Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Nombre del estudiante: Rosa Eva González

Materia: Computational Structural Mechanics and Dynamics

Fecha de entrega: 19/03/2018

Descripción: Deber 6

a) Program In MatLab the Timoshenko 2 Nodes Beam element with reduce integration for the shear stiffness matrix

$$\mathbf{K}_b^{(e)} = \left(\frac{EI}{l}\right)^{(e)} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \text{(The point interpolation is exact for } \mathbf{K}_b^{(e)})$$

$$\mathbf{K}_s^{(e)} = \left(\frac{GA^*}{l}\right)^{(e)} \begin{bmatrix} 1 & \frac{l^{(e)}}{2} & -1 & \frac{l^{(e)}}{2} \\ \cdots & \frac{(l^{(e)})^2}{4} & -\frac{l^{(e)}}{2} & \frac{(l^{(e)})^2}{4} \\ & \cdots & 1 & -\frac{l^{(e)}}{2} \\ \text{Simetr.} & & \cdots & \frac{(l^{(e)})^2}{4} \end{bmatrix} \text{(Reduced integration)}$$

El codigo modificado del archivo de Matlab Beam_Timoshenko, se muestra al final del documento, los cambios realizados son los siguientes:

- Matriz de rigidez de corte modificada por integración reducida:

```
%%%MATRIZ MODIFICADA POR K REDUCTION INTEGRATION

K_shear = [   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 ];
```

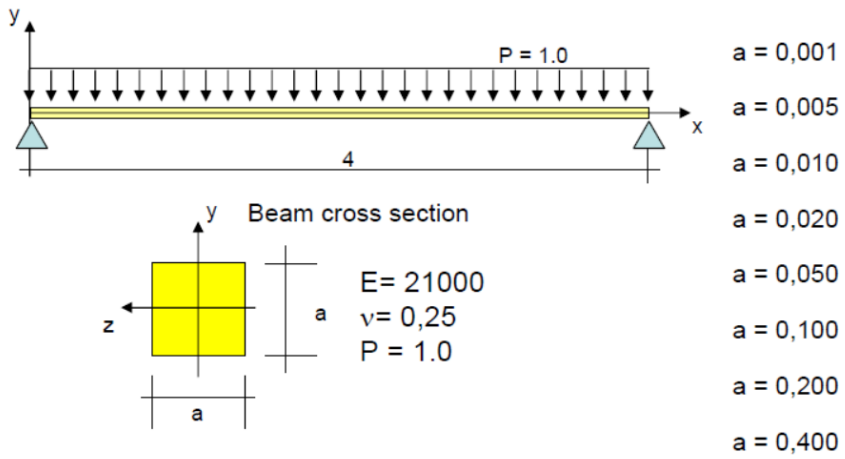
- Puntos de integración de Gauss:

```
gaus1 =0;
gaus2 = 0;
```

b) 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

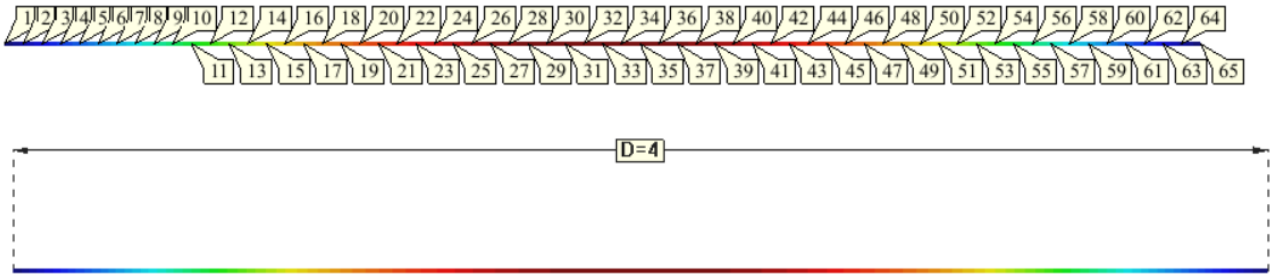


Código de entrada:
Se modifico a, para cada caso.

```
a=0.001;

young  = 21000;
 poiss  = 0.25;
 area   = a*a;
 inercia= a^4/12 ;
 denss  = 1.000000000 ;

% Coordinates se muestra solo el primero y el final, hay
coordenadas cada 0.0625, para tener 65 nodos.
%
global coordinates
coordinates = [
      0.0      ;…
   4.0000 ];

%
% Elements se muestra solo el primero y el final, el código
completo tiene 64 elementos
%
global elements
elements = [
    1,   2;…
   64,  65];

global fixdesp
%
```

```
% Fixed Nodes condiciones de apoyo, en nudos inicial y final,
restringiéndolo en el eje y.
%
fixdesp  = [
        1, 1, 0.0000;
           65, 1, 0.00000];

%
% Fixed Rotations
%
fixrotx  = [];
%
% Point loads
%
pointload = [ ];
%
```

Resultados en Gid:



En los diferentes tipos de análisis depende de la relación a/L, para los resultados obtenidos, en el caso del Momento flector y el esfuerzo cortante no varía para el caso de análisis por Timochenko con integración reducida, y únicamente el momento flector para de análisis de Euler Bernoulli.

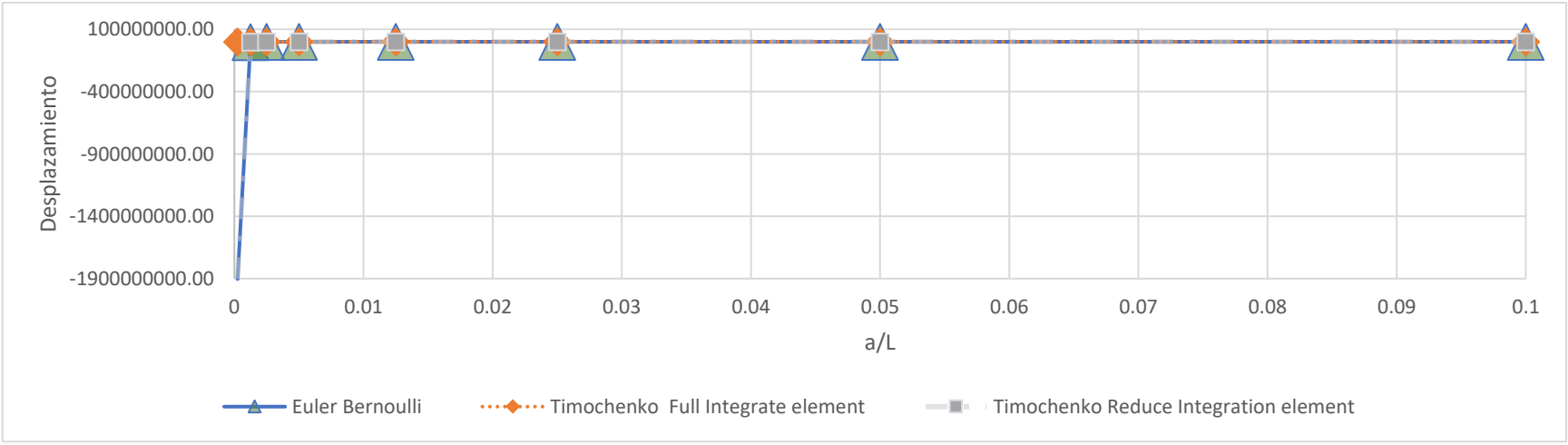
Los valores de desplazamiento son iguales para los análisis de Euler Bernulli, y Timochenko con integración reducida, y cuando tienen una relación a/L mayor a 0.025 difieren muy levemente con los resultados del análisis de Timochenko con integración completa.

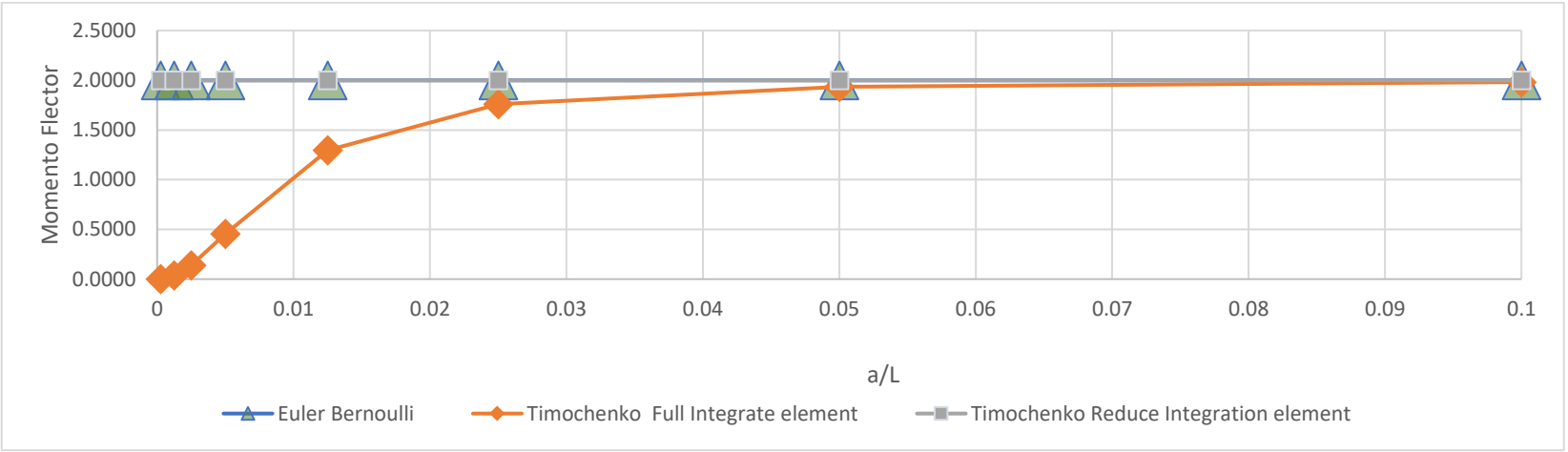
El valor de momento flector en el tercer caso de análisis tiende a igualarse a los otros dos tipos de análisis, cuando la relación a/L es mayor, e igualmente con el esfuerzo cortante pero este únicamente con el segundo tipo de análisis.

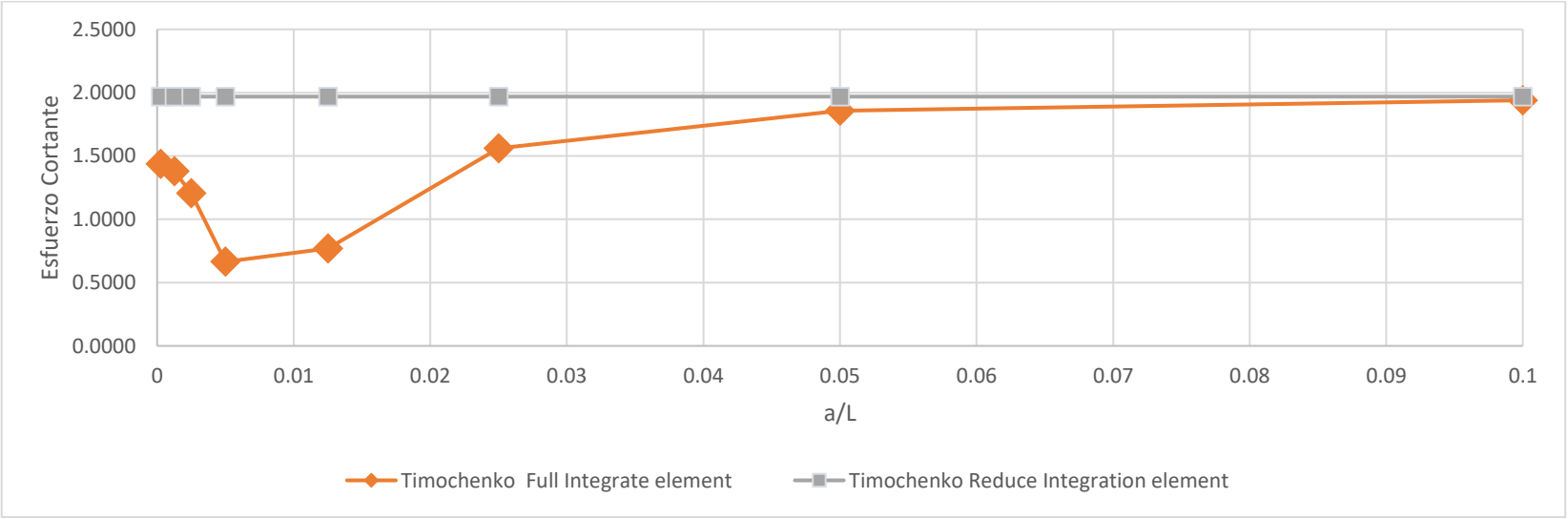
L	4

a	a/L	Desplazamiento			Momento Flector			Esfuerzo Cortante	
		1	2	3	1	2	3	2	3
0.001	0.00025	-1.90E+09	-1.40E+06	-1.90E+09	1.9999	0.0015	1.9990	1.4387	1.9688
0.005	0.00125	-3.05E+06	-5.51E+04	-3.05E+06	1.9999	0.0362	1.9990	1.3795	1.9688
0.01	0.0025	-1.90E+05	-1.31E+04	-1.90E+05	1.9999	0.1373	1.9990	1.2071	1.9688
0.02	0.005	-1.19E+04	-2.71E+03	-1.19E+04	1.9999	0.4553	1.9990	0.6646	1.9688
0.05	0.0125	-3.05E+02	-1.98E+02	-3.05E+02	1.9999	1.2959	1.9990	0.7694	1.9688
0.1	0.025	-1.90E+01	-1.68E+01	-1.91E+01	1.9999	1.7603	1.9990	1.5615	1.9688
0.2	0.05	-1.19E+00	-1.16E+00	-1.20E+00	1.9999	1.9335	1.9990	1.8569	1.9688
0.4	0.1	-7.44E-02	-7.55E-02	-7.61E-02	1.9999	1.9822	1.9990	1.9401	1.9688

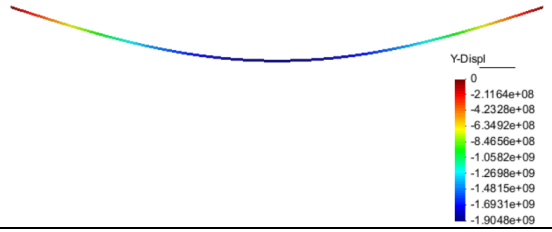
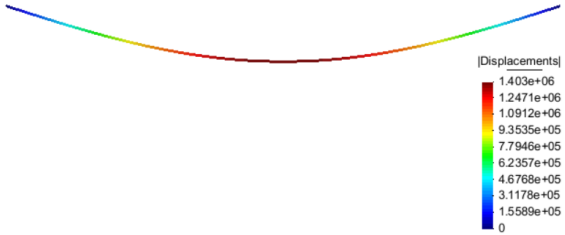
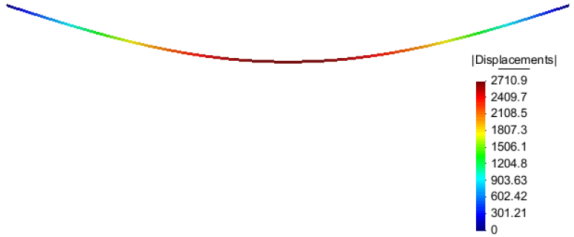
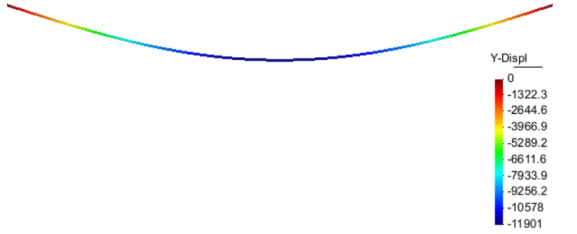
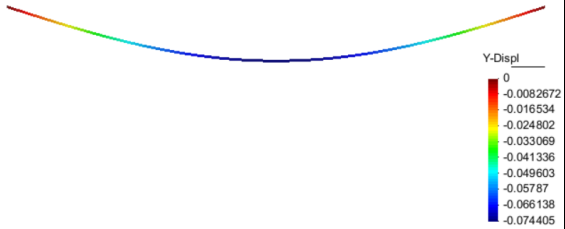
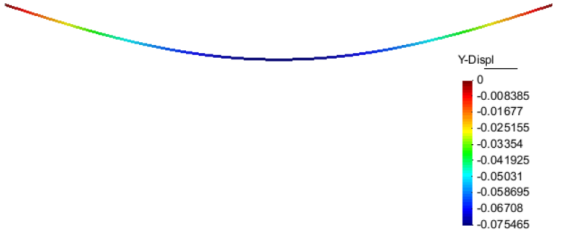
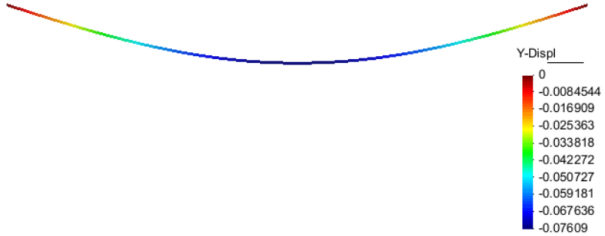
1	Euler Bernoulli
2	Timochenko Full Integrate element
3	Timochenko Reduce Integration element



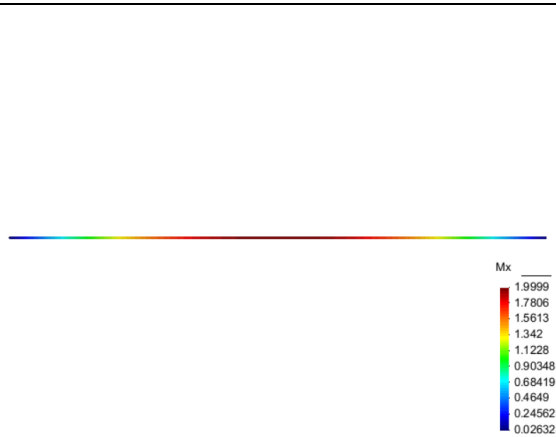
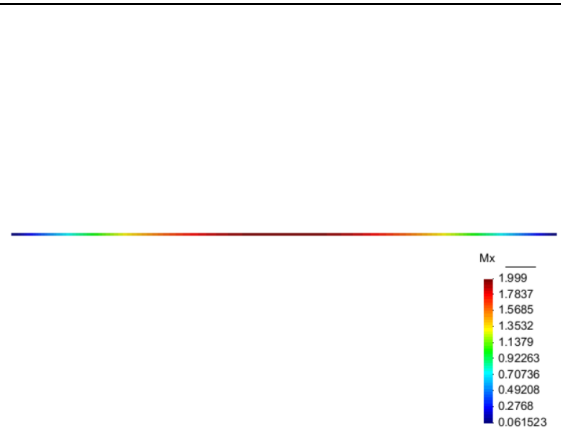

Momento Flector
2.5000
2.0000
1.5000
1.0000
0.5000
0.0000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
a/L
Euler Bernoulli
Timochenko Full Integrate element
Timochenko Reduce Integration element


Esfuerzo Cortante
2.5000
2.0000
1.5000
1.0000
0.5000
0.0000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
a/L
Timochenko Full Integrate element
Timochenko Reduce Integration element

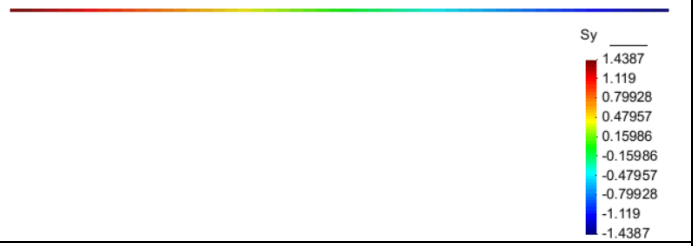
Desplazamiento:

a	1	2	3
0.001			
0.02			
0.4			

Momento Flector:

a	1	2	3
0.001			
0.02			
0.4			

Esfuerzo Cortante:

	2	3
0.001		
0.02		
0.4		

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%2 Nodes Beam using Timoshenko Theory%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%MODIFY Program In MatLab the Timoshenko 2
%Nodes Beam element with reduce integration for the shear stiffness matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Deber Rosa Gonzalez%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% Clear memory and variables.
  clear

% The variables are readed as a MAT-fem subroutine
% young =   Young Modulus+0
% poiss =   Poission Ratio
% thick =   thickness
% denss =   density
% coordinates = [ x , y ] coordinate matrix nnode x ndime (2)
% elements    = [ inode, jnode, knode ] element conectivities matrix
%               nelem x nnode; nnode = 3
% fixdesp     = [node number, dimension, fixed value] matrix with
%               dirichlet restrictions.
% pointload   = [node number, dimension, load value] matrix with
%               nodal loads.

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

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

% Finds basics dimentions
  npnod  = size(coordinates,1);         % Number of nodes
  nelem  = size(elements,1);            % Number of elements
  nnode  = size(elements,2);            % Number of nodes per element
  nndof  = npnod*2;                    % 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
  Str      = zeros  ( nelem , 3 );      % Create array for streses
  u        = zeros (nndof, 1);          % Nodal variables

%  Material properties (Constant over the domain).
  dmatf = young*inercia;
  dmats = young/(2*(1+poiss))*area*5/6;
%Se considera seccion rectangular A*

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

%  Element cycle.
  for ielem = 1 : nelem

    lnods_i = elements(ielem,1);
    lnods_j = elements(ielem,2);

    x_i = coordinates(lnods_i); % Elem. coordinates
    x_j = coordinates(lnods_j); % Elem. coordinates
    len = x_j - x_i;

    const = dmatf/len;

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

    K_flex  = K_flex * const;
```

```
    const =   dmats/len;

    %%%MATRIZ MODIFICADA POR K REDUCTION INTEGRATION

    K_shear = [   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_shear = K_shear * const;

    K_elem = K_flex + K_shear;


    f       = denss*len/2;

    ElemFor = [-f, 0,-f, 0];

% Finds the equation number list for the i-th element
    eqnum(1) = (lnods_i-1)*2+1 ;     % Build the equation number list
    eqnum(2) = (lnods_i-1)*2+2 ;
    eqnum(3) = (lnods_j-1)*2+1 ;
    eqnum(4) = (lnods_j-1)*2+2 ;

% Assamble the force vector and the stiffnes matrix
    for i = 1 : 4
      ipos = eqnum(i);
      force (ipos) = force(ipos)  + ElemFor(i);
      for j = 1 : 4
         jpos = eqnum(j);
         StifMat (ipos,jpos) = StifMat (ipos,jpos) + K_elem(i,j);
      end
    end

  end  % End element cicle

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

%  Add point loads conditions to the force vector
  for i = 1 : size(pointload,1)
    ieqn = (pointload(i,1)-1)*2+pointload(i,2);       % Finds eq. number
    force(ieqn) = force(ieqn) + pointload(i,3);       % add the force
  end

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

 %  Applies the Dirichlet conditions and adjust the right hand side.

  j = 0;
  for i = 1 : size(fixdesp,1)
    ieqn = (fixdesp(i,1)-1)*2+fixdesp(i,2);       % Finds eq. number
    u (ieqn) = fixdesp(i,3);              %and store the solution in u
    j = j + 1;
    fix(j) = ieqn;                        % and mark the eq as a fix value
  end

  force  = 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 ); % Finds the free node list and

  u(FreeNodes) = StifMat(FreeNodes,FreeNodes) \ force(FreeNodes);

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

 %  Compute the reactions on the fixed nodes as a 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

%  Element cycle.
  for ielem = 1 : nelem

    lnods_i = elements(ielem,1);
    lnods_j = elements(ielem,2);

    eqnum(1) = (lnods_i-1)*2+1 ;      % Build the equation number list
    eqnum(2) = (lnods_i-1)*2+2 ;
    eqnum(3) = (lnods_j-1)*2+1 ;
    eqnum(4) = (lnods_j-1)*2+2 ;

    u_elem(1:4)=u(eqnum(1:4));

    lnods_i = elements(ielem,1);
    lnods_j = elements(ielem,2);

    x_i = coordinates(lnods_i); % Elem. coordinates
    x_j = coordinates(lnods_j); % Elem. coordinates
    len = x_j - x_i;

    %%%recomendacion
    gaus1 =0;
    gaus2 = 0;

    bmat_f=[ 0, -1/len, 0, 1/len];

    bmat_s1=[-1/len,-(1-gaus1)/2, 1/len,-(1+gaus1)/2];
    bmat_s2=[-1/len,-(1-gaus2)/2, 1/len,-(1+gaus2)/2];

    Str(ielem,1) = dmatf*(bmat_f *transpose(u_elem));
    Str(ielem,2) = dmats*(bmat_s1*transpose(u_elem));
    Str(ielem,3) = dmats*(bmat_s2*transpose(u_elem));

  end

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

% Graphic representation.
ToGiD_VigaD (file_name,u,reaction,Str);

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