Universitat Politècnia de Catalunya

ESEIAAT

DEGREE IN AEROSPACE TECHNOLOGY ENGINEERING

Assignment 4

ELASTODYNAMIC ANALYSIS OF A CANTILEVER BOX BEAM

Author
Bernardo Márquez López

Professor
Joaquín Hernández Ortega
Department of materials and structural resistance in engineering

December 2022

Contents

1 Introduction

This assignment has the aim of developing a modal analysis of the structure studied in assignment 3, consistent in a cantilevier square section box beam. Therefore, an elastodynamic analysis of the structure will be developed in order to study its oscillatory behaviour until the new static equilibrium when an applied load is instantaneously withdrawn.

2 Mass matrix assembly and computation

In the study developed in the assignment, the moviment equations of the system in elasdynamic conditions have the following form:

$$M \cdot \ddot{d} + D \cdot \dot{d} + K \cdot d - F = 0 \tag{1}$$

Where M is the mass matrix, D is the damping, K is the stiffness matrix, F is the forces vector and d represents the displacements of the structure.

First of all, the Mass matrix needs to be implemented in the code from assignment 3 for the elastodynamic analysis.The Matlab code that operates the assembly of this matrix is shown in appendix A.1. The assembly process of this matrix is similar to the one developed for the stiffness matrix in the previous assignment, making use of an auxiliary function *ComputeMeMatrix* that determines the mass matrix of each element. The algorithm computed in that implementation consist in the discretization of the following integral using using finite Gauss quadrature for 2D approximation:

$$M^e := \int_{\Omega_e} \rho N^{eT} N^e d\Omega \tag{2}$$

3 Modes and natural frequencies of the system

Once the mass matrix has been assemblied with the process explained, a new Matlab script has been developed in order to determine the modes of the system, which will allow to study the evolution of its amplitud of vibration over time. This code is shown in Appendix A.4.

The first step followed by this code is the determination of the eigenvalues of the modes (given by the matrix Φ), by solving the follwing equation:

$$(K - \omega^2 \cdot M) \cdot \Phi = 0 \tag{3}$$

The Matlab routine $UndampedFREQ$ computes the modes and frequencies of the system in the case of free vibrations, solving this equation.
Then, the time amplitud t_0 is defined in the code and determined for the different modes of the system by computing the mass matrix M and the displacements. Finally, the displacements for the simulation are calculated by the program using the following expression:

$$d = \sum_{i=0}^{n} \Phi_i \cdot e^{-\overline{\xi}_i \cdot \omega_i \cdot t} \cdot \left(q_i^0 \cdot \cos\left(\overline{\omega}_i \cdot t\right) + \frac{\dot{q}_i^0 + \overline{xi}_i \cdot \omega_i \cdot q_i^0}{\overline{\overline{\omega}_i}} \cdot \sin\left(\overline{\omega}_i \cdot t\right) \right) \tag{4}$$

A Appendix

A.1 Code for the Assembly of Mass Matrix (M)

```
function M = ComputeM(COOR,CN,TypeElement,densglo)
%%%%
% This subroutine   returns the mass stiffness matrix M (ndim*nnode x ndim*nnode)
% Inputs:   COOR: Coordinate matrix (nnode x ndim),
% CN: Connectivity matrix (nelem x nnodeE),
% TypeElement: Type of finite element (quadrilateral,...),
% densglo (nstrain x nstrain x nelem) : Array of density values
% Dimensions of the problem
if nargin == 0
    load('tmp1.mat')
end
nnode = size(COOR,1);
ndim = size(COOR,2);
nelem = size(CN,1);
nnodeE = size(CN,2);

% nstrain = size(celasglo,1) ;
% Shape function routines (for calculating shape functions and derivatives)
TypeIntegrand = 'K';
[weig,posgp,shapef,dershapef] = ComputeElementShapeFun(TypeElement,nnodeE,TypeIntegrand) ;
% Assembly of matrix M
% ----------------
M = zeros(nnode*ndim);
for e = 1:nelem
    dens = densglo(:,:,e) ;  % Stiffness matrix of element "e"
    CNloc = CN(e,:) ;          % Coordinates of the nodes of element "e"
    Xe = COOR(CNloc,:)' ;      % Computation of elemental stiffness matrix
    Me = ComputeMeMatrix(dens,weig,dershapef,Xe);
    submatrix=zeros(nnodeE*3,1);
    for n=1:nnodeE
    submatrix((n-1)*3+1,1)=(CN(e,n)-1)*3+1;
    submatrix((n-1)*3+2,1)=(CN(e,n)-1)*3+2;
    submatrix((n-1)*3+3,1)=(CN(e,n)-1)*3+3;
    end

    M(submatrix,submatrix,1)=M(submatrix,submatrix,1)+Me;
end
```

A.2 ComputeMeMatrix Function

```
function Me = ComputeMeMatrix(dens,weig,shapef,dershapef,Xe) ;
% Given
% celas: Elasticity Matrix
% weig :   Vector of Gauss weights (1xngaus),
% dershapef:   Array with the derivatives of shape functions, with respect to  element ...
    coordinates (ndim x nnodeE x ngaus),
% Xe: Global coordinates of the nodes of the element,
% this function returns the element stiffness matrix Ke
ndim = size(Xe,1) ;
ngaus = length(weig) ;
nnodeE = size(Xe,2)  ;
Me = zeros(nnodeE*ndim,nnodeE*ndim) ;

for  g = 1:ngaus
    % Matrix of derivatives for Gauss point "g"
    BeXi = dershapef(:,:,g) ;
    NeSCL = shapef(g,:);
    % Jacobian Matrix
```

```
    Je = Xe*BeXi' ;
    % JAcobian
    detJe = det(Je) ;
    % Matrix of shape functions at point g
    Ne = StransfN(NeSCL,ndim) ;
    % Me assembly
    Me = Me + weig(g)*detJe*(Ne'*dens*Ne) ;
end
```

A.3 Undamped Vibrations code

```
 function [MODES FREQ] = UndampedFREQ(M,K,neig)
 % neig = Number of modes to be calculated
%  [MODES EIGENVAL] = eigs(M,K,neig) ;
%  EIGENVAL = diag(EIGENVAL) ;
%  FREQ = sqrt(1./EIGENVAL) ;
if nargin == 0
    load('tmp.mat')

end


% We turn M and K symmetric, because otherwise
% the modes are not orthogonalized with respect to M
% It should be noted that the lack of symmetry is caused by finine machine precision:   ...
    theoretically speaking,
% M and K are symmetric by construction
M = (M+M')/2;
K = (K+K')/2;


  [MODES EIGENVAL] = eigs(K,M,neig,'sm') ;
  EIGENVAL = diag(EIGENVAL) ;
  FREQ = sqrt(EIGENVAL) ;

  [FREQ,imodes] = sort(FREQ) ;
  MODES= MODES(:,imodes);

 end
```

A.4 Code for studyin the Elastodynamic analysis

```
 clc;
clear;

%   PROGRAM INPUTS

clc;
clear;

%   PROGRAM INPUTS

neig = 25;      % number of modes
xi_mod = 0.01; % Damping ratio
m = 40;        % m times the natural period of the system for the simulation
steps = 2000;   % qty of steps for the simulation
NAME_INPUT_DATA = 'SIMULATION'; % Name for the simulation output file

%   SOLVER
load('INFO_FE.mat');
[MODES, omega_av] = UndampedFREQ(M(DOFl, DOFl),K(DOFl,DOFl),neig);
```

```
20
21  GidPostProcessModes(COOR,CN,TypeElement,MODES,posgp,NameFileMesh,DATA,DOFl);
22
23  t0 = MODES.'*M(DOFl,DOFl)*d(DOFl);
24  dt0 = zeros(neig, 1);
25  xi = xi_mod*ones(neig,1);
26  omega_av = zeros(size(omega_av));
27
28  for i = 1:length(omega_av)
29      omega_av(i) = omega_av(i)*sqrt(1-xi(i)^2);
30  end
31
32
33  % Plot amplitudes for each mode
34  bar(abs(t0));
35  title('Amplitude for each mode',Interpreter='latex');
36  xlabel('Modde',Interpreter='latex');
37  ylabel('Amplitude [rad/s]',Interpreter='latex');
38
39  % Determination of displacements over time
40
41  tstep = (m*2*pi/abs(omega_av(1)))/steps;
42  DISP = zeros(size(M,1), steps);
43
44  for nstep = 1:steps
45      t = nstep*tstep;
46      for mode = 1:neig
47          e_val = exp(-xi(mode)*omega_av(mode)*t);
48          sin_coeff_val = (dt0(mode) + xi(mode)*omega_av(mode)*t0(mode))/omega_av(mode);
49          DISP(DOFl, nstep) = DISP(DOFl, nstep) + ...
                MODES(:,mode)*e_val*(t0(mode)*cos(omega_av(mode)*t) + ...
                sin_coeff_val*sin(omega_av(mode)*t));
50      end
51  end
52
53  t = 0:tstep:(m*2*pi/abs(omega_av(1))-tstep);
54
55  GidPostProcessDynamic(COOR,CN,TypeElement,DISP,NAME_INPUT_DATA,posgp,NameFileMesh,t);
```