl),1);
kk=1;
for i=1:length(col2_ddl)
    test=0;
    for j=1:length(cycg2_ddl)
        if (cycg2_ddl(j,1)==i)
            test=1;
        end;
        if (cycd2_ddl(j,1)==i)
            test=1;
        end;
    end;
    if (test==0)
        vect_int(kk,1)=i;
        kk=kk+1;
    end;
end;
% VECT_INT contains the position of all the dofs being "internal"
% (neithe boundary conditions neither cyclic boundaries)
temp=zeros(length(col2_ddl),3);
for i=1:length(cycg2_ddl)
    temp(i,1)=i;
    temp(i,2)=cycg2_ddl(i,1);
    temp(i,3)=1;
end;
for i=length(cycg2_ddl)+1:length(col2_ddl)-length(cycd2_ddl)
    temp(i,1)=i;
    temp(i,2)=vect_int(i-length(cycg2_ddl),1);
    temp(i,3)=1;
end;
for i=length(col2_ddl)-length(cycd2_ddl)+1:length(col2_ddl)
    temp(i,1)=i;
    temp(i,2)=cycd2_ddl(i-length(col2_ddl)+length(cycd2_ddl),1);
    temp(i,3)=1;
end;
% Reorganizing mass and stiffness matrices
L=spconvert(temp);
% Kra.dat and Mra.dat are stiffness and mass matrices given by Ansys
```

```
    % after conversion from HB format to Matlab format.
load Kra.dat load Mra.dat
    K=spconvert(Kra);
    M=spconvert(Mra);
    Kreo=L*K*L';
    Mreo=L*M*L';
    % Defining the blocks
    Mgg=Mreo(1:length(cycg2_ddl),1:length(cycg2_ddl));
    Mgi=Mreo(1:length(cycg2_ddl),length(cycg2_ddl)+1:length(cycg2_ddl)+...
        length(vect_int));
    Mgd=Mreo(1:length(cycg2_ddl),length(cycg2_ddl)+length(vect_int)+1:end);
    Mii=Mreo(length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int),...
        length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int));
    Mid=Mreo(length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int),...
        length(cycg2_ddl)+length(vect_int)+1:end);
    Mdd=Mreo(length(cycg2_ddl)+length(vect_int)+1:end,length(cycg2_ddl)+...
        length(vect_int)+1:end);
    Kgg=Kreo(1:length(cycg2_ddl),1:length(cycg2_ddl));
    Kgi=Kreo(1:length(cycg2_ddl),length(cycg2_ddl)+1:length(cycg2_ddl)+...
        length(vect_int));
    Kgd=Kreo(1:length(cycg2_ddl),length(cycg2_ddl)+length(vect_int)+1:end);
    Kii=Kreo(length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int),...
        length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int));
    Kid=Kreo(length(cycg2_ddl)+1:length(cycg2_ddl)+length(vect_int),...
        length(cycg2_ddl)+length(vect_int)+1:end);
    Kdd=Kreo(length(cycg2_ddl)+length(vect_int)+1:end,length(cycg2_ddl)+...
        length(vect_int)+1:end);
    % Defining elementary cyclic symmetry blocks
    LL=speye(length(col2_ddl)-2*length(cycg2_ddl),length(col2_ddl)-2*...
        length(cycg2_ddl));
    for i=1:length(col2_ddl)-2*length(cycg2_ddl)
        LL(i,i)=0;
    end;

    a=size(Mid);
    MAT_O=zeros(length(cycg2_ddl),a(1,2));

    MO=[Mgg+Mdd Mgi; Mgi' Mii];
    M1=[Mgd MAT_O;Mid LL];
    KO=[Kgg+Kdd Kgi; Kgi' Kii];
    K1=[Kgd MAT_O;Kid LL];
```

```
save BLOCS K0 M0 K1 M1

% Harmonic blocks and calculation of the eigenfrequencies

K_0=K0+(K1+K1');
M_0=M0+(M1+M1');

clear LL Kdd Kid Kii Kgd Kgg Kreo Mreo Mgg Mgi Mgd Mii Mid Mdd K M temp

[Z,W]=eigs(K_0,M_0,10,'sm');
F=sqrt(diag(W))/2/pi;
clear K_0 M_0
clear K0 M0 K1 M1
nd=0;
file=['Blocs_harmonique_',num2str(nd)]
save (file,'F','Z','W')
nd
for nd=1:27
    nd
    load BLOCS
    K=[K0+(K1+K1')*cos(nd*2*pi/56) (K1-K1')*sin(nd*2*pi/56); ...
        (K1'-K1)*sin(nd*2*pi/56) K0+(K1+K1')*cos(nd*2*pi/56)];
    M=[M0+(M1+M1')*cos(nd*2*pi/56) (M1-M1')*sin(nd*2*pi/56); ...
        (M1'-M1)*sin(nd*2*pi/56) M0+(M1+M1')*cos(nd*2*pi/56)];
    clear K0 K1 M0 M1
    [Z,W]=eigs(K,M,10,'sm');
    F=sqrt(diag(W))/2/pi;

    clear K M

    file=['Blocs_harmonique_',num2str(nd)]
    save (file,'F','Z','W')

    clear F
end

load BLOCS
K_0=K0-(K1+K1');
M_0=M0-(M1+M1');
clear K0 K1 M0 M1
```

```
    [Z,W]=eigs(K_0,M_0,10,'sm');
    F=sqrt(diag(W))/2/pi;
    clear K_0 M_0
    nd=28;
    nd
    file=['Blocs_harmonique_',num2str(nd)]
    save (file,'F','Z','W')
```

A Nodes selection procedure

This procedure describes how to obtain a file containing a list of nodes picked manually using Ansys interface. This can also be useful to get any kind of list such as built-in nodes or nodes on a boundary. On the main Ansys window, follow these steps :

1. Click on ***Select>Entities...***
2. Check that the following options are selected : ***Nodes***, ***By Num/Pick*** and ***From Full*** and Click on ***ok***
3. Pick with the cursor (one by one or a group using the ***box*** option) the nodes you need
4. Click on ***Apply*** and Click on ***ok***
5. Click on ***List>Nodes...***
6. Choose your organization options and click on ***ok*** : the list of the selected nodes appear in an Ansys window
7. Save it with the following extension : ***.lis***. In order to use these files with Matlab they must be modified by deleting all the non numerical lines.

B status.m program

```
clear;
    clc;
    % Reading status.lis file which non numerical lines have been removed
    % The extension of the file is now ".txt"
load status.txt;
    A=size(status);
    a=A(1,1);
    b=A(1,2);
    % We only keep the columns giving the node numbers
    Cor1=status(:,3:4);
    % The lines containing virtual negative values are cancelled
    k=1;
    for i=1:a
```

```
        if (Cor1(i,2)<=0)
            if (floor(i/2)==i/2)
                Cor1(i,2)=0;
                Cor1(i-1,2)=0;
            else
                Cor1(i,2)=0;
                Cor1(i+1,2)=2;
            end
        end
    end
    % Fitting in two columns
    for i=1:a/2
        Cor2(i,1)=Cor1(2*i-1,2);
        Cor2(i,2)=Cor1(2*i,2);
    end
    % Deleting the "0" lines
    for i=1:a/2
        if Cor2(i,1)>0
            Correspondance(k,1)=Cor2(i,1);
            Correspondance(k,2)=Cor2(i,2);
            k=k+1;
        end
    end
    % Checking
    size(Correspondance)
    % The "Correspondance" vector actually contains the two boundaries:
    % cyc_g and cyc_d
    % The boundary "M01H" is the left column, the boundary "M01L" is the
    % right column.
    cycg=Correspondance(:,2);
    cycd=Correspondance(:,1);
    save cycg.txt cycg -ascii
    save cycd.txt cycd -ascii
    save Correspondance.txt Correspondance -ascii
    % Memory clear
    clear;clc;
```

C record5.m program

The following code is a very shot program to reorganize the record5.txt file obtained from Ansys.

```
clear; clc;
% This program requires the file to be cleared (last two columns filled
% with '<' and line numbers must be erased). "0" must be added if the last
% line contains less than 5 numbers.
load record_5.txt a=length(record_5);
b=length(find(record_5(a,:)>0)); L=5*(a-1)+b;
record_modify=zeros(L,1);
```

```
for i=1:5
    for j=1:a
        k=(j-1)*5+i;
        if k>L
            save record_modify.txt record_modify -ascii
            break;
        end
        record_modify(k,1)=record_5(j,i);
    end
end

length(record_modify)
nnz(record_5)
nnz(record_modify)
```