Computational Structural Mechanics and Dynamics
Assignment 6 - Beams

Federico Valencia Otálvaro

Master's in Numerical Methods in Engineering
Universitat Politècnica de Catalunya

March 26^{th}, 2020

Contents

1 Implementation of Reduced Integration for Two-nodes Timoshenko Beam Element Shear Stiffness Matrix

In order to implement reduced integration for the shear stiffness matrix, the only change that had to be made in the $Beam\_Timoshenko\_v1\_3.m$ Matlab routine (see Appendix A.1) was to include the following sitffness matrix:

$$K_s = \frac{GA^*}{l}\begin{bmatrix} 1 & \frac{l}{2} & -1 & \frac{l}{2} \\ & \frac{2}{4} & -\frac{l}{2} & \frac{l^2}{4} \\ & & 1 & -\frac{l}{2} \\ Sym & & & \frac{l^2}{4} \end{bmatrix} \tag{1}$$

2 Simply Supported Beam Solved with Euler and Timoshenko (Full and Reduced Integration) MatFEM Programs

The problem consists of a simply supported beam with a span of 4 units, discretized into 64 elements of length $l = \frac{1}{16}$. The beam is subjected to a unitary distributed vertical load along its entire length. The cross section is a square of side length a, which takes the following values for different analysis:

a	0.001	0.005	0.01	0.02	0.05	0.1	0.2	0.4
a/L	$\frac{1}{4000}$	$\frac{1}{800}$	$\frac{1}{400}$	$\frac{1}{200}$	$\frac{1}{80}$	$\frac{1}{40}$	$\frac{1}{20}$	$\frac{1}{10}$

Appendix A.2 contains the input file which has all the problem information. In order to compare the obtained results, graphics for maximum displacements, moments and shear for every value of a will be presented.

2.1 Displacement

The maximum displacements were measured at the center of the span (node 33) for all cases. Since all vertical displacements are negative and the range of values is very large, their absolute values were taken in order to enable plotting in logarithmic scale.

It may be observed that for very slender beams, the displacements of the Timoshenko beam with full integration for shear are smaller than the ones obtained in the other two cases. When the a/L ratio of the beam reaches 0.25, all three models start exhibiting very similar displacement values. However, the Timoshenko beam with reduced integration and the Euler beam exhibit practically identical displacement values for any a/L ratio.

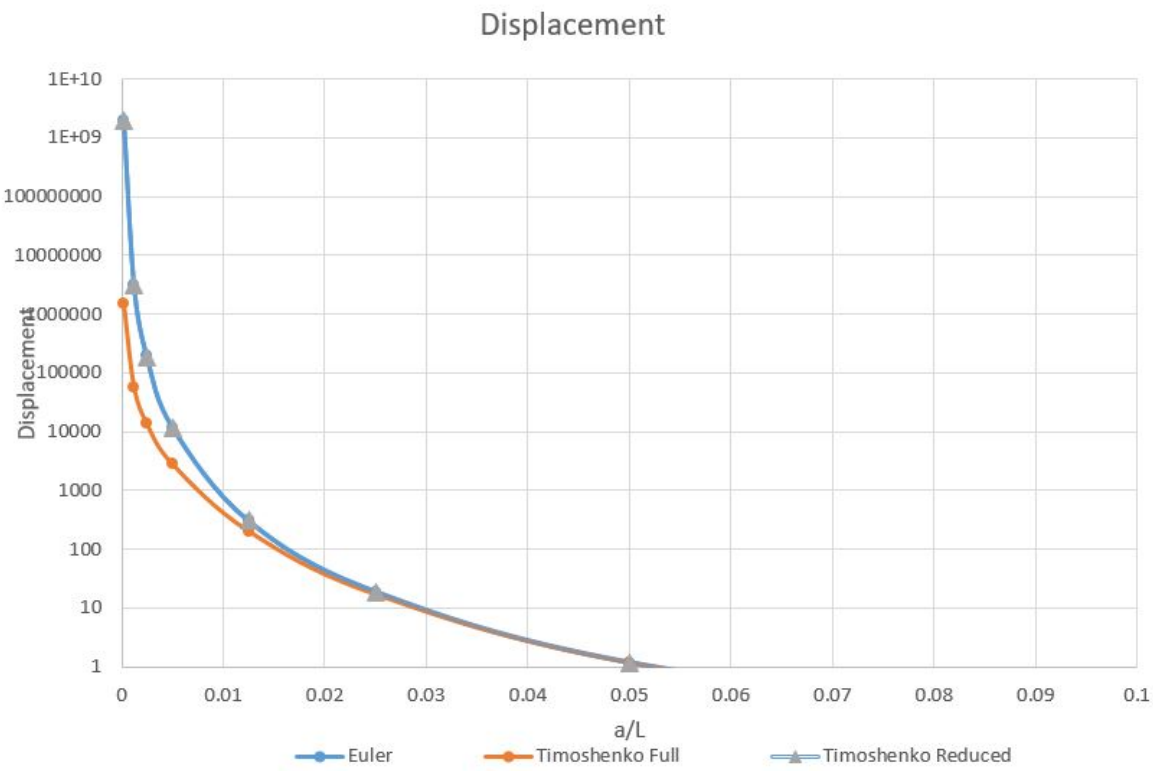

Figure 1: Max. Displacements vs. a/L

2.2 Moments

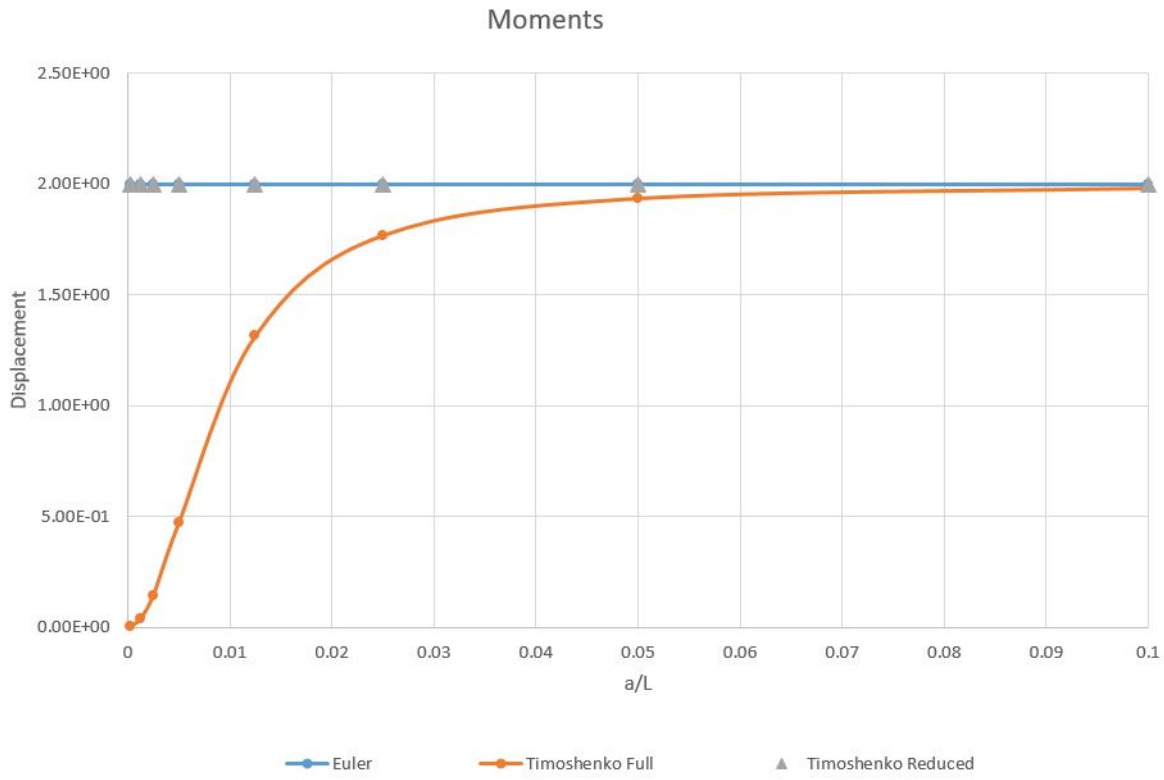

Figure 2: Max. Moment vs. a/L

Just like it was the case for displacements, the moments obtained for the Timoshenko beam with reduced integration and the Euler beam are almost identical, while they show a significant difference with the Timoshenko beam with full integration for highly slender beams. The analytical expression for the moment at the center of the span on a simply supported beam subjected to uniform distributed load W is given by:

$$M = \frac{WL^2}{8} = \frac{(1)(4^2)}{8} = 2 \tag{2}$$

The moments obtained for the Timoshenko beam with full integration start being similar to the analytical moment for $a/L > 0.05$, while the other two models exhibit accurate results for any a/L ratio.

2.3 Shear

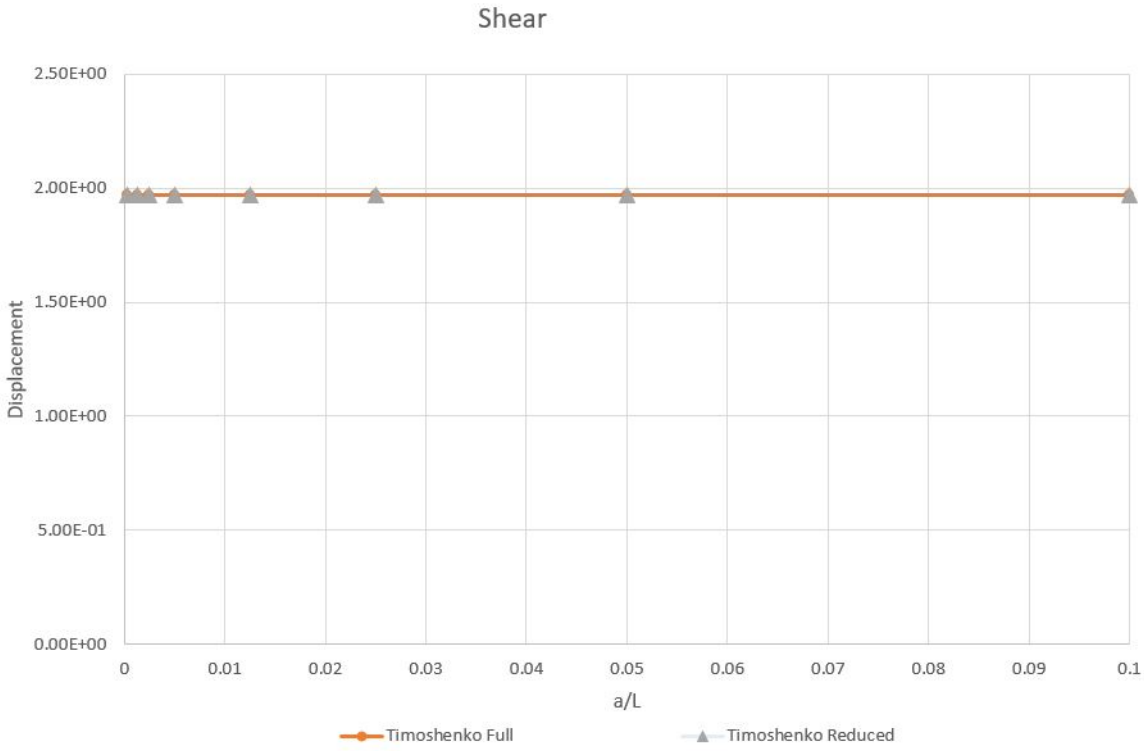

Figure 3: Max. Shear vs. a/L

The maximum shear values of a simply supported beam under a uniform distributed load should be located at the supports of the beam and have a magnitude equal to the support reaction producing the shear on the beam. In this case, both supports should have a vertical reaction of magnitude 2, which means that the results obtained with both Timoshenko beam models are accurate for any slenderness ratio, since they remain constant for any a/L. However, it was not possible to evaluate shear for the Euler beam model.

A Appendices

A.1 Modified Routine For Thimoshenko Reduced Integration

```
%% MAT-fem_Beams
% 2 Nodes Beam using Timoshenko Theory

% Clear memory and variables
  clear

% The variables are read as a MAT-fem subroutine
% young   = Young Modulus
% poiss   = Poission Ratio
% denss   = Material density
% area    = Cross section area
% inertia = Cross section inertia
% coordinates = [ x ] matrix size: nnode x ndime (1)
% elements    = [ inode , jnode ] element connectivity matrix.
%               Matrix size: nelem x nnode; nnode = 2
% fixnodes    = [ node number , dof , fixed value ] matrix with
%               Dirichlet restrictions, were dof=1 for vertical
%               displacement and dof=2 for vertical derivative
% pointload   = [ node number , dof , load value ] matrix with
%               nodal loads, were dof=1 for vertical load
%               and dof=2 for moment
% uniload     = [ uniform vertical load ] sparse matrix size: nelem x 1

  file_name = input('Enter the file name: ','s');

  tic;                    % Start clock
  ttim = 0;               % Initialize time counter
  eval(file_name);        % Read input file

% Find basic dimensions
  npnod = size(coordinates,1);          % Number of nodes
  nelem = size(elements,1);             % Number of elements
  nnode = size(elements,2);             % Number of nodes por element
  dofpn = 2;                            % Number of DOF per node
  dofpe = nnode*dofpn;                  % Number of DOF per element
  nndof = npnod*dofpn;                  % Number of total DOF

  ttim = timing('Time needed to read the input file',ttim); %Reporting time

% Dimension the global matrices
  StifMat  = sparse( nndof , nndof );  % Create the global stiffness matrix
  force    = sparse( nndof , 1 );      % Create the global force vector
  force1   = sparse( nndof , 1 );      % Create the global force vector
  reaction = sparse( nndof , 1 );      % Create the global reaction vector
  u        = sparse( nndof , 1 );      % Nodal variables

% Material properties (Constant over the domain)
  D_matb = young*inertia;
  D_mats = young/(2*(1+poiss))*area*5/6;
  reduce = 1;           % 1 for reduced integreation for shear matrix

  ttim = timing('Time needed to  set initial values',ttim); %Reporting time

% Element cycle
  for ielem = 1 : nelem

    lnods(1:nnode) = elements(ielem,1:nnode);

    coor_x(1:nnode) = coordinates(lnods(1:nnode),1);   % Elem. X coordinate
```

```
    len = coor_x(2) - coor_x(1);  % x_j > x_i

    const = D_matb/len;

    K_b = [ 0 ,  0 ,  0 ,  0 ;
            0 ,  1 ,  0 , -1 ;
            0 ,  0 ,  0 ,  0 ;
            0 , -1 ,  0 ,  1 ];

    K_b = K_b * const;

    const = D_mats/len;

    switch reduce

        case 0
            K_s = [   1   ,   len/2 ,   -1   ,   len/2 ;
                    len/2 , len^2/3 , -len/2 , len^2/6 ;
                     -1   ,  -len/2 ,    1   ,  -len/2 ;
                    len/2 , len^2/6 , -len/2 ,  len^2/3 ];

        case 1
            K_s = [   1   ,   len/2 ,   -1   ,   len/2 ;
                len/2 , len^2/4 , -len/2 , len^2/4 ;
                 -1   ,  -len/2 ,    1   ,  -len/2 ;
                len/2 , len^2/4 , -len/2 ,  len^2/4 ];
    end

    K_s = K_s * const;

    K_elem = K_b + K_s;

    f       = (-denss*area + uniload(ielem))*len/2;
    ElemFor = [ f, 0, f, 0];

% Find the equation number list for the i-th element
    for i = 1 : nnode
      ii = (i-1)*dofpn;
      for j = 1 : dofpn
        eqnum(ii+j) = (lnods(i)-1)*dofpn + j;   % Build the eq. number list
      end
    end

% Assemble the force vector and the stiffness matrix
    for i = 1 : dofpe
      ipos = eqnum(i);
      force(ipos) = force(ipos) + ElemFor(i);
      for j = 1 : dofpe
        jpos = eqnum(j);
        StifMat(ipos,jpos) = StifMat(ipos,jpos) + K_elem(i,j);
      end
    end

  end  % End element cycle

  ttim = timing('Time to assemble the global system',ttim); %Reporting time

% Add point load conditions to the force vector
  for i = 1 : size(pointload,1)
    ieqn = (pointload(i,1)-1)*dofpn + pointload(i,2);   % Find eq. number
```

```
    force(ieqn) = force(ieqn) + pointload(i,3);          % and add the force
  end

  ttim = timing('Time for apply side and point load',ttim); %Reporting time

% Apply the Dirichlet conditions and adjust the right hand side
  for i = 1 : size(fixnodes,1)
    ieqn = (fixnodes(i,1)-1)*dofpn + fixnodes(i,2);  % Find equation number
    u(ieqn) = fixnodes(i,3);                    % and store the solution in u
    fix(i) = ieqn;                           % and mark the eq. as a fix value
  end

  force1 = force - StifMat * u;       % Adjust the rhs with the known values

% Compute the solution by solving StifMat * u = force for the remaining
% unknown values of u
  FreeNodes = setdiff( 1:nndof , fix );            % Find the free node list
                                                   % and solve for it
  u(FreeNodes) = StifMat(FreeNodes,FreeNodes) \ force1(FreeNodes);

  ttim = timing('Time to solve the stiffness matrix',ttim); %Reporting time

% Compute the reactions on the fixed nodes as R = StifMat * u - F
  reaction(fix) = StifMat(fix,1:nndof) * u(1:nndof) - force(fix);

  ttim = timing('Time to solve the  nodal reactions',ttim); %Reporting time

% Compute the stresses
  Strnod = Stress_Beam_Timoshenko_v1_3(D_matb,D_mats,u);

  ttim = timing('Time to  solve the  nodal stresses',ttim); %Reporting time

% Graphic representation
  ToGiD_Beam_Timoshenko_v1_3(file_name,u,reaction,Strnod);

  ttim = timing('Time  used to write  the  solution',ttim); %Reporting time
  itim = toc;                                               %Close last tic
  fprintf(1,'\nTotal running time %12.6f \n\n',ttim); %Reporting final time
```

A.2 Input Data File

```
%
% Material Properties
%
 young  = 21000 ;
 poiss  = 0.25 ;
 a = 0.4;
 area   = a^2;
 inertia= a^4/12 ;
 denss  = 0.000000000 ;
%
% Coordinates
%
len = 4;      % total beam length
n = 64;      % number of elements
l = len/n;    % element length

global coordinates
```

```
coordinates = [];
for i = 0:n
    coordinates(i+1,1) = i*l;
end


%
% Elements
%
global elements
elements = [];
for i = 1:n
    elements(i,1) = i;
    elements(i,2) = i+1;
end

%
% Fixed Nodes
%
fixnodes  = [
        1, 1, 0.0000;
        65, 1, 0];
%
% Fixed Rotations
%
fixrotx  = [];
%
% Point loads
%
pointload = [ ];
%
uniload = sparse(n,1);
uniload(:,1) = -1;
```