Universitat Politècnica de Catalunya
Numerical Methods in Engineering
Computational Structural Mechanics and Dynamics

Beams

Assignment 6

Eduard Gómez
March 25, 2020

Contents

1 Assignment 6.1 **1**
1.1 Statement . . . 1
1.2 Solution . . . 1

2 Assignment 6.2 **2**
2.1 Statement . . . 2
2.2 Solution . . . 2

A Appendix **4**
A.1 Reduced integration Timoshenko . . . 4
A.2 Main file . . . 6

1 Assignment 6.1

1.1 Statement

Program In Mat Lab the Timoshenko 2 Nodes Beam element with reduce integration for the shear stiffness matrix.

1.2 Solution

The only change needed was changing matrix K_s so that it became:

$$\begin{bmatrix} 1 & \frac{L}{2} & -1 & \frac{L}{2} \\ & \frac{L^2}{4} & -\frac{L}{2} & \frac{L^2}{4} \\ & & -1 & -\frac{L}{2} \\ \text{sym} & & & \frac{L^2}{4} \end{bmatrix} \tag{1}$$

Other than this, I changed the code to automatize the plots shown next section. The changes were mainly moving the data entry outside of the files. Appendix A.1 shows the modified file. Appendix A.2 shows the main from which it was called.

This main file calls the Bernulli, Timoshenko and Timoshenko reduced in order to make the plots. These three files have therefore been modified so as not to export to gid, not repeat the data entry and the timing has been removed.

2 Assignment 6.2

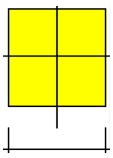
2.1 Statement

Solve the following problem with a 64 element mesh with the

- 2 nodes Euler Bernulli element
- 2 nodes Timoshenko Full Integrate element
- 2 nodes Timoshenko Reduce Integration element.

Compare maximum displacements, moments and shear for the 3 elements against the a/L relationship

$$E = 21000 \qquad \nu = 0.25$$
$$P = 1.0 \qquad L = 4$$



2.2 Solution

The goal of this section was to compare the results of the three methods when varying the aspect ratio of the beam. Here are the results of altering it:

a	Aspect ratio (a/L)	Area	Inertia
0.001	0.0003	1.0×10^{-6}	8.33×10^{-14}
0.005	0.001	2.5×10^{-5}	5.21×10^{-11}
0.01	0.003	1.0×10^{-4}	8.33×10^{-10}
0.02	0.005	4.0×10^{-4}	1.33×10^{-8}
0.05	0.01	2.5×10^{-3}	5.21×10^{-7}
0.1	0.03	1.0×10^{-2}	8.33×10^{-6}
0.2	0.05	4.0×10^{-2}	1.33×10^{-4}
0.4	0.1	1.6×10^{-1}	2.13×10^{-3}

Since the data had no specified dimensions, these results are also of unspecified units, however any consistent system of units would work.

Figure 1 shows the results in log-log plots. The x-axis shows the aspect ratio and the y axis is specified in each plot. The first thing to notice is that for displacement and bending moment, the three methods converge as the aspect ratio grows larger. In the same two subplots we see how Euler-Bernoulli and Timoshenko with reduced integration output very similar results whereas standard Timoshenko gives smaller values.

The shear subplot, however, shows that in this regard both variants of Timoshenko are very similar whereas Euler-Bernoulli differs. Furthermore, Euler-Bernoulli shows a total independence from aspect ratio.

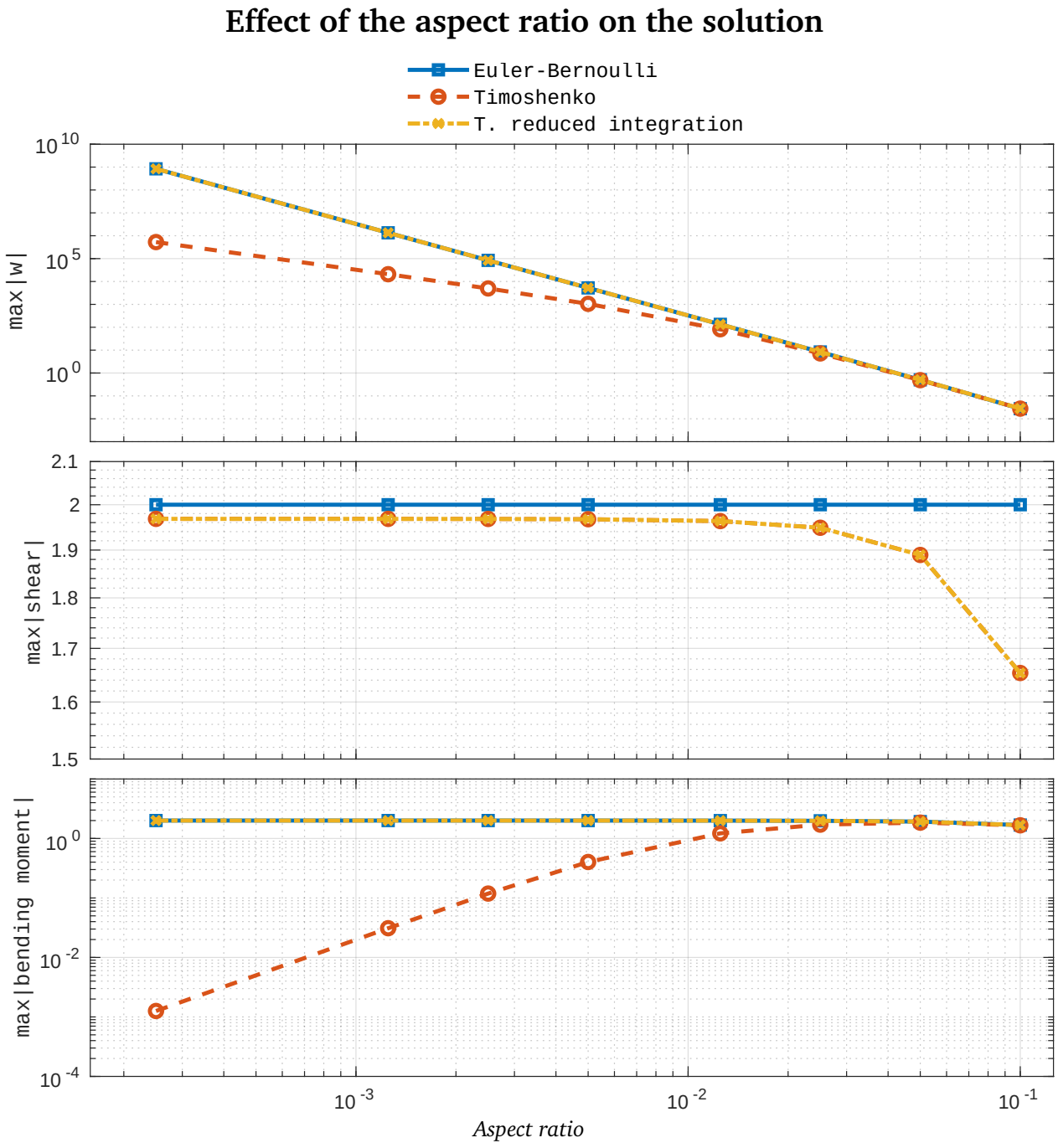


Figure 1: Effect of the aspect ratio on displacement, shear and bending moments.

A Appendix

A.1 Reduced integration Timoshenko

```
% Finds basics 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

% Dimension the global matrices.
StifMat  = sparse ( nndof , nndof ); % Create the global stiffness matrix
force    = 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;

%  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;

    %%%% Only change
    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 ];
    %%%%
    K_s = K_s * const;
    K_elem = K_b + K_s;

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

```
45
46      % Finds the equation number list for the i-th element
47      for i=1:nnode
48          ii = (i-1)*dofpn;
49          for j =1:dofpn
50              eqnum(ii+j) = (lnods(i)-1)*dofpn+j; % Build the equation number list
51          end
52      end
53
54      % Assemble the force vector and the stiffness matrix
55      for i = 1 : dofpe
56          ipos = eqnum(i);
57          force (ipos) = force(ipos)  + ElemFor(i);
58          for j = 1 : dofpe
59              jpos = eqnum(j);
60              StifMat (ipos,jpos) = StifMat (ipos,jpos) + K_elem(i,j);
61          end
62      end
63
64  end  % End element cycle
65
66  % Add point loads conditions to the force vector
67  for i = 1 : size(pointload,1)
68      ieqn = (pointload(i,1)-1)*dofpn+pointload(i,2);      % Finds eq. number
69      force(ieqn) = force(ieqn) + pointload(i,3);          % add the force
70  end
71
72  % Applies the Dirichlet conditions and adjust the right hand side.
73
74  for i = 1 : size(fixnodes,1)
75      ieqn = (fixnodes(i,1)-1)*dofpn+fixnodes(i,2);       % Finds eq. number
76      u (ieqn) = fixnodes(i,3);                % and store the solution in u
77      fix(i) = ieqn;                         % and mark the eq as a fix value
78  end
79  force  = force  - StifMat  * u;    % adjust the rhs with the known values
80
81  %  Compute the solution by solving StifMat * u = force for the
82  %  remaining unknown values of u.
83  FreeNodes = setdiff ( 1:nndof, fix );
84
85  u(FreeNodes) = StifMat(FreeNodes,FreeNodes) \ force(FreeNodes);
86
87  %  Compute the reactions on the fixed nodes as a R = StifMat * u - F
88  reaction(fix) = StifMat(fix,1:nndof) * u(1:nndof) - force(fix);
89
90  % Compute the stresses
91  Strnod = Stress_Beam_Timoshenko_RI(D_matb,D_mats,u);
```

A.2 Main file

```
%% Data entry

% Material Properties
young = 47619.04762 ;
poiss = 0.058201058 ;
denss = 1.000000000 ;
a_array = [0.001
           0.005
           0.010
           0.020
           0.050
           0.100
           0.200
           0.400]';

% Coordinates
global coordinates
n = 64;
length = 4;
coordinates = linspace(0,length,n)';

% Elements
global elements
elements = [(1:n-1)', (2:n)'];

% Fixed Nodes
fixnodes = [ 1, 1, 0.0000;
             n, 1, 0.0000];
% Loads
pointload = [ ];
uniload = ones(size(elements,1));

%% Initializing plot arrays
for i = 1:3
    max_d{i} = [];
    max_moment{i} = [];
    max_shear{i} = [];
end

%% Solving
for a = a_array
    area  = a*a;
    inertia  = a^4 / 12;

    % Printing LaTeX table
    line_to_print = sprintf('$%6.g$\t&\t$%6.g$\t&\t$%6.1e$\t&\t$%8.2e$\t\\\\'...
```

```
                                                                ,a,a/length,area,inertia);
    line_to_print = replace(line_to_print,'e','\times 10 ^{_');
    k = strfind(line_to_print,'_');

    for i = size(k,2):-1:1
        j = k(i);
        num = line_to_print(j+1:j+3);
        if num(2) == '0'
            num(2) = ' ';
        end
        line_to_print(j:j+3) = [num,'}'];
    end
    disp(line_to_print);

    % Euler
    run Beam_EulerBernoulli_v1_2.m
    max_d{1}(end+1) = max(abs(u(1:2:end)));
    max_moment{1}(end+1) = max(abs(Strnod(:,1)));
    max_shear{1}(end+1) = max(abs(Strnod(:,2)));


    % Timoshenko
    run Beam_Timoshenko_v1_2.m
    max_d{2}(end+1) = max(abs(u(1:2:end)));
    max_moment{2}(end+1) = max(abs(Strnod(:,1)));
    max_shear{2}(end+1) = max(abs(Strnod(:,2)));

    % Timoshenko Reduced integration
    run Beam_Timoshenko_RI.m
    max_d{3}(end+1) = max(abs(u(1:2:end)));
    max_moment{3}(end+1) = max(abs(Strnod(:,1)));
    max_shear{3}(end+1) = max(abs(Strnod(:,2)));
end

%% Plots
style = {'s-','o--','x-.'};
axis_x_limits = [10^-3.8 10^-0.9];
font = 'DejaVu Serif Condensed';

%% Displacement
figure('Position', [10 10 800 900])
subplot(311);
for i=1:3
    loglog(a_array./length,max_d{i},style{i},'LineWidth',2);
    hold on
end
ylabel('max|w|','FontName',font );
grid on
set(gca,'xticklabel',[])
```

```
axis([axis_x_limits, 10^-3, 1e3]);

%% Shear
subplot(312);
for i=1:3
    loglog(a_array./length,max_shear{i},style{i},'LineWidth',2);
    hold on
end
hold off
set(gca,'xticklabel',[])
ylabel('max|shear|','FontName',font );
grid on
axis([axis_x_limits, 1e-6 10]);
pos = get(gca, 'Position');
pos(2) = pos(2) + 0.07;
set(gca, 'Position', pos);

%% Moment
subplot(313);
for i=1:3
    loglog(a_array./length,max_moment{i},style{i},'LineWidth',2);
    hold on
end
pos = get(gca, 'Position');
pos(2) = pos(2) + 0.14;
set(gca, 'Position', pos);
lg = legend('Euler-Bernoulli','Timoshenko','T. reduced integration','Location','S');
lg.FontSize = 10;
lg.FontName = font;
set(lg,'Box','off')
ylabel('max|bending moment|','FontName',font);
axis([axis_x_limits, 1e-10  10^0.5]);
grid on
```