12

PROGRAMMING THE FEM FOR BEAM, PLATE AND SHELL ANALYSIS IN MAT-fem

written by Francisco Zárate[1]

12.1 INTRODUCTION

We present in this chapter the implementation of several of the elements for beam, plate and shell analysis studied in this book in the MAT-fem code environment written using the MATLAB® and GiD programming tools [On4]. MAT-fem includes several codes for FEM analysis of different structures. These codes can be freely downloaded from www.cimne.con/MAT-fem.

MATLAB® has been designed to work with matrices, facilitating the matrix algebra operations from the numerical and storage points of view, while providing a simple and easy way to handle complex routines.

Having an efficient analysis code is not the only requirement to work with the FEM. It is necessary to rely on a suitable interface to prepare the analysis data, to generate meshes in a simple and fast manner and to display the results so that their interpretation is clear. An ideal complement to MATLAB® for these purposes is the pre/postprocessor program GiD (www.gidhome.com and Appendix D of [On4]).

GiD allows users to treat any geometry via Computer Aided Design (CAD) tools and to easily assign to it the data needed for FE computations, i.e. material properties, boundary conditions, loads, etc. Different tasks, such as the mesh generation and data writing levels in a pre-defined format become a transparent task for the user with GiD.

The easy visualization of the analysis data and the numerical results with GiD allows users concentrating on their interpretation.

[1] Dr. F. Zárate can be contacted at zarate@cimne.upc.edu

E. Oñate, *Structural Analysis with the Finite Element Method. Linear Statics: Volume 2: Beams, Plates and Shells*, Lecture Notes on Numerical Methods in Engineering and Sciences, DOI 10.1007/978-1-4020-8743-1_12,

Structural type	Code name	GiD interface
Slender beams	Beam_EulerBernoulli	MAT-fem_Beams
Thick/slender beams	Beam_Timoshenko	MAT-fem_Beams
Thin plates	Plate_MZC	MAT-fem_Plates
Thick/thin plates	Plate_QLLL	MAT-fem_Plates
Thick/thin plates	Plate_Q4_Rect	MAT-fem_Plates
Thick/thin plates	Plate_Q4_Iso	MAT-fem_Plates
Thick/thin plates	Plate_TQQL	MAT-fem_Plates
3D shells with flat elements	Shell_QLLL	MAT-fem_Shells
Axisymmetric shells	Troncoconical_RM_Shell	MAT-fem_AxiShells

Table 12.1 Definition of the problem type, name of code and GiD interface

MAT-fem has been written thinking on the close interaction of GiD with MATLAB® for FEM analysis. GiD allows manipulating geometries and discretizations for writing the input data files required by MATLAB®. The calculation program is executed in MATLAB® without losing any of the MATLAB® advantages. Finally GiD gathers the output data files for graphical visualization and interpretation of results.

This scheme allows us understanding the development of a FEM program in detail, following each one of the code lines if desired, and making possible for users to solve examples that due to their dimensions would fall outside the capabilities of any program with educational aims.

We *have chosen to write a different MATLAB® code for the analysis of the different structures considered*, as shown in Table 12.1. The same programming strategy is followed and the variables have the same meaning in all cases. This simplifies the learning process and facilitates the modification of a specific code without introducing errors in the rest.

In the following sections the different MAT-fem codes for beam, plate and shell analysis are described in some detail. The description starts with the general input data file instructions, automatically generated by GiD, and follows with the information to understand the particular features of each code within MAT-fem.

Finally, the user interface implemented in GiD for each code is briefly described by means of examples of application.

12.2 MAT-fem

As already mentioned, MAT-fem contains several codes for analysis of beams, plates and shells (Table 12.1). All codes share the same programming philosophy and the differences are in the DOFs the constitutive

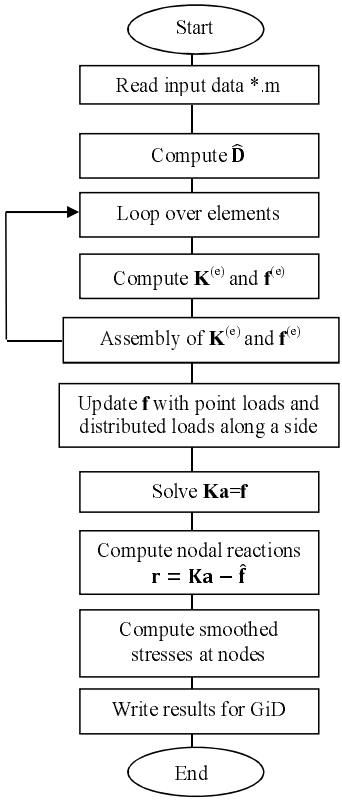


Fig. 12.1 MAT-fem flow chart for a typical problem type

matrix $\hat{\mathbf{D}}$, the generalized strain matrix $\mathbf{B}$ and the expressions for the element stiffness matrix and the equivalent nodal force vector.

MAT-fem is a top-down execution program. The program flow chart is shown in Figure 12.1. The input data module is implemented in the same file were the data is defined, as described in the next sections.

We consider that all elements have the same material properties. Hence, the constitutive matrix is evaluated outside the loop over the elements within which the element stiffness matrix and the equivalent nodal force vector are computed.

To save memory the stiffness matrix and the equivalent nodal force vector are assembled immediately after they are evaluated for each element.

Outside the element loop the equivalent nodal force vector is updated with the nodal point forces and the distributed loads acting along a side.

(a)

```
%
% Material Properties
% Beams
%
  young  =   4.761904762e+04 ;
  poiss  =   5.820105800e-02 ;
  denss  =   0.000000000e+00 ;
  area   =   1.600000000e-01 ;
  inertia=   2.133330000e-03 ;
```

(b)

```
%
% Material Properties
% Plates and Shells
%
  young  =   2.100000000e+11 ;
  poiss  =   3.000000000e-01 ;
  denss  =   0.000000000e+00 ;
  thick  =   1.000000000e-01 ;
```

Fig. 12.2 Input data file. Example of definition of material properties. a) Beams. b) Plates and shells

Once the unknown DOFs are found, the program evaluates the nodal reactions at the prescribed nodes and the smoothed stresses at the nodes. The final step is the writing of the numerical results to visualize them in GiD (Figure 12.1).

12.3 DATA FILES

Before executing MAT-fem it is necessary to feed it with information on the nodal coordinates, the mesh topology, the boundary conditions, the material properties and the loading.

The input data file uses MATLAB® syntax. The program variables are defined directly in that file. The name of the file will take the MATLAB® extension `.m`.

Inside the data file we distinguish three groups of variables: those associated to the material properties, those defining the topology of the mesh and those defining the boundary conditions.

12.3.1 Material data

With the intention of simplifying the code, an isotropic linear elastic material is used for all problems. Hence the material data appears only once in the data file.

Figure 12.2 shows the variables associated to the material data for each one of the structures considered. The definition of each variable is shown in Table 12.2.

We note that the program is free of data validation mechanisms. Hence it does not check up aspects such the Poisson's ratio rank ($0 <= \texttt{poiss} < 0.5$) and others. The reason is that these details, although they are important in practice, would hide the core of the FEM algorithm.

Variable	Description	Beams	Plates	3D Shells	Axisym. Shell
`young`	Young modulus	√	√	√	√
`poiss`	Poisson's ratio	√	√	√	√
`dens`	Density	√	√	√	√
`area`	Cross section area	√			
`inertia`	Cross section inertia moment	√			
`thick`	Thickness		√	√	√

Table 12.2 Material parameters for each structure

(a)

```
%
% Coordinates
%
global coordinates
coordinates = [
  0.00 ;
  0.50 ;

  2.00 ;
  2.50 ];
%
% Elements
%
elements = [
    1,   2 ;
    2,   3 ;

   17,  16 ;
   18,  17 ];
```

(b)

```
%
% Coordinates
%
global coordinates
coordinates = [
  0.00 , 0.00 ;
  0.50 , 0.00 ;

  2.00 , 2.50 ;
  2.50 , 2.50 ];
%
% Elements
%
elements = [
    1,   2,   5,   6 ;
    2,   3,   7,   5 ;

   17,  16,  20,  21 ;
   18,  17,  22,  20 ];
```

Fig. 12.3 Input data file. Topology definition. a) Beam. b) Plate

12.3.2 Mesh topology

The variable group that describes the mesh topology is defined with the attribute of a global variable that is accessible within the code by any subroutine. Figure 12.3 shows the definition of the coordinates and the nodal connectivities for a beam and a plate by means of the variables `coordinates` and `elements`.

`coordinates` is a matrix with as many rows as nodes in the mesh (`npnod`) and columns as the number of dimensions of the problem (1 for beams, 2 for plates and 3 for shells). For a beam the dimensions of the `coordinates` matrix are `npnod`×`1`. For a plate the coordinates x and y of each node are needed and `coordinates` has the dimensions `npnod`×`3`. For a shell, `coordinates` is a `npnod`×`3` matrix. The number of a node corresponds to the position that its coordinates have in the `coordinates` matrix, i.e. node number `25` has the position `25` in `coordinates`.

Structure type / Code name	Element		
	nnode = 2 	nnode = 3 	nnode = 4 
Slender beams **Beam_EulerBernoulli**	2-noded Euler Bernoulli beam element		
Thick/slender beams **Beam_Timoshenko**	2-noded Timoshenko beam element		
Thin plates **Plate_MZC**			MZC plate element
Thick/thin plates **Plate_QLLL** **Plate_Q4_Rect** **Plate_Q4_Iso** **Plate_TQQL**		TQQL	QLLL Q4 Rectangle Q4 (Isoparametric)
3D shells with flat elements **Shell_QLLL**		QLLL + Q4 (plane stress)	
Axisymmetric shells **Troncoconical_RM_Shell**	2-noded Reissner-Mindlin troncoconical element		

Table 12.3 Element type for each structure

The `elements` matrix defines the number of elements and their nodal connectivities. `elements` has as many rows as the number of elements in the mesh and columns as the number of nodes on each element (`nelem` × `nnode`). Table 12.3 shows the number of nodes for the elements considered in MAT-fem. The number of an element corresponds with the row number where its nodes are stored in `elements`.

12.3.3 Boundary conditions

The last group of variables defines the boundary conditions of the problem, as shown in Figure 12.4.

The `fixnodes` matrix defines the DOFs prescribed for the particular problem to be solved. The number of rows in `fixnodes` corresponds to the number of prescribed DOFs and the number of columns describes in the following order: the prescribed node number, the fixed DOF parameter (1 if the node is fixed in the x direction and 2 if it is fixed in the y direction, etc.) and the prescribed DOF value. Hence, if a node is prescribed in both the x and y directions two lines are necessary to define this condition.

Table 12.4 shows the parameters associated to each prescribed DOF for the different problems considered.

```
%
% Prescribed nodes
%
fixnodes = [
    1, 1, 0.0 ;
    1, 2, 0.0 ;

   13, 1, 0.0 ;
   13, 2, 0.0 ];
%
% Point loads
%
pointload = [
               6, 2, -1.0 ;

              18, 2, -1.0 ];
%
% Side loads
%
uniload = sparse ( 24,1 );
uniload ( 1 ) =  -1.0000e+00  ;
uniload ( 2 ) =  -1.0000e+00  ;
uniload ( 3 ) =  -1.0000e+00  ;
uniload ( 4 ) =  -1.0000e+00  ;
```

Fig. 12.4 Input data file. Boundary conditions definition

DOF	u	v	w	dw/dx	θ	θ_x	θ_y	$\theta_{x'}$	$\theta_{y'}$
Associated point force	F_x	F_y	F_z	M	M	M_x	M_y	$M_{x'}$	$M_{y'}$
`Slender beams`		1		2					
`Thick/slender beams`		1			2				
`Thin plates`			1			2	3		
`Thick/thin plates`			1			2	3		
`3D shells with flat elements`	1	2	3					4	5
`Axisymmetric shells`	1	2				3			

Table 12.4 Parameters for the prescribed DOF and the associated point forces

12.3.4 Point and surface loads

The `pointload` matrix defines the point loads acting at a node. This is a matrix where the number of rows is the number of point loads acting on the structure and each of the three columns defines the number of the loaded node, the associated DOF and the magnitude of the load (Figure 12.4). Displacement DOFs are associated to point loads, while rotations are associated to bending moments. Point loads are defined in the global coordinate system. If there are no point loads, `pointload` is defined as an empty matrix by means of the command `pointload = [];`

```
%% MAT-fem
%
% Clear memory and variables.
  clear

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

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

% Finds basic dimentions
  npnod  = size(coordinates,1);        % Number of nodes
  nelem  = size(elements,1);           % Number of elements
  nnode  = size(elements,2);           % Number of nodes per element
  dofpn  = 3;                         % 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);
```

Fig. 12.5 Program initialization and data reading

Finally, `uniload` is a sparse matrix that contains the information for uniformly distributed loads acting on the *normal direction* to each element. The distributed load is assumed to be constant for each element. Hence only the value of the load is needed for each element and the dimension of `uniload` is equal to `nelem`. If no uniform load acts, the `uniload` matrix remains empty with no memory consumption.

Figure 12.4 shows an example of the definition of uniform loads.

The name of the data file is up to the user. Nevertheless, the extension must be `.m` so that MATLAB® can recognize it.

12.4 START

MAT-fem begins by making all variables equal to zero with the `clear` command. Next it asks the user the name of the input data file that he/she will use (the `.m` extension in not included in the filename). Figure 12.5 shows the first lines of the code corresponding to the variables boot as well as the clock set up, which stores the total execution time in `ttim`.

Data reading, as previously said, is a direct variable allocation task in the program. From the data matrices it is possible to extract the basic dimensions of the problem, such as the number of nodal points, `npnod`, which corresponds to the number of lines in the `coordinates` matrix and the number of elements `nelem` which is equal to the number of lines in

```
% Dimension the global matrices.
  StifMat  = sparse ( nndof , nndof ); % Create the global stiffness matrix
  force    = sparse ( nndof , 1 );     % Create the global equivalent nodal force vector
  reaction = sparse ( nndof , 1 );     % Create the global reaction vector
  Str      = zeros  ( nelem , 2 );     % Create array for stresses
  u        = sparse (nndof, 1);        % Nodal variables
```

Fig. 12.6 Initialization of global stiffness matrix and equivalent nodal force vector

the `elements` matrix. The number of nodes for each element (Table 12.3) (`nnode`) is the number of columns in `elements`.

The total number of DOFs per element (`dofpe`) is equal to `nnode` multiplied by the number of DOFs for each node (`dofpn`).

Finally, the number of equations in the problem (`nndof`) is computed by multiplying the total number of nodes (`npnod`) by `dofpn`.

Note that these variables are defined in the data structure, which simplifies the code interpretation.

Throughout the program the `timing` routine is used to calculate the run time between two statements in the code. In this way the user can check the program modules that require higher computational effort. Inside `timing` the `tic` and `toc` MATLAB® commands are used.

12.5 STIFFNESS MATRIX AND EQUIVALENT NODAL FORCE VECTOR FOR SELF-WEIGHT AND DISTRIBUTED LOAD

12.5.1 Generalities

The code lines shown in Figure 12.6 define the global stiffness matrix (`stifMat`) and the equivalent nodal force (`force`) vector as a `sparse` matrix and vector, respectively. The reactions at the prescribed nodes are stored in `reaction`. Matrix `Str` and vector `u` are respectively used for storing the resultant stresses (at element level) and the nodal displacements. Table 12.5 shows the resultant stresses for each problem.

MAT-fem uses sparse matrices to optimize the memory using MATLAB® tools. In this manner and without additional effort, MAT-fem uses very powerful algorithms without losing its simplicity.

As the program's main purpose is to demonstrating the implementation of the FEM, some simplifications are made like using a single material for the whole structure. Consequently, the constitutive matrix does not vary between adjacent elements and it is evaluated before initiating the computation of the element stiffness matrix (Figure 12.1).

Structural type/ program name	N° of resultant stresses	Resultant stresses
Slender beams `Beam_EulerBernoulli`	1	M
Thick/slender beams `Beam_Timoshenko`	2	$M \quad Q$
Thin plates `Plate_MZC`	3	$M_x \quad M_y \quad M_{xy}$
Thick/thin plates `Plate_QLLL` `Plate_Q4_Rect` `Plate_Q4_Iso` `Plate_TQQL`	5	$M_x \quad M_y \quad M_{xy} \quad Q_x \quad Q_y$
3D shells with flat elements `Shell_QLLL`	6	$N_{x'} \quad M_{y'} \quad N_{x'y'} \quad M_{x'} \quad M_{y'} \quad M_{x'y'}$
Axisymmetric shells `Troncoconical_RM_Shell`	5	$N_{x'} \quad N_{y'} \quad M_{x'} \quad M_{y'} \quad Q_{z'}$

Table 12.5 Resultant stresses for each problem

The generalized constitutive matrix is typically split in the membrane (`D_matm`), bending (`D_matb`) and transverse shear (`D_mats`) components.

MAT-fem recalculates the values for each variable instead of storing them. The recalculation is performed in a fast manner and does not reduce significantly the code's efficiency. This leaves more memory for solving larger problems.

The definition of the Gauss point coordinates and weights is performed before entering the element loop for computing the element stiffness matrix $\mathbf{K}^{(e)}$ and the equivalent nodal force vector $\mathbf{f}^{(e)}$.

Figure 12.7 shows the element loop within which $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ are computed and assembled for uniformly distributed load and self-weight. The loop begins recovering the geometrical properties for each element. Vector `lnods` stores the nodal connectivities for the element. For 3D shells, the variables `coor_x`, `coor_y` and `coor_z` store the x, y, z coordinates for the nodes. For plates only the variables `coor_x` and `coor_y` are needed, while for beams just `coor_x` is used.

The next step is the computation of the element stiffness matrix. The same subroutine evaluates the equivalent nodal force vector for self-weight and uniformly distributed load for the element. The use of the same integration quadrature for integrating $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ allows us the organization

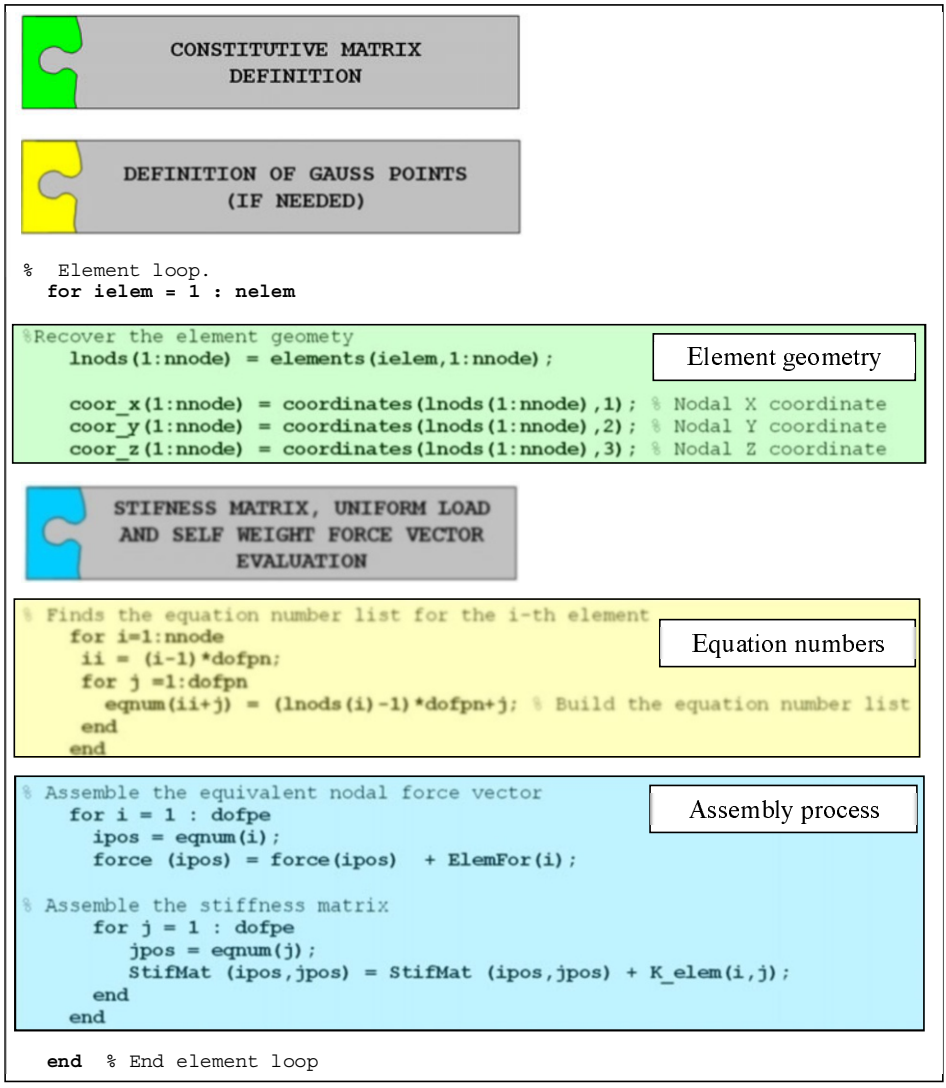


Fig. 12.7 Loop for computation and assembly of the stiffness matrix and the equivalent nodal force vector (uniform load and self-weight) for the element

of the code in this manner. The computation of $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ for each of the elements considered is detailed in the following sections.

Vector `eqnum` is defined before the assembly of the global equations. It contains the global number for each of the equations in the element stiffness matrix. This involves a loop over the number of nodes (`nnode`) and a second loop over the DOFs of each node (`dofpn`) (see Figure 12.7).

```
%  Add point loads to the equivalent nodal force vector
  for i = 1 : size(pointload,1)
    ieqn = (pointload(i,1)-1)* dofpn + pointload(i,2);   % Finds eq. number
    force(ieqn) = force(ieqn) + pointload(i,3);          % adds point load
  end
```

Fig. 12.8 Computation of the equivalent nodal force vector for point loads

```
%  Applies the Dirichlet conditions and adjusts the right-hand side.

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

  force = force - StifMat * u;       % adjust the rhs with the known values
```

Fig. 12.9 Update the equivalent nodal force vector for prescribed nodes

The assembly process is implemented by means of two loops from `1` to `dofpe` (number of equations for each element). In the first loop the equivalent nodal force vector is assembled and in the second one the element stiffness matrix is assembled. This scheme avoids storing the element matrices temporarily.

12.5.2 Point loads

Point loads acting at nodes (either forces or moments) are directly assembled in the global equivalent nodal force vector stored in the data file. This involves a loop over the number of loads contained in the `poinload` variable, finding the equations number associated to the load and adding the load value to the `force` vector (Figure 12.8).

12.6 PRESCRIBED DISPLACEMENTS

Figure 12.9 shows the loop over the prescribed DOFs and how their values, defined by the `fixnodes` matrix, are assigned to the nodal displacement vector `u`. Also the `fix` vector is defined to store the equation numbers for the prescribed DOFs.

Finally the `force` vector is updated with the product of the `StifMat` matrix and the `u` vector following the standard procedure [Chapter 1 of [On4]]. Vector `u` at this moment contains the values of the prescribed DOFs only.

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

Fig. 12.10 Solution of the equations system

```
%  Compute the reactions at fixed nodes as R = StifMat * u - F

  reaction(fix) = StifMat(fix,1:nndof) * u(1:nndof) - force(fix);
```

Fig. 12.11 Computation of nodal reactions

12.7 SOLUTION OF THE EQUATIONS SYSTEM

The strategy used in MAT-fem basically consists in solving the global equation system without considering those DOFs whose values are known (i.e. prescribed). The `FreeNodes` vector contains the list of the equations to be solved (Figure 12.10).

The `FreeNodes` vector is used as a DOF index and allows us to write the solution of the equations system in a simple way. MATLAB® takes care of choosing the most suitable algorithm to solve the system. The routines implemented in MATLAB® nowadays compete in speed and memory optimization with the best existing algorithms.

12.8 NODAL REACTIONS

The solution of the equations system is stored in the `u` vector containing the nodal displacements (Figure 12.11). Nodal reactions at the prescribed nodes are computed by means of the expression: `reaction = StifMat*u - force` [On4]. In order to avoid unnecessary calculations we use vector `fix` which contains the list of the equations associated to the prescribed DOFs as shown in Figure 12.11.

12.9 RESULTANT STRESSES

12.9.1 Generalities

Once the nodal displacements have been found it is possible to evaluate the resultant stresses in the elements by means of the $\hat{\mathtt{D}}\mathtt{Bu}$ expression. Since the generalized strain matrix `B` was previously computed at the integration

```
% Compute the stresses for QLLL plate element

  Strnod = Stress_Plate_QLLL(D_matb,D_mats,gauss_x,gauss_y,u);

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

Fig. 12.12 Call the subroutine for computing the resultant stresses for the 4-noded Reissner-Mindlin QLLL plate element

points, the resultant stresses are also computed at these points which are also optimal for evaluation of stresses (Section 6.7 of [On4]). The next step is to transfer the values of the stresses from the integration points to the element nodes. This step is treated in the next section. Figure 12.12 shows the call for the subroutine for computing the resultant stresses for the QLLL plate element which are stored in the `Strnod` matrix.

12.9.2 Computation of the stresses at the nodes

Every element has it own subroutine for computing the resultant stresses. Typically, this subroutine requires the generalized constitutive matrix, the coordinates of the integration points and the nodal displacements.

Figure 12.13 shows the general form of the subroutine for computing the resultant stresses. The resultant stresses are first computed at the Gauss integration points within each element and then they are extrapolated to the nodes following a particular stress projection and smoothing scheme. The resultant stress computation starts with the definition of the variables `nelem`, `nnode`, `npnod`, `dofpn` and `dofpe`, as it was done at the beginning of the program (Figure 12.5). This avoids the transfer of these variables as arguments when the subroutine is called and preserves the clarity in the code. Similarly, the `Strnod` and `eqnum` matrices are dimensioned. `Strnod` contains the smoothed resultant stresses at the nodes and an additional parameter that defines the number of elements that surround each node that is required to perform the smoothing.

Next a loop over all the elements is performed for recovering the nodal connectivities (`lnods`), the nodal coordinates and the equation number associated to each DOF of the element nodes (`eqnum`). These operations were already performed for computing the element stiffness matrices and they are repeated here in order to save storing the data. Repeating some computations is typically a more efficient and faster procedure than storing the previously computed values in memory.

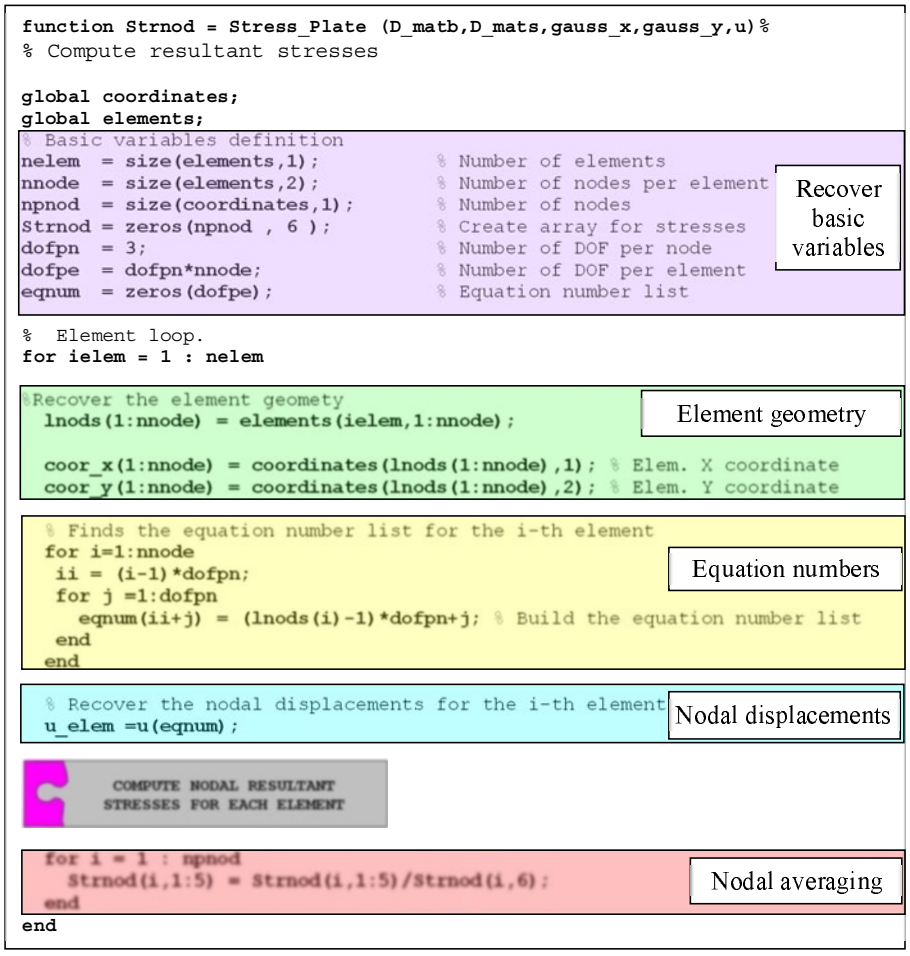


Fig. 12.13 Computation of resultant stresses

An important step is to identify the nodal displacements for the element. This is a simple task as the number of the equations associated to the element DOFs stored in `eqnum` allows us to recover the nodal displacement values in the `u_elem` vector.

The computation of the resultant stresses requires building the generalized strain matrix $\mathbf{B}$ at each Gauss point of the element. The resultant stresses are computed at each Gauss point as $\hat{\boldsymbol{\sigma}} = \hat{\mathbf{D}}\mathbf{B}\mathbf{a}^{(e)}$ and are stored in `Strnod`.

The next step is to extrapolate the resultant stresses from the Gauss points to the nodes.

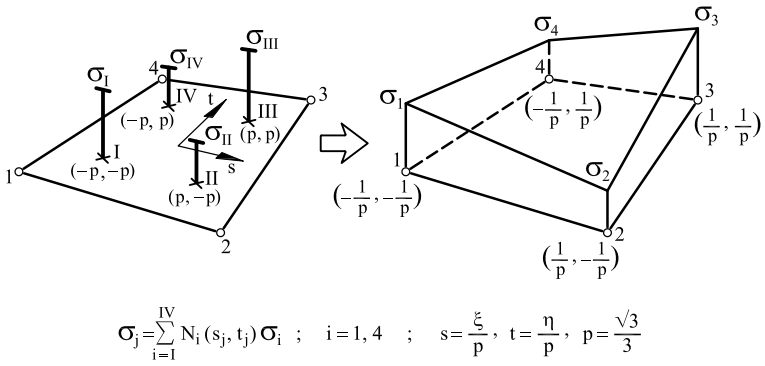


Fig. 12.14 Extrapolation of the Gauss point stresses to the nodes for a 4-noded quadrilateral

For the 3-noded triangle the resultant stresses are constant over the element and nodal extrapolation is trivial. This is not the case for 4-noded quadrilateral where the resultant stresses typically have a bilinear variation over the element and the extrapolation to the nodes is performed using the shape functions as explained in Section 9.8.2 of [On4].

For instance, for the 4-noded quadrilateral the nodal value of each resultant stress component σ is obtained as

$$\sigma_j = \sum_{i=I}^{IV} N_i(s_j, t_j)\sigma_i \qquad j = 1, 4 \tag{12.1}$$

where σ_j is the value of the stress at the jth node (j is the local number of the node), σ_i is the value of the stress component at each Gauss point and the coordinates s and t range from $1/p$ to $-1/p$ for the four element nodes as shown in Figure 12.14. For more details see [On4].

Once the resultant stresses have been extrapolated from the Gauss point to the nodes, an averaging of the nodal values contributed from each element sharing the node is performed to compute a smoothed resultant stress field at each node.

12.10 POSTPROCESSING STEP

Once the nodal displacements, the reactions and the resultant stresses have been calculated, their values are transferred to the postprocessing files from where GiD will be able to display them in graphical form. This is performed in the subroutine `ToGiD` (Figure 12.15).

```
% Graphic representation.
  ToGiD (file_name,u,reaction,Strnod);
```

Fig. 12.15 Call for the postprocessing step via GiD

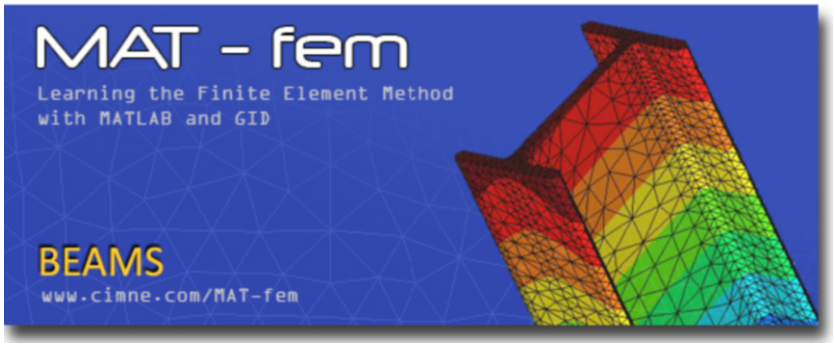


Fig. 12.16 GUI for MAT-fem-Beams

12.11 GRAPHICAL USER INTERFACE

12.11.1 Preprocessing

In this section the Graphical User Interface (GUI) implemented in GiD is described. In order to access the GUI it is necessary to select from the GiD's Data menu the adequate module for each of the MAT-fem codes. When selected, an image similar to that shown in Figure 12.16 appears.

All the GiD capabilities are part of MAT-fem. These include geometry generation, import and handling, as well as a variety of meshing, input data and results visualization techniques. All this provides MAT-fem with capacities difficult to surpass for an educational code.

There is plenty of information on GiD available in Internet. We recommend visiting the GiD web site at www.gidhome.com.

Solving a problem with MAT-fem is very simple once the geometry has been defined. Just follow the icons of the MAT-fem graphical menu that appears when MAT-fem is activated (Figure 12.17).

The first button  in Figure 12.17 works to identify the geometrical entities (point or lines) that have nodes with prescribed displacements. When pressing on, an emergent window will appear to select the points or lines where the displacements are prescribed (Figure 12.18) The check

Fig. 12.17 MAT-fem graphical menu

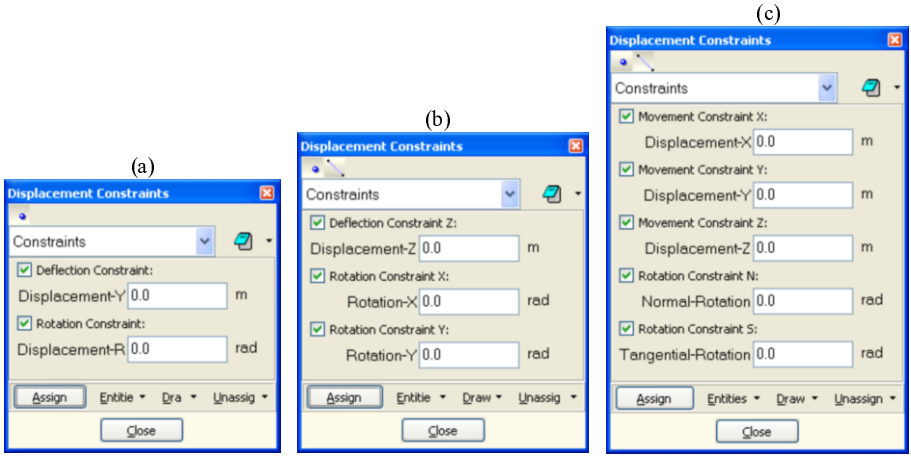


Fig. 12.18 Assign conditions at prescribed nodes. a) Beams. b) Plates. c) Shells

boxes identify the prescribed directions. Also it is possible to assign a non-zero value to the constraint.

The second button  in Figure 12.17 is used for point loads allocation. When selected, an emergent window (Figure 12.19) allows introducing the point load values in the global coordinate system. Then it is necessary to select the nodes were the point load is applied.

Point loads act normal to the beam axis and the plate surface, or in an arbitrary direction for a 3D shell. Point bending moments are defined as positive if they act in an anti-clockwise sense.

The third button  in Figure 12.17 is associated to uniformly distributed loads along the element sides and permits to assign this condition on geometry lines. The emergent window (Figure 12.20) allows introducing the value of the side load per unit length (or area). Uniform loads are assumed to act normal to the element surface (or the beam axis).

Material properties

The material properties are defined with the fourth button  in Figure 12.17. This leads to the emergent window shown in Figure 12.21 which allows users defining the material parameters associated to each structure

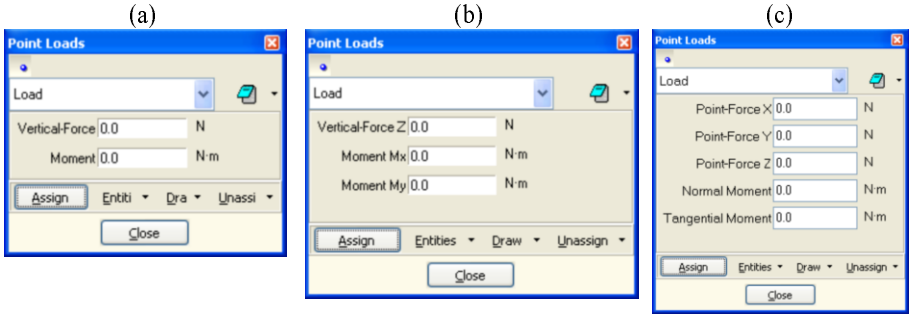


Fig. 12.19 Assign point load. a) Beams. b) Plates. c) Shells

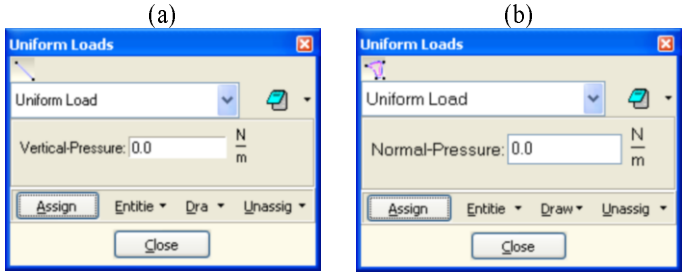


Fig. 12.20 Assign uniformly distributed load. a) Beams. b) Plates and shells

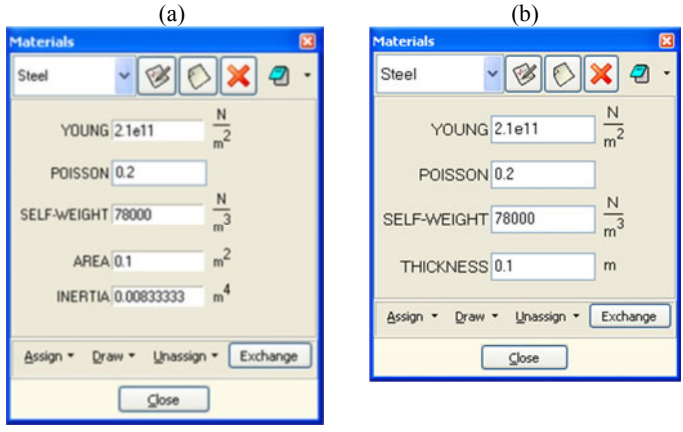


Fig. 12.21 Definition of material properties. a) Beams. b) Plates and shells

like the Young modulus, the Poisson's ratio, the density, the thickness, the transverse cross section, the inertia modulus, etc. It is necessary to assign these properties over the geometry entities that define the analysis domain (lines for beams and axisymmetric shells and surfaces for plates

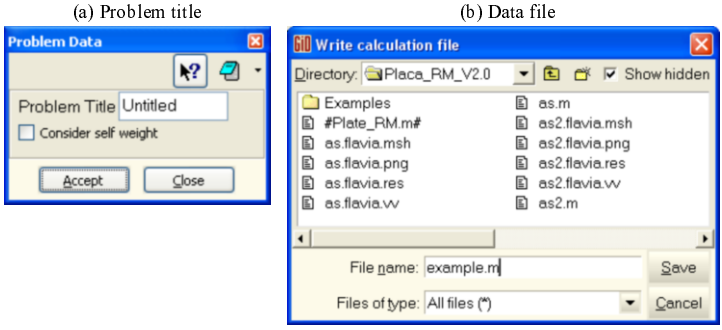


Fig. 12.22 (a) General title of the problem. (b) Writing of data file

and 3D shells). As mentioned earlier, only one type of material is allowed in MAT-fem for the sake of simplicity.

The general properties button (the fifth button of Figure 12.17) allows users to access the window shown in Figure 12.22a were the title of the problem is defined as well as the problem type (plane stress or plane strain) and the self-weight load option.

Once the boundary conditions and the material properties have been defined it is necessary to generate the mesh. The sixth button of Figure 12.17 is used to create the mesh with GiD.

The writing of the data file is made when pressing the last button of Figure 12.17. All the geometrical and material properties of the problem, as well as the boundary conditions and the loads are written on the data file in the specific reading format for MAT-fem. Recall that the file name needs the `.m` extension as shown in Figure 12.22b.

It is important to remark that the file extension `.m` can be set only when the box *Files of type* is set to *All files* (*) (Figure 12.22b).

12.11.2 Program execution

The problem calculation is performed with MATLAB® by using the appropriate code (i.e. Beam_EulerBernoulli, Plate_MZC, Shell_QLLL, etc.). The code execution does not have other complications than knowing the directory where the output file will be written. A good practice is to set this directory as the working directory were the postprocessing file will be also written.

During the code execution the total time used by the code will appear in the MATLAB® console as well as the time consumed in each subrou-

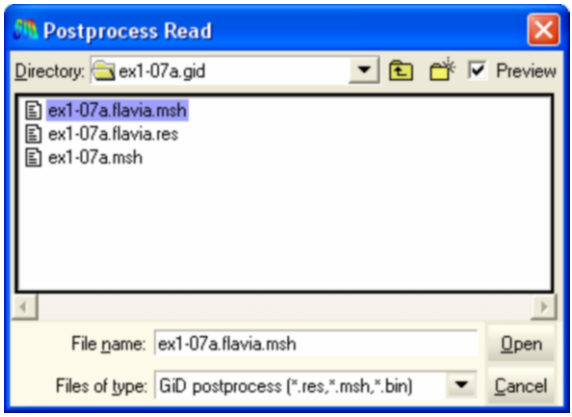


Fig. 12.23 Postprocessing file reading

tine. The largest time consumption in the academic problems solved with MAT-fem is invested in the calculation and assembly of the global stiffness matrix, while the solution of the equations system represents a small percentage of the consumed time. The opposite happens when solving larger structural problems.

Once the code execution is finished, the variables are still recorded inside MATLAB®, thus allowing users experiment with the collection of internal functions available.

12.11.3 Postprocessing

Once the problem execution in MATLAB® is concluded it is necessary to return to GiD for the postprocessing step in order to analyze the results. The next step is to open any of the generated files that contain the extension *.flavia.msh or *.flavia.res (Figure 12.23).

The results visualization step is performed using the GiD graphical possibilities which permit to visualize the results by means of iso-lines, cuts and graphs. This facilitates the interpretation of the MAT-fem results.

The `spy(StifMat)` command of MATLAB® displays the profile of the global stiffness matrix (see Figure 12.29b). Other MATLAB® commands allow users to find out the properties of this matrix, such as its rank, eigenvectors, determinant, etc.

In the following sections we describe the particular features of the different Mat-fem codes implemented.

```
%  Material properties (Constant over the domain).
  D_matb = young*inertia;

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

Fig. 12.24 Constitutive matrix for 2-noded Euler Bernoulli beam element

```
    len = coor_x(2) - coor_x(1);
    const = D_matb/len^3;

    K_elem = [ 12    , 6*len  , -12    ,  6*len  ;
                 6*len, 4*len^2,  -6*len,  2*len^2;
               -12    ,-6*len  ,  12    , -6*len  ;
                 6*len, 2*len^2,  -6*len,  4*len^2];

    K_elem = K_elem * const;

    f       = (-denss*area + uniload(ielem))*len/2;
    ElemFor = [ f, f*len/6, f,-f*len/6];
```

Fig. 12.25 Stiffness matrix and equivalent nodal force vector for 2-noded Euler-Bernoulli beam element

12.12 2-NODED EULER-BERNOUILLI BEAM ELEMENT

The formulation of this beam element can be found in Section 1.3.

We present next the parts of the Beam_EulerBernoulli code for computing the constitutive matrix, the stiffness matrix, the equivalent nodal force vector (for uniformly distributed load) and the bending moments. The rest of the code is identical to that explained in the previous section.

12.12.1 Stiffness matrix and equivalent nodal force vector

The constitutive matrix for the 2-noded Euler-Bernoulli beam element contains the bending stiffness (EI_y) only (Figure 12.24).

The element stiffness matrix is given explicitly in Eq.(1.20). The expression for the equivalent nodal force vector for a uniformly distributed loading is shown in Eq.(1.21b).

The computation of $\mathbf{K}^{(e)}$ (`K_elem`) and $\mathbf{f}^{(e)}$ (`ElemFor`) is shown in Figure 12.25.

12.12.2 Computation of bending moment

The bending moment within each element is first computed at the two Gauss points that integrate exactly the element stiffness matrix. The ben-

```
 % Two gauss point for bending moment evaluation
  gaus1 =-1/sqrt(3);  % Gauss Point 1
  gaus2 = 1/sqrt(3);  % Gauss Point 2

  a = (1+sqrt(3))/2;
  b = (1-sqrt(3))/2;

  len2 = len^2;

 % Bmat and bending moment at Gauss point 1
bmat_1=[6*gaus1/len2,(-1+3*gaus1)/len,-6*gaus1/len2,(1+3*gaus1)/len];
    Str_g1(ielem,1) = D_matb*(bmat_1*transpose(u_elem));

 % Bmat and bending moment at Gauss point 2
bmat_2=[6*gaus2/len2,(-1+3*gaus2)/len,-6*gaus2/len2,(1+3*gaus2)/len];
    Str_g2(ielem,2) = D_matb*(bmat_2*transpose(u_elem));%

    Strnod(lnods(1),1) = Strnod(lnods(1),1 +a*Str_g1+b*Str_g2;
    Strnod(lnods(2),1) = Strnod(lnods(2),1)+b*Str_g1+a*Str_g2;
    Strnod(lnods(1),2) = Strnod(lnods(1),2)+1;
    Strnod(lnods(2),2) = Strnod(lnods(2),2)+1;
```

Fig. 12.26 Computation of the bending moment at the two Gauss points for the 2-noded Euler-Bernoulli beam element

ding moment at each Gauss point is computed as

$$M_i = EI_y \mathbf{B}_i \mathbf{a}^{(e)} \quad , \quad i = 1, 2$$

where $\mathbf{B}_i^{(e)}$ is the curvature matrix of Eq.(1.16a) computed at the ith Gauss point. The bending moment at the Gauss points is stored in `Str_g1` and `Str_g2` for the subsequent nodal smoothing and visualization.

The computation of the bending moment is shown in Figure 12.26.

12.12.3 Example. Clamped slender cantilever beam under end point load

Figure 12.27 shows the beam geometry, the material properties and the load. The section is square and the beam slenderness ratio is $r = \frac{L}{h} = 100$.

The exact solution is a unit displacement at the free end and a bending moment of 50 N/m at the clamped end.

Figure 12.27 also shows the menus for defining the boundary conditions, the load and the material properties. The discretization and the writing of the input data file is carried out using the last two bottoms of the menu of Figure 12.17.

The problem has been solved with meshes of 2, 4, 8, 16, 32 and 64 2-noded Euler-Bernoulli beam elements. Figure 12.28 shows the numbering of elements and nodes for the 8 element mesh. This mesh is taken as the reference for the input data file also shown in Figure 12.28.

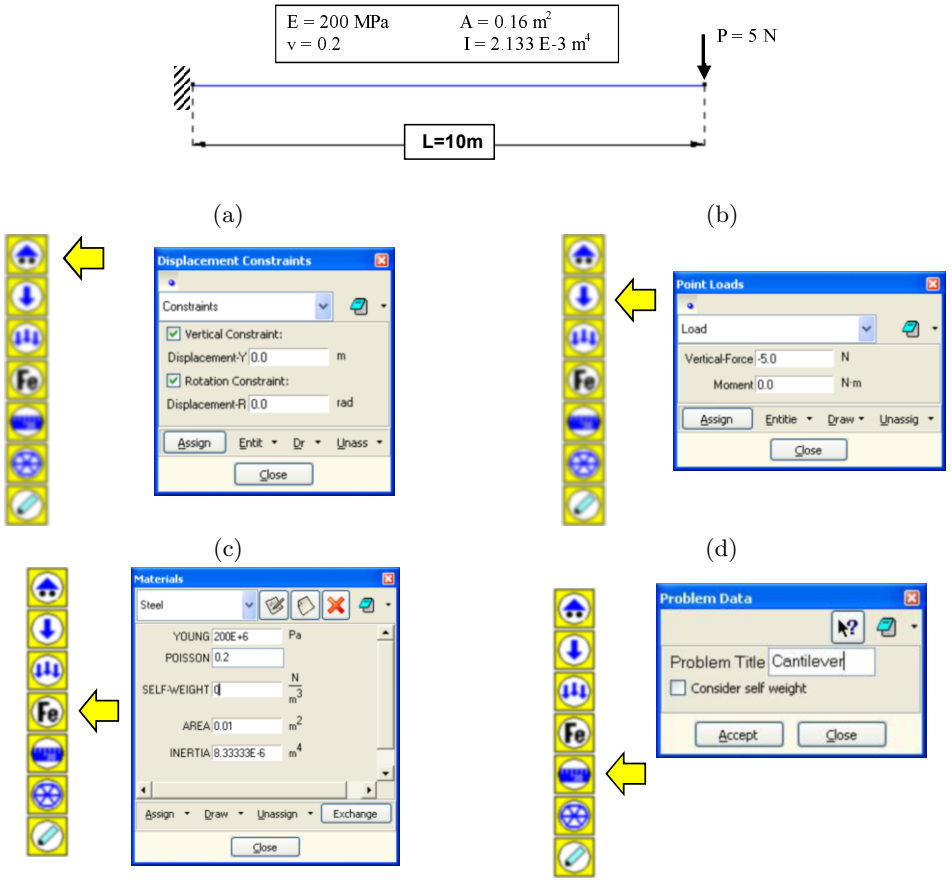


Fig. 12.27 Slender clamped cantilever beam ($r = 100$) under end point load. Definition of (a) boundary conditions, (b) loads, (c) materials and (d) problem characteristics

The code is executed within MATLAB® using the Beam_EulerBernoulli command, once the directory file where this code is located is selected. The total execution time for this problem (8 element mesh) is 0.010 sec.

Note that the assembly of the stiffness matrix takes most of the total execution time, as the internal indices of the sparse matrix need to be updated (Figure 12.29a).

Figure 12.29b shows the profile of the global stiffness matrix, as displayed by MATLAB®.

Figure 12.30 shows the deformed shape of the beam and the bending moment distribution. The *vertical displacement under the force is "exact"*

```
%========================================================================
% MAT-fem_Beams 1.0  - MAT-fem is a learning tool for undestanding
%                      the Finite Element Method with MATLAB and GiD
%========================================================================
% PROBLEM TITLE = Slender cantilever beam under end point load. Analysis
%                 with eight 2-noded Euler-Bernoulli beam elements
%  Material Properties
  young  =   2.000000000e+08 ;
  poiss  =   2.000000000e-01 ;
  denss  =   0.000000000e+00 ;
  area   =   1.000000000e-02 ;
  inertia=   8.333330000e-06 ;
% Coordinates
global coordinates
coordinates = [
   0.000000000e+00   ;
   1.250000000e+00   ;
   2.500000000e+00   ;
   3.750000000e+00   ;
   5.000000000e+00   ;
   6.250000000e+00   ;
   7.500000000e+00   ;
   8.750000000e+00   ;
   1.000000000e+01   ] ;
% Elements
global elements
elements = [
      1   ,      2   ;
      2   ,      3   ;
      3   ,      4   ;
      4   ,      5   ;
      5   ,      6   ;
      6   ,      7   ;
      7   ,      8   ;
      8   ,      9   ] ;
% Fixed Nodes
fixnodes = [
      1  , 1 ,   0.000000000e+00  ;
      1  , 2 ,   0.000000000e+00  ] ;
% Point loads
pointload = [
      9  , 1 ,  -5.000000000e+00  ;
      9  , 2 ,   0.000000000e+00  ] ;
% Distributed loads
uniload = sparse ( 8,1 );
```

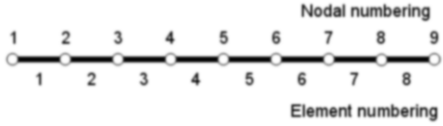


Fig. 12.28 Slender cantilever beam ($r = 100$) under end point load. Input data file for eight element mesh of 2-noded Euler-Bernoulli beam elements

for all the meshes considered. This is a particular feature of this problem, as explained in Section 1.3.4 (p. 19).

The bending moment at the clamped node is 50.0 Nxm. This value has been obtained from the reaction at that node which yields the exact solution for all meshes.

12.13 2-NODED TIMOSHENKO BEAM ELEMENT

The formulation of this beam element can be found in Section 2.3.

(a)

```
>> Beam_EulerBernoulli
Enter the file name:cant_pun_008
Time needed to read the input file     0.003935
Time needed to set initial values      0.000072
Time to assemble the global system     0.000774
Time for apply side and point loads    0.000033
Time to solve the stiffness equations  0.000262
Time to solve the nodal reactions      0.000045
Time to solve the nodal stresses       0.000267
Time used to write the solution        0.005371

Total running time                     0.010758
```

(b)

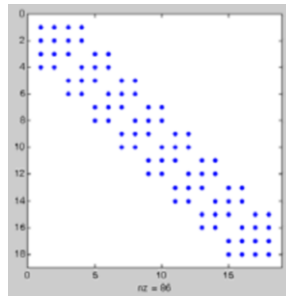


Fig. 12.29 Slender cantilever beam under end point load. (a) Execution time for eight element mesh. (b) Profile of stiffness matrix for the eight element mesh

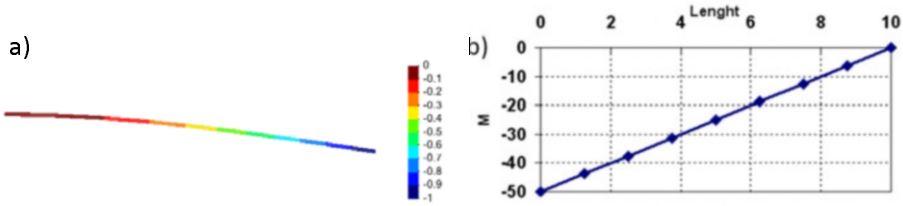


Fig. 12.30 Slender cantilever under end point load analized with eight 2-noded Euler-Bernoulli beam elements. a) Deformed shape. b) Moment distribution

```
%  Material properties (Constant over the domain).
  D_matb = young*inertia;
  D_mats = young/(2*(1+poiss))*area*5/6;

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

Fig. 12.31 2-noded Timoshenko beam element. Constitutive matrices

The structure of the code is very similar to that for the 2-noded Euler-Bernoulli beam element described in the previous section.

12.13.1 Stiffness matrix and equivalent nodal force vector

The constitutive matrix has been split in two parts: the bending term $\hat{D}_b$ (`D_matb`) and the transverse shear term $\hat{D}_s$ (`D_mats`). The shear correction factor has been taken equal to 5/6 (rectangular section). The computation of the constitutive matrices `D_matb` and `D_mats` are shown in Figure 12.31.

Figure 12.32 shows the explicit computation of the element stiffness matrix by sum of the bending and transverse shear contributions using a single integration point (see Eq.(2.25)).

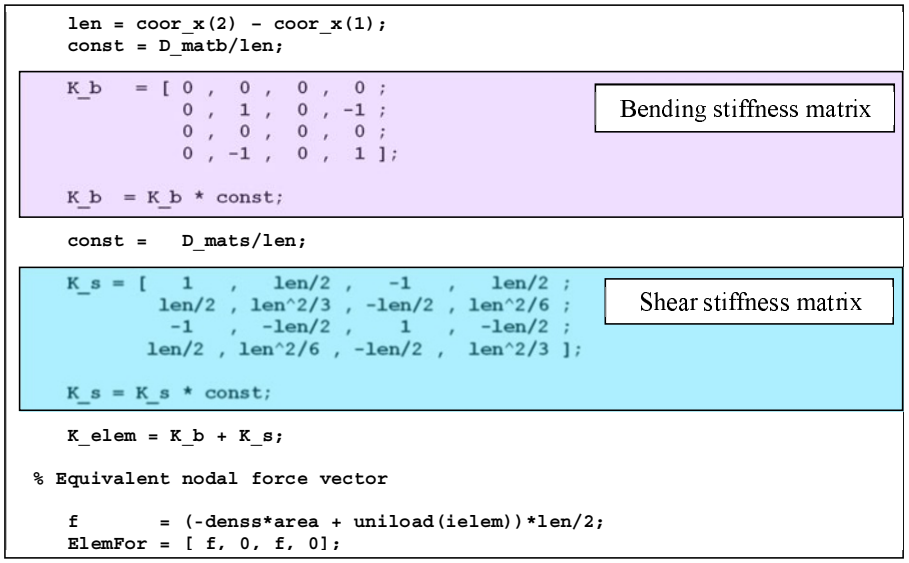


Fig. 12.32 2-noded Timoshenko beam element. Computation of stiffness matrix and equivalent nodal force vector for self-weight and uniform load

The last lines of Figure 12.32 show the computation of the equivalent nodal force vector for a uniform load (`uniload`) and self-weight (`denss*area`) using a single integration point.

12.13.2 Computation of bending moment and shear force

Figure 12.33 shows the computation of the bending moment M and the shear force Q at the element center via Eq.(2.17).

The bending moment and the shear force are stored in `Strnod` for the subsequent representation.

12.13.3 Example. Thick cantilever beam under end-point load

Figure 12.34 shows the beam geometry of the cantilever, the material properties and the load. The section is square and the beam slenderness ratio is $r = \frac{L}{h} = 25$.

Table 12.6 shows the convergence of the deflection under the load and the bending moment and the shear force at the clamped end with the number of elements. The values for M_1 and Q_1 reported are the extrapolation of the values at the elements center. Note that M_1 and Q_1 can be

```
% One Gauss point for resultant stresses evaluation
   gaus0 =  0.0;
   bmat_b=[ 0, -1/len, 0, 1/len];
   bmat_s1=[-1/len,-(1-gaus0)/2, 1/len,-(1+gaus0)/2];

% Resultant stresses at Gauss points
   Str1_g0 = D_matb*(bmat_b *transpose(u_elem));
   Str2_g0 = D_mats*(bmat_s1*transpose(u_elem));

% Nodal extrapolation of resultant stresses
   Strnod(lnods(1),1) = Strnod(lnods(1),1)+Str1_g0;
   Strnod(lnods(2),1) = Strnod(lnods(2),1)+Str1_g0;
   Strnod(lnods(1),2) = Strnod(lnods(1),2)+Str2_g0;
   Strnod(lnods(2),2) = Strnod(lnods(2),2)+Str2_g0;
   Strnod(lnods(1),3) = Strnod(lnods(1),3)+1;
   Strnod(lnods(2),3) = Strnod(lnods(2),3)+1;
```

Fig. 12.33 2-noded Timoshenko beam element. Computation of bending moment M and shear force Q at the element center and the element nodes

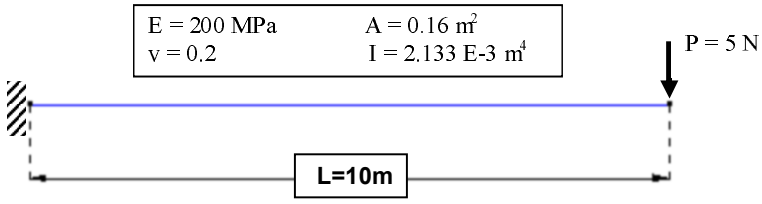


Fig. 12.34 Thick cantilever beam ($r = 25$) under end point load

Thick cantilever beam under end point load 2-noded Timoshenko beam element					
Elements	Nodes	DOF	ω_N	M_1	Q_1
2	3	6	-0,00007	-0,68	5.00
4	5	10	-0,00027	-3,00	5.00
8	9	18	-0,00089	-10,68	5.00
16	17	34	-0,00212	-26,21	5.00
32	33	66	-0,00323	-40,61	5.00
64	65	130	-0,00371	-47,11	5.00
128	129	258	-0,00386	-49,15	5.00
Exact [Ti2]:			-0,00391	-50.00	5.00

Table 12.6 Thick cantilever beam ($r = 25$) under end point load analyzed with 2-noded Timoshenko beam elements. Convergence of free end deflection (w_N) and bending moment (M_1) and shear force (Q_1) at the clamped node

directly computed as the reactions at the clamped node *which would yield the exact solution* for all meshes.

```
%  Material properties (constant over the domain).

   aux0 = thick^3 / 12 ;
   aux1 = aux0*young/(1-poiss^2);
   aux2 = poiss*aux1;
   aux3 = aux0*young/2/(1+poiss);

   D_matb  = [aux1,aux2,   0;
              aux2,aux1,   0;
                 0,   0,aux3];
```

Fig. 12.35 Bending constitutive matrix for thin plate elements

```
    gauss_x(1) =-1/sqrt(3);
    gauss_y(1) =-1/sqrt(3);
    gauss_x(2) = 1/sqrt(3);
    gauss_y(2) =-1/sqrt(3);
    gauss_x(3) = 1/sqrt(3);
    gauss_y(3) = 1/sqrt(3);
    gauss_x(4) =-1/sqrt(3);
    gauss_y(4) = 1/sqrt(3);
```

Fig. 12.36 Local coordinates of Gauss points

12.14 4-NODED MZC THIN PLATE RECTANGLE

This plate element was studied in Section 5.4.1.

12.14.1 Element stiffness matrix and equivalent nodal force vector

Figure 12.35 shows the subroutine for computing the constitutive matrix for thin plate elements (Eq.(5.15b)) including bending terms only.

The Gauss point coordinates in the local axes s, t are shown in Figure 12.36. Recall that the weights for this quadrature are the unity.

Figure 12.37 shows the subroutine for computing $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ for the MZC plate rectangle. A 2×2 Gauss quadrature is used for the integration of $\mathbf{K}^{(e)}$. The computation of the bending strain matrix is shown in Figure 12.38.

Indeed, for this element the analytical expressions for the bending stiffness matrix shown in Box 5.1 could have been used directly. For didactic reasons, however, we have preferred to implement the numerical integration of $\mathbf{K}^{(e)}$ using a 2×2 Gauss quadrature.

The last lines of Figure 12.37 show the computation of the equivalent nodal force vector for self-weight (`denss*thick`) and a uniformly distributed load (`uniload`). The expression of $\mathbf{f}^{(e)}$ coincides with Eq.(5.47a).

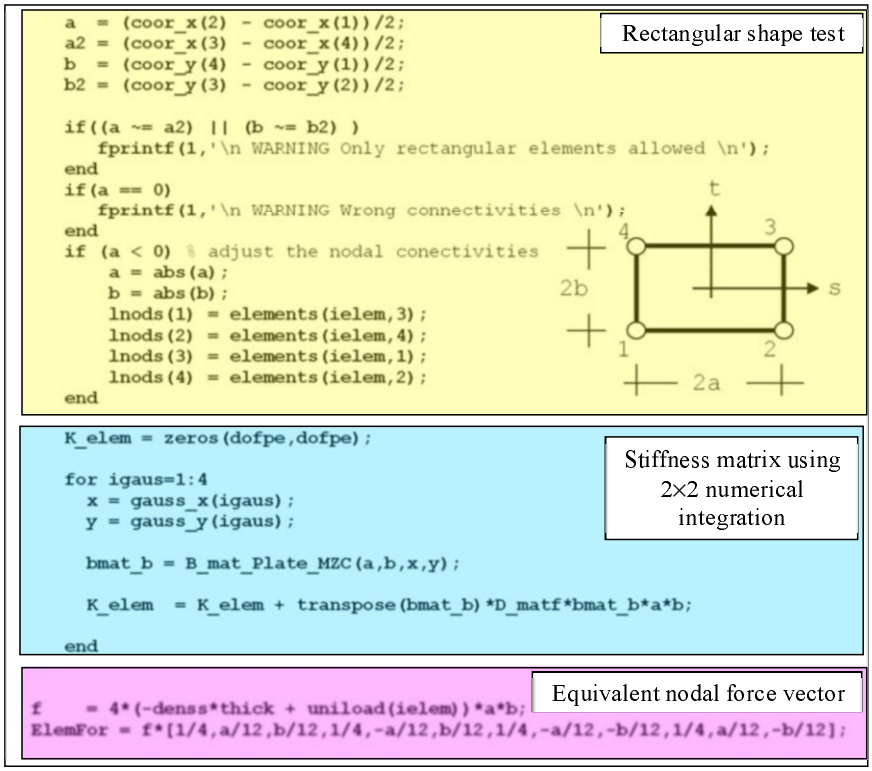


Fig. 12.37 4-noded MZC thin plate rectangle. Stiffness matrix and equivalent nodal force vector

12.14.2 Computation of bending moments

Figure 12.39 shows the subroutine for computing the bending moments for the 4-noded MZC plate rectangle. The moments are computed at the 2×2 Gauss point in `Str1` and then are stored in `Strnod` for the subsequent nodal smoothing.

12.14.3 Example. Clamped thin square plate under uniform loading

Figure 12.40 shows the plate geometry, the material properties and the uniform load value. The analytical values for the deflection and the bending moments at the center are -0.12653 and -2.31, respectively. Units are in the International System (SI).

Figure 12.41 show the menus for introducing the boundary conditions, the material properties and the load.

```
function bmat = B_mat_Plate_MZC(a,b,x,y)

      d2N(1,1) = 3*( x - x*y )/(4*a^2);
      d2N(2,1) = 3*(-x + x*y )/(4*a^2);
      d2N(3,1) = 3*(-x - x*y )/(4*a^2);
      d2N(4,1) = 3*( x + x*y )/(4*a^2);
      d2N(1,2) = 3*( y - x*y )/(4*b^2);
      d2N(2,2) = 3*( y + x*y )/(4*b^2);
      d2N(3,2) = 3*(-y - x*y )/(4*b^2);
      d2N(4,2) = 3*(-y + x*y )/(4*b^2);
      d2N(1,3) = 2*(  1/2 - 3*x^2/8 - 3*y^2/8)/(a*b);
      d2N(2,3) = 2*( -1/2 + 3*x^2/8 + 3*y^2/8)/(a*b);
      d2N(3,3) = 2*(  1/2 - 3*x^2/8 - 3*y^2/8)/(a*b);
      d2N(4,3) = 2*( -1/2 + 3*x^2/8 + 3*y^2/8)/(a*b);

      d2NN(1,1) = ( (3*a*x - 3*a*x*y - a + a*y)/4 )/a^2;
      d2NN(2,1) = ( (3*a*x - 3*a*x*y + a - a*y)/4 )/a^2;
      d2NN(3,1) = ( (3*a*x + 3*a*x*y + a + a*y)/4 )/a^2;
      d2NN(4,1) = ( (3*a*x + 3*a*x*y - a - a*y)/4 )/a^2;
      d2NN(1,2) = 0;
      d2NN(2,2) = 0;
      d2NN(3,2) = 0;
      d2NN(4,2) = 0;
      d2NN(1,3) = 2*( -3/8*a*x^2 + a*x/4 + a/8 )/(a*b);
      d2NN(2,3) = 2*( -3/8*a*x^2 - a*x/4 + a/8 )/(a*b);
      d2NN(3,3) = 2*(  3/8*a*x^2 + a*x/4 - a/8 )/(a*b);
      d2NN(4,3) = 2*(  3/8*a*x^2 - a*x/4 - a/8 )/(a*b);

      d2NNN(1,1) = 0;
      d2NNN(2,1) = 0;
      d2NNN(3,1) = 0;
      d2NNN(4,1) = 0;
      d2NNN(1,2) = ( (3*b*y - 3*b*x*y - b + b*x)/4 )/b^2;
      d2NNN(2,2) = ( (3*b*y + 3*b*x*y - b - b*x)/4 )/b^2;
      d2NNN(3,2) = ( (3*b*y + 3*b*x*y + b + b*x)/4 )/b^2;
      d2NNN(4,2) = ( (3*b*y - 3*b*x*y + b - b*x)/4 )/b^2;
      d2NNN(1,3) = 2*( -3/8*b*y^2 + b*y/4 + b/8 )/(a*b);
      d2NNN(2,3) = 2*(  3/8*b*y^2 - b*y/4 - b/8 )/(a*b);
      d2NNN(3,3) = 2*(  3/8*b*y^2 + b*y/4 - b/8 )/(a*b);
      d2NNN(4,3) = 2*( -3/8*b*y^2 - b*y/4 + b/8 )/(a*b);

      bmat_1  = [ -d2N(1,1),-d2NN(1,1),-d2NNN(1,1)  ;
                  -d2N(1,2),-d2NN(1,2),-d2NNN(1,2)  ;
                  -d2N(1,3),-d2NN(1,3),-d2NNN(1,3)] ;
      bmat_2  = [ -d2N(2,1),-d2NN(2,1),-d2NNN(2,1)  ;
                  -d2N(2,2),-d2NN(2,2),-d2NNN(2,2)  ;
                  -d2N(2,3),-d2NN(2,3),-d2NNN(2,3)] ;
      bmat_3  = [ -d2N(3,1),-d2NN(3,1),-d2NNN(3,1)  ;
                  -d2N(3,2),-d2NN(3,2),-d2NNN(3,2)  ;
                  -d2N(3,3),-d2NN(3,3),-d2NNN(3,3)] ;
      bmat_4  = [ -d2N(4,1),-d2NN(4,1),-d2NNN(4,1)  ;
                  -d2N(4,2),-d2NN(4,2),-d2NNN(4,2)  ;
                  -d2N(4,3),-d2NN(4,3),-d2NNN(4,3)] ;

      bmat = [bmat_1,bmat_2,bmat_3,bmat_4];
```

Evaluation of second derivatives of shape functions $\mathbf{N}$

Evaluation of second derivatives of shape functions $\bar{\mathbf{N}}$

Evaluation of second derivatives of shape functions $\bar{\bar{\mathbf{N}}}$

Assembly of $\mathbf{B}_b$ matrix

Fig. 12.38 Bending strain matrix for the MZC plate rectangle

```
% Shape function matrix for extrapolation of bending moments to nodes
  aa = 1 + sqrt(3);
  bb = 1 - sqrt(3);
  mstres = [ aa*aa , aa*bb , bb*bb , aa*bb ;
             bb*aa , aa*aa , aa*bb , bb*bb ;
             bb*bb , aa*bb , aa*aa , bb*aa ;
             aa*bb , bb*bb , bb*aa , aa*aa ]/4;

% Bending moments at Gauss point
    for igaus=1:4
      x = gauss_x(igaus);
      y = gauss_y(igaus);

      bmat = B_mat_Plate_MZC(a,b,x,y);

      Str1=D_matb*bmat*transpose(u_elem);

      Strx(igaus)  = Str1(1);
      Stry(igaus)  = Str1(2);
      Strxy(igaus) = Str1(3);

    End

% Nodal extrapolation of bending moments
    Str1 = mstres * transpose(Strx) ;
    Strnod(lnods(1:4),1) = Strnod(lnods(1:4),1)+ Str1(1:4);
    Str1 = mstres * transpose(Stry) ;
    Strnod(lnods(1:4),2) = Strnod(lnods(1:4),2)+ Str1(1:4);
    Str1 = mstres * transpose(Strxy) ;
    Strnod(lnods(1:4),3) = Strnod(lnods(1:4),3)+ Str1(1:4);
    Strnod(lnods(1:4),4) = Strnod(lnods(1:4),4)+ 1;
```

Fig. 12.39 Computation of bending moments for the MZC plate rectangle

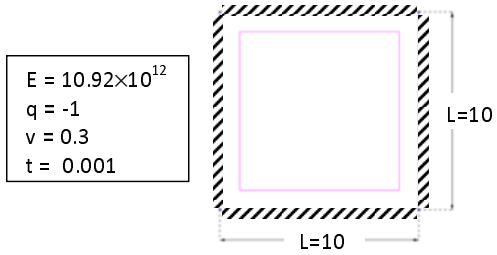


Fig. 12.40 Clamped thin square plate under uniformly distributed loading.Units are in the SI system

The discretization and the writing of the data file is performed with the last two bottoms of the MAT-fem-Plates menu. The problem has been solved with several meshes ranging from 2×2 to 20×20 MZC elements.

Figure 12.42 shows the input data file for the 2×2 mesh.

Table 12.7 shows the convergence of the central deflection and the bending moment M_x at the plate center with the number of elements. No advantage has been taken of the symmetry of the problem.

Figure 12.43 shows the contour plots for the vertical deflection and the bending moment M_x on the plate for a 10×10 element mesh.

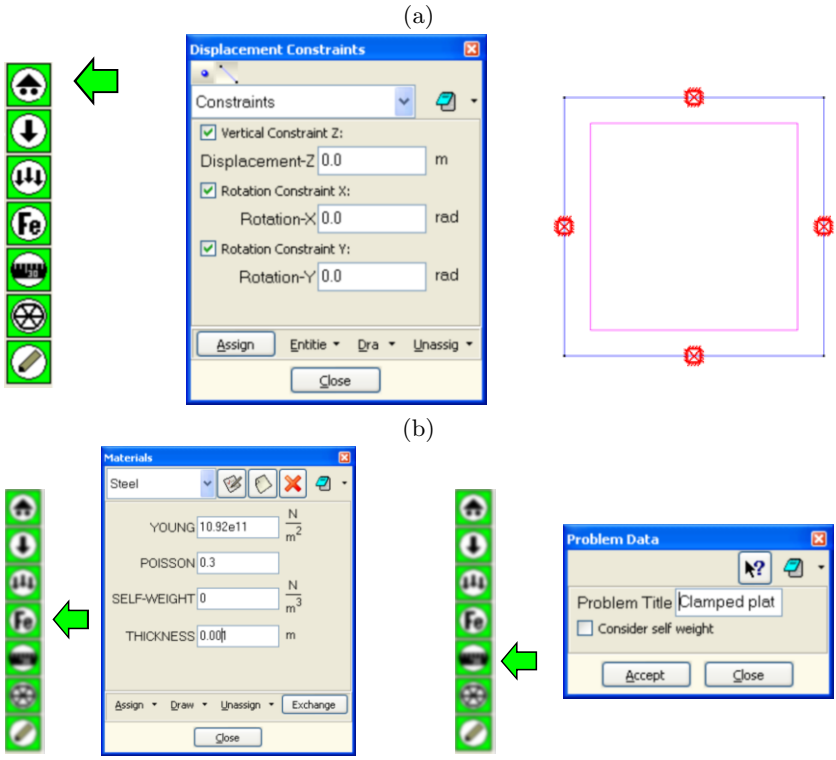


Fig. 12.41 MZC Definition of boundary conditions (a), material properties and distributed load (b) for the clamped thin square plate

Thin clamped square plate under uniform loading			
Mesh of MZC elements	Nodes	ω_c	M_{x_c}
2 × 2	9	-0.15625	-4.88
4 × 4	25	-0.14077	-2.80
6 × 6	49	-0.13333	-2.50
8 × 8	81	-0.13043	-2.41
10 × 10	121	-0.12904	-2.36
12 × 12	169	-0.12828	-2.34
14 × 14	225	-0.12782	-2.33
16 × 16	289	-0.12752	-2.32
18 × 18	361	-0.12731	-2.31
20 × 20	441	-0.12716	-2.31
Exact [TW]:		-0.12653	-2.31

Table 12.7 Thin clamped square plate under uniform load. Convergence of central deflection ω_c and central bending moment M_{x_c} using the MZC plate rectangle

```
%=====================================================================
% MAT-fem_Plates 1.0  - MAT-fem is a learning tool for undestanding
%                       the Finite Element Method with MATLAB and GiD
%=====================================================================
% PROBLEM TITLE = Clamped square plate analyzed with 2×2 MZC plate rectangles
%  Material Properties
 young  = 1.092e+12 ;
 poiss  = 0.3 ;
 thick  = 0.001;
 denss  = 0.00;
% Coordinates
global coordinates
coordinates = [
    0.0,     10.0 ;
    5.0,     10.0 ;
    0.0,      5.0 ;
    5.0,      5.0 ;
    0.0,      0.0 ;
   10.0,     10.0 ;
    5.0,      0.0 ;
   10.0,      5.0 ;
   10.0,      0.0 ];
% Elements
global elements
elements = [
      5,  7,  4,  3;
      7,  9,  8,  4;
      3,  4,  2,  1;
      4,  8,  6,  2];
% Fixed nodes
global fixdesp
fixdesp  = [
    1, 1,  0.0;
    1, 2,  0.0;
    1, 3,  0.0;
    2, 1,  0.0;
    2, 2,  0.0;
    2, 3,  0.0;
...
    9, 1,  0.0;
    9, 2,  0.0;
    9, 3,  0.0];
% Point loads
pointload = [ ];
% Distributed load
uniload = sparse ( 4 , 1 );
uniload (      1  ) =  -1.000000000e+00  ;
uniload (      2  ) =  -1.000000000e+00  ;
uniload (      3  ) =  -1.000000000e+00  ;
uniload (      4  ) =  -1.000000000e+00  ;
```

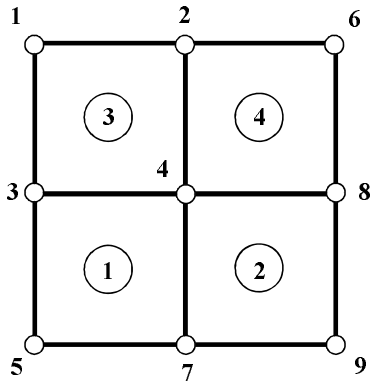


Fig. 12.42 Input data for clamped square plate analyzed with 2×2 MZC plate rectangles

12.15 Q4 REISSNER-MINDLIN PLATE RECTANGLE

We present the key subroutines for the 4-noded Q4 Reissner-Mindlin (RM) plate element of Section 6.5.1 *in its rectangular form*. The stiffness matrix is computed with selective integration, i.e. a 2×2 quadrature for the bending stiffness terms and a reduced one point quadrature for the shear

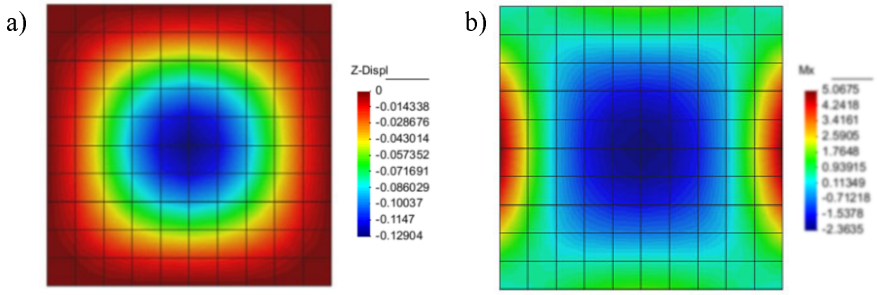


Fig. 12.43 Thin clamped square plate under uniform load. Contours of vertical deflection w (a) and M_x (b) for a mesh of 10×10 MZC plate rectangles

```
aux4 = (5/6)*thick*young/2/(1+poiss);

D_mats = [aux4,    0 ;
             0,aux4];
```

Fig. 12.44 Shear constitutive matrix for Reissner-Mindlin plate elements

stiffness terms. These subroutines can be easily extended for programming higher order RM plate rectangles based on selective integration techniques, as described in Section 6.5. The extension to non-rectangular shapes is straightforward using an isoparametric formulation. An isoparametric Q4 RM plate quadrilateral has been implemented in the `Plate_Q4_Iso` code.

12.15.1 Stiffness matrix and equivalent nodal force vector

The bending terms in the constitutive matrix coincide with those of Figure 12.35. The shear constitutive matrix $\hat{\mathbf{D}}_s$ of Eq.(6.24) is shown in Figure 12.44.

Figure 12.45 shows the subroutine for computing the stiffness matrix and the equivalent nodal force vector for the Q4 RM plate rectangle. Note the two loops for computing the bending and shear stiffness matrices using selective integration.

Figure 12.46 shows the computation of the generalized bending and shear strain matrices.

The last two rows of Figure 12.45 show the computation of the equivalent nodal force for self-weight (`dense*thick`) and a uniformly distributed load (`uniload`). As the element is a rectangle the nodal forces are simply computed as one fourth of the total force acting over the element.

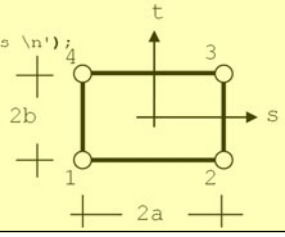
```
% Local Gauss point coordinates

gauss_x(1) =-1/sqrt(3);
gauss_y(1) =-1/sqrt(3);
gauss_x(2) = 1/sqrt(3);
gauss_y(2) =-1/sqrt(3);
gauss_x(3) = 1/sqrt(3);
gauss_y(3) = 1/sqrt(3);
gauss_x(4) =-1/sqrt(3);
gauss y(4) = 1/sqrt(3);
```

```
a  = (coor_x(2) - coor_x(1))/2;
a2 = (coor_x(3) - coor_x(4))/2;
b  = (coor_y(4) - coor_y(1))/2;
b2 = (coor_y(3) - coor_y(2))/2;

if((a ~= a2) || (b ~= b2) )
   fprintf(1,'\n WARNING Only rectangular elements allowed \n');
end
if(a == 0)
   fprintf(1,'\n WARNING Wrong connectivities \n');
end
if (a < 0) % adjust the nodal conectivities
    a = abs(a);
    b = abs(b);
    lnods(1) = elements(ielem,3);
    lnods(2) = elements(ielem,4);
    lnods(3) = elements(ielem,1);
    lnods(4) = elements(ielem,2);
end
```

Rectangular shape test




```
K_elem = zeros(dofpe,dofpe);

 for igaus=1:4
  x = gauss_x(igaus);
  y = gauss_y(igaus);

  [bmat_b, ~] = B_mat_Plate_Q4_v2_3(a,b,x,y);
  K_b  = transpose(bmat_b)*D_matb*bmat_b*a*b;
  K_elem  = K_elem + K_b;
end

% One gauss point for shear
x = 0;
y = 0;
[~,bmat_s] = B_mat_Plate_Q4_v2_3(a,b,x,y);
K_s  = transpose(bmat_s)*D_mats*bmat_s*4*a*b;
K_elem  = K_elem + K_s;

 f       = 4*(-denss*thick + uniload(ielem))*a*b;
ElemFor = f*[1/4,0,0,1/4,0,0,1/4,0,0,1/4,0,0];
```

Numerical integration of stiffness matrix

Equivalent nodal force vector

Fig. 12.45 Q4 Reissner-Mindlin plate rectangle. Stiffness matrix and equivalent nodal force vector

12.15.2 Computation of resultant stresses

Figure 12.47 shows the subroutine for computing the resultant stresses for the Q4 RM plate rectangle. The bending moment (`Str1`) are computed at the 2×2 Gauss points, while the shear forces (`Str2`) are computed at

```
function [bmat_b,bmat_s] = B_mat_Plate_Q4(a,b,x,y)

      x = x*a;
      y = y*b;

      N(1) = (1 - x/a)*(1 - y/b)/4;
      N(2) = (1 + x/a)*(1 - y/b)/4;
      N(3) = (1 + x/a)*(1 + y/b)/4;
      N(4) = (1 - x/a)*(1 + y/b)/4;

      dxN(1) = -( b - y )/(4*a*b);
      dxN(2) =  ( b - y )/(4*a*b);
      dxN(3) =  ( b + y )/(4*a*b);
      dxN(4) = -( b + y )/(4*a*b);

      dyN(1) = -( a - x )/(4*a*b);
      dyN(2) = -( a + x )/(4*a*b);
      dyN(3) =  ( a + x )/(4*a*b);
      dyN(4) =  ( a - x )/(4*a*b);

Bending strain matrix

      bmat_b1  = [ 0,-dxN(1),      0  ;
                   0,       0,-dyN(1) ;
                   0,-dyN(1),-dxN(1)];

      bmat_b2  = [ 0,-dxN(2),      0  ;
                   0,       0,-dyN(2) ;
                   0,-dyN(2),-dxN(2)];

      bmat_b3  = [ 0,-dxN(3),      0  ;
                   0,       0,-dyN(3) ;
                   0,-dyN(3),-dxN(3)];

      bmat_b4  = [ 0,-dxN(4),      0  ;
                   0,       0,-dyN(4) ;
                   0,-dyN(4),-dxN(4)];

      bmat_b = [bmat_b1,bmat_b2,bmat_b3,bmat_b4];

Transverse shear strain matrix

      bmat_s1  = [ dxN(1), -N(1),     0  ;
                   dyN(1),     0, -N(1)];

      bmat_s2  = [ dxN(2), -N(2),     0  ;
                   dyN(2),     0, -N(2)];

      bmat_s3  = [ dxN(3), -N(3),     0  ;
                   dyN(3),     0, -N(3)];

      bmat_s4  = [ dxN(4), -N(4),     0  ;
                   dyN(4),     0, -N(4)];
```

Fig. 12.46 Q4 Reissner-Mindlin plate rectangle. Generalized bending and transverse shear strain matrices

the element center. The resultant stresses are accumulated in `Strnod` for the subsequent nodal smoothing.

12.15.3 Example. Thick clamped square plate under uniformly load

The plate geometry, load and material properties are identical to those of Figure 12.40, with the exception of the thickness that now is $t = 1.0$.

```
% Shape Function matrix for extrapolation of resultant stresses to nodes
  aa = 1 + sqrt(3);
  bb = 1 - sqrt(3);
  mstres = [ aa*aa , aa*bb , bb*bb , aa*bb ;
             bb*aa , aa*aa , aa*bb , bb*bb ;
             bb*bb , aa*bb , aa*aa , bb*aa ;
             aa*bb , bb*bb , bb*aa , aa*aa ]/4;

    a  = (coor_x(2) - coor_x(1))/2;
    b  = (coor_y(4) - coor_y(1))/2;

% Resultant stresses at Gauss point
    for igaus=1:4
      x = gauss_x(igaus);
      y = gauss_y(igaus);
      [bmat_b, ~] = B_mat_Plate_Q4_v2_3(a,b,x,y);

      Str1=D_matb*bmat_b*transpose(u_elem);

      Strx(igaus)  = Str1(1);
      Stry(igaus)  = Str1(2);
      Strxy(igaus) = Str1(3);
    end

% Resultant stresses at nodes
    x = 0;
    y = 0;
    [~,bmat_s] = B_mat_Plate_Q4_v2_3(a,b,x,y);
    Str2=D_mats*bmat_s*transpose(u_elem);
    StrQx(1:4) = Str2(1);
    StrQy(1:4) = Str2(2);

    Str1 = mstres * transpose(Strx) ;
    Strnod(lnods(1:4),1) = Strnod(lnods(1:4),1)+ Str1(1:4);
    Str1 = mstres * transpose(Stry) ;
    Strnod(lnods(1:4),2) = Strnod(lnods(1:4),2)+ Str1(1:4);
    Str1 = mstres * transpose(Strxy) ;
    Strnod(lnods(1:4),3) = Strnod(lnods(1:4),3)+ Str1(1:4);
    Str2 = mstres * transpose(StrQx) ;
    Strnod(lnods(1:4),4) = Strnod(lnods(1:4),4)+ Str2(1:4);
    Str2 = mstres * transpose(StrQy) ;
    Strnod(lnods(1:4),5) = Strnod(lnods(1:4),5)+ Str2(1:4);
    Strnod(lnods(1:4),6) = Strnod(lnods(1:4),6)+ 1;
```

Fig. 12.47 Q4 Reissner-Mindlin plate rectangle. Computation of bending moments and shear forces

The reference solution for this problem is: central deflection, $w_c = -0.1505 \times 10^{-9}$, central bending moment $Mx_c = -2.31$ and maximum shear force at the clamped edge $Q_{y_m} = 4.12$ [TW].

The data input process is the same as for the MZC rectangle (Section 12.4). The input data file is similar as that shown in Figure 12.42.

Table 12.8 lists the results for w_c, M_{x_c} and Q_{y_m} for different meshes of Q4 RM plate rectangles.

12.16 QLLL REISSNER-MINDLIN PLATE QUADRILATERAL

We present the main parts of the Mat-fem code for the 4-noded QLLL Reissner-Mindlin plate quadrilateral studied in Section 6.7.1. The *element is derived using an isoparametric formulation* and, therefore, is not restricted to rectangular shapes.

Q4 Reissner Mindlin plate rectangle				
Mesh	Nodes	w_c	Mx_c	Q_{y_m}
2 × 2	9	-0.3571E-09	-0.00	1.250
4 × 4	25	-0.1458E-09	-2.262	2.524
6 × 6	49	-0.1486E-09	-2.408	3.159
8 × 8	81	-0.1494E-09	-2.339	3.374
10 × 10	121	-0.1498E-09	-2.337	3.526
12 × 12	169	-0.1500E-09	-2.331	3.626
14 × 14	225	-0.1501E-09	-2,329	3.700
16 × 16	289	-0.1502E-09	-2,327	3.753
18 × 18	361	-0.1502E-09	-2,325	3.796
20 × 20	441	-0.1503E-09	-2,324	3.830
Exact [TW]:		-0.1504E-09	-2.310	4.120

Table 12.8 Thick clamped square plate under uniform load. Central deflection w_c, central bending moment Mx_c and maximum shear force at the clamped edge Q_{y_m} for different meshes of Q4 Reissner-Mindlin plate rectangles

12.16.1 Stiffness matrix and equivalent nodal force vector

The *constitutive matrix* includes the bending and shear contributions as defined in Figures 12.35 and 12.44.

Figure 12.48 shows the subroutine for computing the bending and shear stiffness matrices (termed `K_b` and `K_s`, respectively). A 2 × 2 Gauss quadrature is used for the numerical integration.

The expression for the bending stiffness coincides precisely with $\mathbf{K}_b^{(e)}$ given in Eq.(6.39). The computation of the bending strain matrix is shown in Figure 12.49.

The shear stiffness matrix is obtained by substituting matrix $\mathbf{B}_s$ by $\bar{\mathbf{B}}_s$ in the expression of $\mathbf{K}_s^{(e)}$ of Eq.(6.39b). The derivation of the substitute shear strain matrix $\bar{\mathbf{B}}_s$ for the QLLL element is detailed in Section 6.7.1. The subroutine for computing $\bar{\mathbf{B}}_s$ is shown in Figure 12.50.

12.16.2 Computation of resultant stresses

Figure 12.51 shows the subroutine for computing the bending moments (`Str1`) and the shear forces (`Str2`) at the 2 × 2 Gauss point in the QLLL RM plate quadrilateral.

The resultant stresses are accumulated in `Strnod` for the subsequent smoothing.

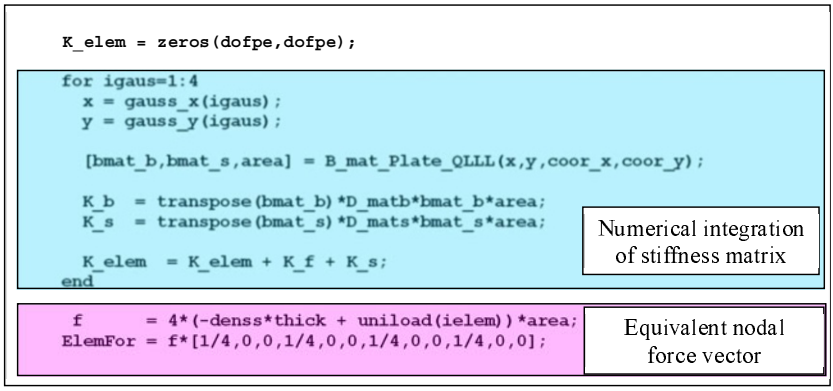


Fig. 12.48 4-noded QLLL Reissner-Mindlin plate quadrilateral. Computation of the stiffness matrix and the equivalent nodal force vector

QLLL plate quadrilateral				
Mesh	Nodes	w_c	M_{x_c}	Q_{y_m}
2 × 2	9	-0.026E-09	-0.000	1.875
4 × 4	25	-0.143E-09	-2.364	3.253
6 × 6	49	-0.147E-09	-2.367	3.331
8 × 8	81	-0.148E-09	-2.343	3.494
10 × 10	121	-0.149E-09	-2.334	3.602
12 × 12	169	-0.149E-09	-2.330	3.680
14 × 14	225	-0.149E-09	-2.327	3.740
16 × 16	289	-0.150E-09	-2,325	3.786
18 × 18	361	-0.150E-09	-2,324	3.822
20 × 20	441	-0.150E-09	-2,323	3.852
Exact [TW]:		-0.150E-09	-2.31	4.55

TQQL plate triangle				
Mesh	Nodes	w_c	M_{x_c}	Q_{y_m}
2 × 2	25	-0.63E-09	-0.97	3.62
4 × 4	81	-0.202E-09	-1.11	5.40
6 × 6	169	-0.197E-09	-1.84	4.65
8 × 8	289	-0.182E-09	-2.07	4.32
10 × 10	441	-0.173E-09	-2.16	4.21
12 × 12	625	-0.167E-09	-2.21	4.16
14 × 14	841	-0.163E-09	-2.24	4.14
16 × 16	1089	-0.160E-09	-2,26	4.13
18 × 18	1369	-0.158E-09	-2,27	4.12
20 × 20	1681	-0.157E-09	-2,28	4.11
Exact [TW]:		-0.150E-09	-2.31	4.55

Table 12.9 Thick clamped square plate under uniform loading. Results for deflection w_c and bending moment M_{x_c} at the plate center and maximum shear force Q_{y_m} at the clamped edge for different meshes of QLLL and TQQL plate elements

12.16.3 Example. Thick clamped plate under uniform distributed load

The example coincides with that used in Section 12.15.3 for testing the Q4 RM plate rectangle.

Table 12.9 shows the results for the deflection w_c and the bending moment Mx_c at the plate center and the maximum shear force Q_{y_m} at the clamped edge for different meshes of QLLL elements. Results for the same problem solved with TQQL elements are presented.

Note the excellent accuracy for the deflection and bending moment values for relatively coarse meshes.

```
function [bmat_b,bmat_s,area] = B_mat_Plate_QLLL(xgs,ygs,x,y)

 dxNl(1) = (-1+ygs)/4;
 dxNl(2) = ( 1-ygs)/4;
 dxNl(3) = ( 1+ygs)/4;
 dxNl(4) = (-1-ygs)/4;

 dyNl(1) = (-1+xgs)/4;
 dyNl(2) = (-1-xgs)/4;
 dyNl(3) = ( 1+xgs)/4;
 dyNl(4) = ( 1-xgs)/4;

 xjacm(1,1) = x(1)*dxNl(1) + x(2)*dxNl(2) + x(3)*dxNl(3) + x(4)*dxNl(4);
 xjacm(1,2) = y(1)*dxNl(1) + y(2)*dxNl(2) + y(3)*dxNl(3) + y(4)*dxNl(4);
 xjacm(2,1) = x(1)*dyNl(1) + x(2)*dyNl(2) + x(3)*dyNl(3) + x(4)*dyNl(4);
 xjacm(2,2) = y(1)*dyNl(1) + y(2)*dyNl(2) + y(3)*dyNl(3) + y(4)*dyNl(4);

 xjaci = inv(xjacm);

 area = abs(xjacm(1,1)*xjacm(2,2) - xjacm(2,1)*xjacm(1,2));

 dxN(1) = xjaci(1,1)*dxNl(1)+xjaci(1,2)*dyNl(1);
 dxN(2) = xjaci(1,1)*dxNl(2)+xjaci(1,2)*dyNl(2);
 dxN(3) = xjaci(1,1)*dxNl(3)+xjaci(1,2)*dyNl(3);
 dxN(4) = xjaci(1,1)*dxNl(4)+xjaci(1,2)*dyNl(4);

 dyN(1) = xjaci(2,1)*dxNl(1)+xjaci(2,2)*dyNl(1);
 dyN(2) = xjaci(2,1)*dxNl(2)+xjaci(2,2)*dyNl(2);
 dyN(3) = xjaci(2,1)*dxNl(3)+xjaci(2,2)*dyNl(3);
 dyN(4) = xjaci(2,1)*dxNl(4)+xjaci(2,2)*dyNl(4);
%===================
      bmat_b1  = [ 0,-dxN(1),     0  ;
                   0,      0,-dyN(1) ;
                   0,-dyN(1),-dxN(1)];
      bmat_b2  = [ 0,-dxN(2),     0  ;
                   0,      0,-dyN(2) ;
                   0,-dyN(2),-dxN(2)];
      bmat_b3  = [ 0,-dxN(3),     0  ;
                   0,      0,-dyN(3) ;
                   0,-dyN(3),-dxN(3)];
      bmat_b4  = [ 0,-dxN(4),     0  ;
                   0,      0,-dyN(4) ;
                   0,-dyN(4),-dxN(4)];

      bmat_b = [bmat_b1,bmat_b2,bmat_b3,bmat_b4];
===================
```

Cartesian derivatives of the shape functions

Bending strain matrix $\mathbf{B}_b$

Fig. 12.49 Computation of the bending strain matrix for the 4-noded QLLL plate quadrilateral

12.17 TQQL REISSNER-MINDLIN PLATE TRIANGLE

The TQQL 6-noded plate element was studied in Section 6.8.1. We present next the main subroutines for programming the element using an isoparametric formulation.

12.17.1 Stiffness matrix and equivalent nodal force vector

Figure 12.52 shows the local coordinates of the 3-point Gauss quadrature used for the numerical integration of all the element matrices.

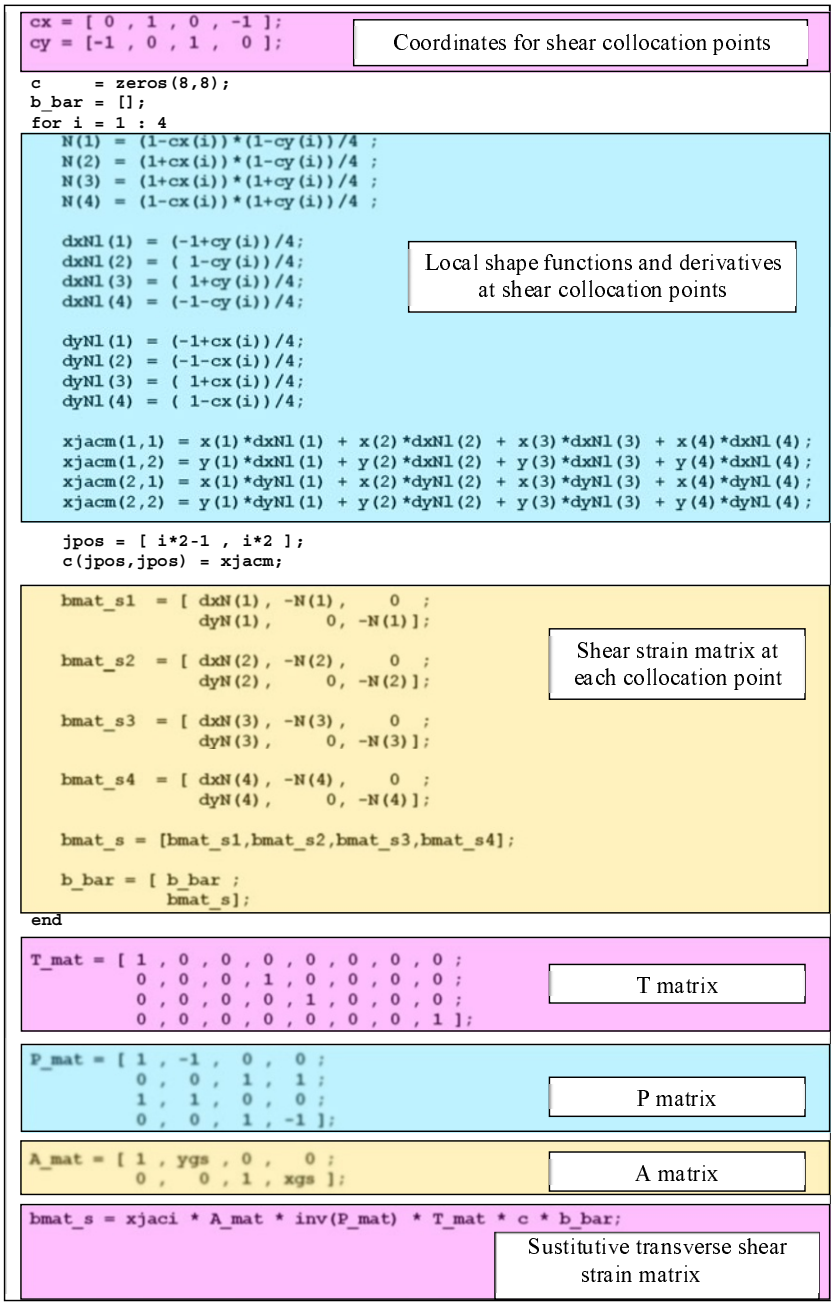


Fig. 12.50 4-noded QLLL plate quadrilateral. Computation of the substitute transverse shear strain matrix $\bar{\mathbf{B}}_s$

```
% Shape Function matrix for extrapolation of resultant stresses to nodes
  aa = 1 + sqrt(3);
  bb = 1 - sqrt(3);
  mstres = [ aa*aa , aa*bb , bb*bb , aa*bb ;
             bb*aa , aa*aa , aa*bb , bb*bb ;
             bb*bb , aa*bb , aa*aa , bb*aa ;
             aa*bb , bb*bb , bb*aa , aa*aa ]/4;

% Resultant stresses at Gauss points
    for igaus=1:4
      x = gauss_x(igaus);
      y = gauss_y(igaus);

      [bmat_b,bmat_s,area] = B_mat_Plate_QLLL(x,y,coor_x,coor_y);

      Str1=D_matb*bmat_b*transpose(u_elem);
      Str2=D_mats*bmat_s*transpose(u_elem);

      Strx(igaus)  = Str1(1);
      Stry(igaus)  = Str1(2);
      Strxy(igaus) = Str1(3);
      StrQx(igaus) = Str2(1);
      StrQy(igaus) = Str2(2);
    end

% Resultant stresses at nodes
    Str1 = mstres * transpose(Strx) ;
    Strnod(lnods(1:4),1) = Strnod(lnods(1:4),1)+ Str1(1:4);
    Str1 = mstres * transpose(Stry) ;
    Strnod(lnods(1:4),2) = Strnod(lnods(1:4),2)+ Str1(1:4);
    Str1 = mstres * transpose(Strxy) ;
    Strnod(lnods(1:4),3) = Strnod(lnods(1:4),3)+ Str1(1:4);
    Str2 = mstres * transpose(StrQx) ;
    Strnod(lnods(1:4),4) = Strnod(lnods(1:4),4)+ Str2(1:4);
    Str2 = mstres * transpose(StrQy) ;
    Strnod(lnods(1:4),5) = Strnod(lnods(1:4),5)+ Str2(1:4);
    Strnod(lnods(1:4),6) = Strnod(lnods(1:4),6)+ 1;

  for i = 1 : npnod
    Strnod(i,1:5) = Strnod(i,1:5)/Strnod(i,6);
  end
```

Nodal averaging of resultant stresses

Fig. 12.51 Computation of resultant stresses for the QLLL plate quadrilateral

Figure 12.52 shows also the subroutine for computing the bending and shear stiffness matrices and the equivalent nodal force vector (`ElemFor`) for self-weight (`denss*thick`) and uniformly distributed load (`uniload`).

The bending stiffness matrix (`K_b`) is computed by Eq.(6.39). The computation of the bending strain matrix (`bmat_b`) is detailed in Figure 12.53.

The transverse shear stiffness matrix is obtained by using the substitute transverse shear strain matrix $\bar{\mathbf{B}}_s$ (Section 6.8.1) in the expression of $\mathbf{K}_s^{(e)}$ of Eq.(6.39b). The computation of $\bar{\mathbf{B}}_s$ is shown in Figure 12.54.

12.17.2 Computation of stress resultants

The bending moments (`Str1`) and the shear forces (`Str2`) are computed first at the 3 Gauss point used for the numerical integration of the stiffness

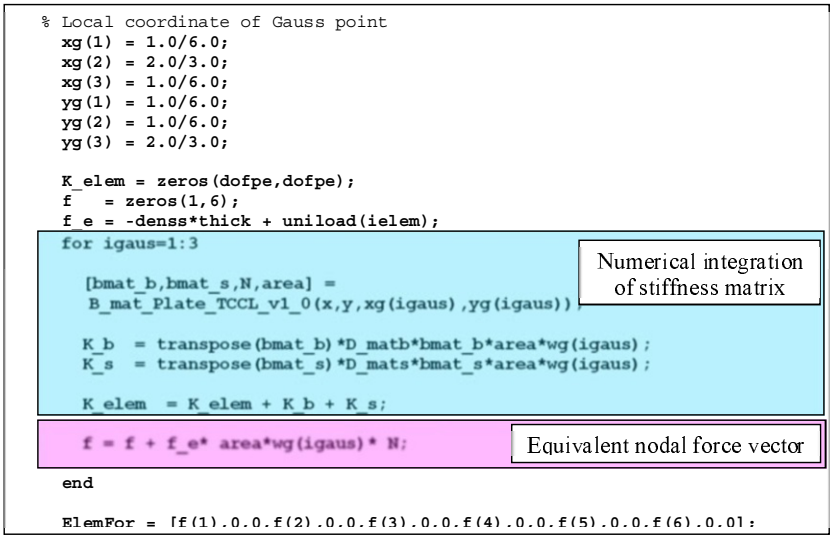


Fig. 12.52 TQQL plate triangle. Computation of stiffness matrix and equivalent nodal force vector

matrix. The Gauss point values for the resultant stresses are accumulated in `Strnod` for the subsequent nodal smoothing (Figure 12.55).

12.17.3 Example. Thick clamped square plate under uniform load

The example coincides with that used in Section 12.15.3 for testing the performance of the Q4 plate rectangle.

Table 12.9 in p. 768 shows the results for the deflection w_c and the bending moment M_{x_c} at the plate center and the maximum shear force Q_{y_m} at the clamped edge for different meshes of TQQL elements.

Note the excellent accuracy for the deflection and the bending moment values for relatively coarse meshes.

12.18 4-NODED QLLL FLAT SHELL ELEMENT

We present the key parts of the Mat-fem code for the 4-noded QLLL flat shell element. This element was studied in Chapter 8 (Sections 8.3–8.11). The element can be used for analysis of any 3D shell structure by discretizing the shell surface into 4-noded quadrilaterals.

The programming of other flat shell elements follows very similar steps as those described for the 4-noded QLLL flat shell element.

```
function [bmat_b,bmat_s,Ngp,area] = B_mat_Plate_TCCL_v1_0(x,y,xgs,ygs)
%==================
  L1=1.0-xgs-ygs;
  L2=xgs;
  L3=ygs;

  dxNloc(1)=1.0-4.0*L1;
  dxNloc(2)=4.0*(L1-L2);
  dxNloc(3)=4.0*L2-1.0;
  dxNloc(4)=4.0*L3;
  dxNloc(5)=0.0;
  dxNloc(6)=-4.0*L3;

  dyNloc(1)=1.0-4.0*L1;
  dyNloc(2)=-4.0*L2;
  dyNloc(3)=0.0;
  dyNloc(4)=4.0*L2;
  dyNloc(5)=4.0*L3-1.0;
  dyNloc(6)=4.0*(L1-L3);
```

Local derivatives of the quadratic shape functions

```
  xjacm(1,1) = x*dxNloc';
  xjacm(1,2) = y*dxNloc';
  xjacm(2,1) = x*dyNloc';
  xjacm(2,2) = y*dyNloc';

  xjaci = inv(xjacm);

  area2 = abs(xjacm(1,1)*xjacm(2,2) - xjacm(2,1)*xjacm(1,2));
  area  = area2/2;
```

Jacobian matrix definition

```
  dxN(1) = xjaci(1,1)*dxNloc(1)+xjaci(1,2)*dyNloc(1);
  dxN(2) = xjaci(1,1)*dxNloc(2)+xjaci(1,2)*dyNloc(2);
  dxN(3) = xjaci(1,1)*dxNloc(3)+xjaci(1,2)*dyNloc(3);
  dxN(4) = xjaci(1,1)*dxNloc(4)+xjaci(1,2)*dyNloc(4);
  dxN(5) = xjaci(1,1)*dxNloc(5)+xjaci(1,2)*dyNloc(5);
  dxN(6) = xjaci(1,1)*dxNloc(6)+xjaci(1,2)*dyNloc(6);

  dyN(1) = xjaci(2,1)*dxNloc(1)+xjaci(2,2)*dyNloc(1);
  dyN(2) = xjaci(2,1)*dxNloc(2)+xjaci(2,2)*dyNloc(2);
  dyN(3) = xjaci(2,1)*dxNloc(3)+xjaci(2,2)*dyNloc(3);
  dyN(4) = xjaci(2,1)*dxNloc(4)+xjaci(2,2)*dyNloc(4);
  dyN(5) = xjaci(2,1)*dxNloc(5)+xjaci(2,2)*dyNloc(5);
  dyN(6) = xjaci(2,1)*dxNloc(6)+xjaci(2,2)*dyNloc(6);
```

Cartesian derivatives of the shape functions.

```
  bmat_b1  = [ 0,-dxN(1),      0  ;
               0,       0,-dyN(1) ;
               0,-dyN(1),-dxN(1)];
  bmat_b2  = [ 0,-dxN(2),      0  ;
               0,       0,-dyN(2) ;
               0,-dyN(2),-dxN(2)];
  bmat_b3  = [ 0,-dxN(3),      0  ;
               0,       0,-dyN(3) ;
               0,-dyN(3),-dxN(3)];
  bmat_b4  = [ 0,-dxN(4),      0  ;
               0,       0,-dyN(4) ;
               0,-dyN(4),-dxN(4)];
  bmat_b5  = [ 0,-dxN(5),      0  ;
               0,       0,-dyN(5) ;
               0,-dyN(5),-dxN(5)];
  bmat_b6  = [ 0,-dxN(6),      0  ;
               0,       0,-dyN(6) ;
               0,-dyN(6),-dxN(6)];

  bmat_b = [bmat_b1,bmat_b2,bmat_b3,bmat_b4,bmat_b5,bmat_b6];
```

Bending strain matrix $\mathbf{B}_b$

Fig. 12.53 TQQL plate triangle. Computation of bending strain matrix

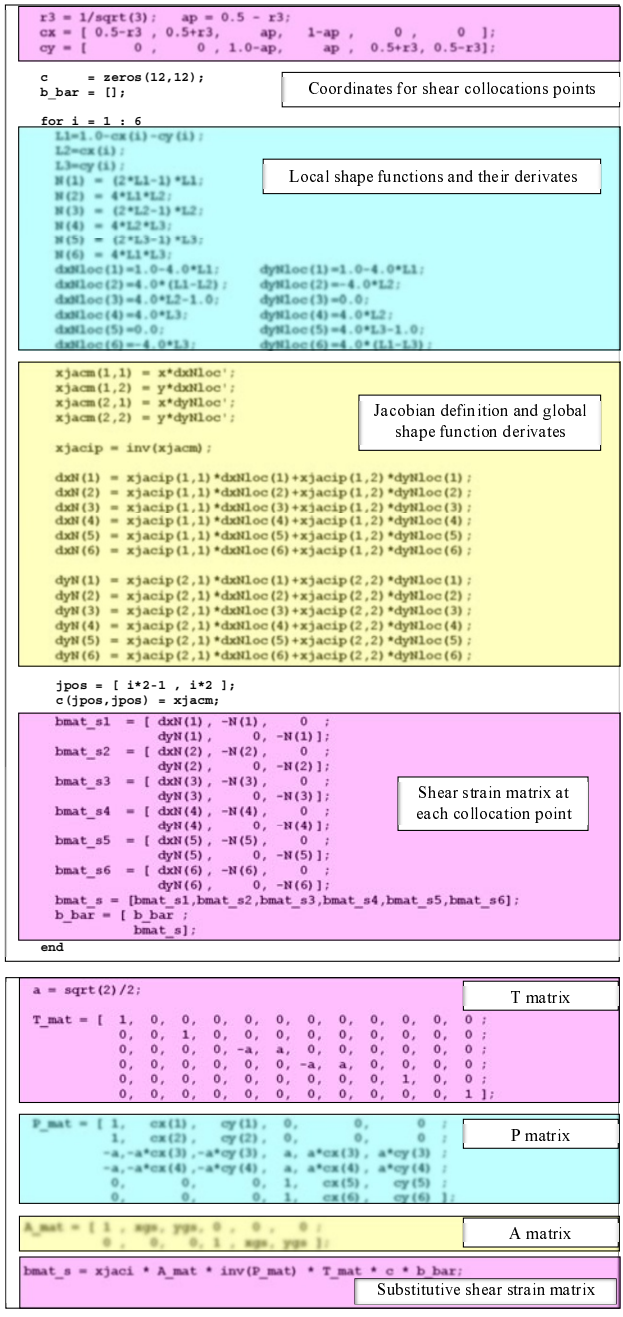


Fig. 12.54 Computation of the substitute transverse shear strain matrix $\bar{\mathbf{B}}_s$ for the TQQL plate triangle

```
% Shape function matrix for extrapolation of resultant stresses to nodes
  mstres = [ 5 , -1 , -1 ;
            -1 ,  5 , -1 ;
            -1 , -1 ,  5 ]/3;
  g_or = [1,3,5]; % Vertex nodes

% Resultant stresses at Gauss points
    for igaus=1:3

      [bmat_b,bmat_s] = B_mat_Plate_TCCL_v1_0(x,y,xg(igaus),yg(igaus));

      Str1=D_matb*bmat_b*transpose(u_elem);
      Str2=D_mats*bmat_s*transpose(u_elem);

      Strx(igaus)  = Str1(1);
      Stry(igaus)  = Str1(2);
      Strxy(igaus) = Str1(3);
      StrQx(igaus) = Str2(1);
      StrQy(igaus) = Str2(2);
    end

% Resultant stresses at nodes
    Str1 = mstres * transpose(Strx) ;
    Strnod(lnods(g_or(1:3)),1) = Strnod(lnods(g_or(1:3)),1)+ Str1(1:3);
    Str1 = mstres * transpose(Stry) ;
    Strnod(lnods(g_or(1:3)),2) = Strnod(lnods(g_or(1:3)),2)+ Str1(1:3);
    Str1 = mstres * transpose(Strxy) ;
    Strnod(lnods(g_or(1:3)),3) = Strnod(lnods(g_or(1:3)),3)+ Str1(1:3);
    Str2 = mstres * transpose(StrQx) ;
    Strnod(lnods(g_or(1:3)),4) = Strnod(lnods(g_or(1:3)),4)+ Str2(1:3);
    Str2 = mstres * transpose(StrQy) ;
    Strnod(lnods(g_or(1:3)),5) = Strnod(lnods(g_or(1:3)),5)+ Str2(1:3);
    Strnod(lnods(g_or(1:3)),6) = Strnod(lnods(g_or(1:3)),6)+ 1;

    for i = 1 : npnod
       Strnod(i,1:5) = Strnod(i,1:5)/Strnod(i,6);
    end
```

Nodal averaging of resultant stresses

Fig. 12.55 Computation of resultant stresses for the TQQL plate triangle

12.18.1 Generalized constitutive matrix

The local generalized constitutive matrix contains the membrane, bending and transverse shear contributions (Eqs.(8.18)). For simplicity, the coupling membrane-bending constitutive matrix $\hat{\mathbf{D}}'_{mb}$ will be neglected.

The computation of the generalized constitutive matrix is shown in Figure 12.56.

12.18.2 Stiffness matrix and equivalent nodal force vector

The first step in the computation of the element stiffness matrix is the definition of the local coordinate system x', y', z'.

The normal vector $\mathbf{v}_{z'}$ is computed as the cross product of vectors joining nodes 1,2 and 1,3. Vector $\mathbf{v}_{x'}$ is found by intersecting the element plane with the global xz plane as described in Section 8.8.2. Vector $\mathbf{v}_{y'}$ is finally found by cross product of $\mathbf{v}_{x'}$ and $\mathbf{v}_{z'}$.

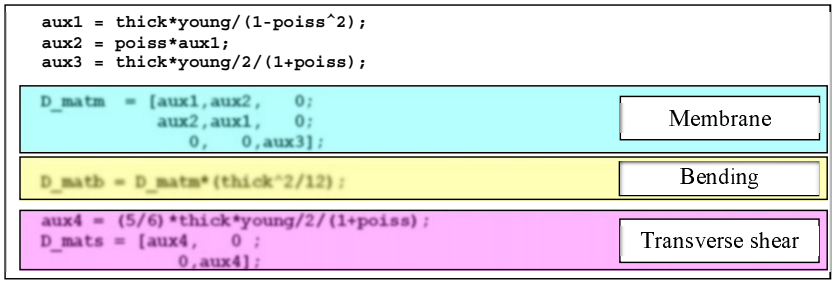


Fig. 12.56 4-noded QLLL flat shell element. Local generalized constitutive matrices

Figure 12.57 shows the computation of vectors $\mathbf{v}_{x'}$, $\mathbf{v}_{y'}$, $\mathbf{v}_{z'}$. These vectors are grouped in matrix `Te`.

All of the stiffness matrix terms are computed with by a 2×2 Gauss quadrature. The computation requires first transforming the global coordinates of the nodes to the local axes. The membrane and bending stiffness matrices are computed using matrices $\mathbf{B}'_m$ and $\mathbf{B}'_b$ of Eqs.(8.31). The transverse shear stiffness matrix is computed using the substitute transverse shear strain matrix $\bar{\mathbf{B}}'_s$ following the procedure explained for the QLLL plate element and shown in Figure 12.50.

The global stiffness matrix for the element is directly found by transforming the local generalized strain matrices to global axes as shown in Eqs.(8.47) and (8.48).

Figures 12.58 and 12.59 respectively show the steps for computing of the membrane and bending strain matrices and the substitute transverse shear strain matrix *in global axes*.

12.18.3 Computation of local resultant stresses

The computation of the local resultant stresses at the 2×2 Gauss points within an element follows the procedure explained in Eqs.(8.49).

The extrapolation of the Gauss point values to the nodes can be performed using Eq.(12.1). However, the nodal averaging of the local resultant stresses is not straightfogrward as the local axes of adjacent elements are not necessarily the same. An alternative is to transform the nodal resultant stresses contributed by each element to global axes, perform the nodal averaging and then transform back to a nodal coordinate system.

Figure 12.60 shows the nodal averaging of the local stresses assuming a negligible change of the local axes between adjacent elements.

Fig. 12.57 QLLL flat shell element. Definition of local axes

12.18.4 Examples

12.18.4.1 Clamped hyperbolic shell under uniform loading

We present results for the analysis of a thick hyperbolic shell clamped at the four edges under a vertical uniformly distributed load. The geometry, material properties and the load are shown in Figure 12.61a.

The problem has been solved with different meshes of QLLL flat shell elements. No advantage of the symmetry of the problem has been taken. Figures 12.61b show snapshots of the data input menus. The general struc-

```
function [bmat_b,bmat_m,bmat_s,area]=B_mat_Shell_CLLL(xgs,ygs,x,y,Te)

dxNl(1) = (-1+ygs)/4;  dyNl(1) = (-1+xgs)/4;
dxNl(2) = ( 1-ygs)/4;  dyNl(2) = (-1-xgs)/4;
dxNl(3) = ( 1+ygs)/4;  dyNl(3) = ( 1+xgs)/4;
dxNl(4) = (-1-ygs)/4;  dyNl(4) = ( 1-xgs)/4;

xjacm(1,1) = x(1)*dxNl(1) + x(2)*dxNl(2) + x(3)*dxNl(3) + x(4)*dxNl(4);
xjacm(1,2) = y(1)*dxNl(1) + y(2)*dxNl(2) + y(3)*dxNl(3) + y(4)*dxNl(4);
xjacm(2,1) = x(1)*dyNl(1) + x(2)*dyNl(2) + x(3)*dyNl(3) + x(4)*dyNl(4);
xjacm(2,2) = y(1)*dyNl(1) + y(2)*dyNl(2) + y(3)*dyNl(3) + y(4)*dyNl(4);

xjaci = inv(xjacm);

area = abs(xjacm(1,1)*xjacm(2,2) - xjacm(2,1)*xjacm(1,2));

dxN(1) = xjaci(1,1)*dxNl(1)+xjaci(1,2)*dyNl(1);
dxN(2) = xjaci(1,1)*dxNl(2)+xjaci(1,2)*dyNl(2);
dxN(3) = xjaci(1,1)*dxNl(3)+xjaci(1,2)*dyNl(3);
dxN(4) = xjaci(1,1)*dxNl(4)+xjaci(1,2)*dyNl(4);
dyN(1) = xjaci(2,1)*dxNl(1)+xjaci(2,2)*dyNl(1);
dyN(2) = xjaci(2,1)*dxNl(2)+xjaci(2,2)*dyNl(2);
dyN(3) = xjaci(2,1)*dxNl(3)+xjaci(2,2)*dyNl(3);
dyN(4) = xjaci(2,1)*dxNl(4)+xjaci(2,2)*dyNl(4);

bmat_b1  = [ 0, 0, 0,-dxN(1),     0  ;
             0, 0, 0,      0,-dyN(1) ;
             0, 0, 0,-dyN(1),-dxN(1)];
bmat_b2  = [ 0, 0, 0,-dxN(2),     0  ;
             0, 0, 0,      0,-dyN(2) ;
             0, 0, 0,-dyN(2),-dxN(2)];
bmat_b3  = [ 0, 0, 0,-dxN(3),     0  ;
             0, 0, 0,      0,-dyN(3) ;
             0, 0, 0,-dyN(3),-dxN(3)];
bmat_b4  = [ 0, 0, 0,-dxN(4),     0  ;
             0, 0, 0,      0,-dyN(4) ;
             0, 0, 0,-dyN(4),-dxN(4)];

bmat_b = [bmat_b1,bmat_b2,bmat_b3,bmat_b4];

bmat_m1d  = [ dxN(1),     0,  0 ;
                    0, dyN(1), 0 ;
              dyN(1), dxN(1), 0];
bmat_m2d  = [ dxN(2),     0,  0 ;
                    0, dyN(2), 0 ;
              dyN(2), dxN(2), 0];
bmat_m3d  = [ dxN(3),     0,  0 ;
                    0, dyN(3), 0 ;
              dyN(3), dxN(3), 0];
bmat_m4d  = [ dxN(4),     0,  0 ;
                    0, dyN(4), 0 ;
              dyN(4), dxN(4), 0];

bmat_mir  = [ 0, 0 ;
              0, 0 ;
              0, 0];

bmat_m1 = [bmat_m1d*Te,bmat_mir];
bmat_m2 = [bmat_m2d*Te,bmat_mir];
bmat_m3 = [bmat_m3d*Te,bmat_mir];
bmat_m4 = [bmat_m4d*Te,bmat_mir];

bmat_m = [bmat_m1,bmat_m2,bmat_m3,bmat_m4];
```

Shape function derivatives in local coordinate system

Isoparametric transformation

Bending strain matrix

Membrane strain matrix

Fig. 12.58 QLLL flat shell element. Computation of membrane and bending strain matrices in global axes

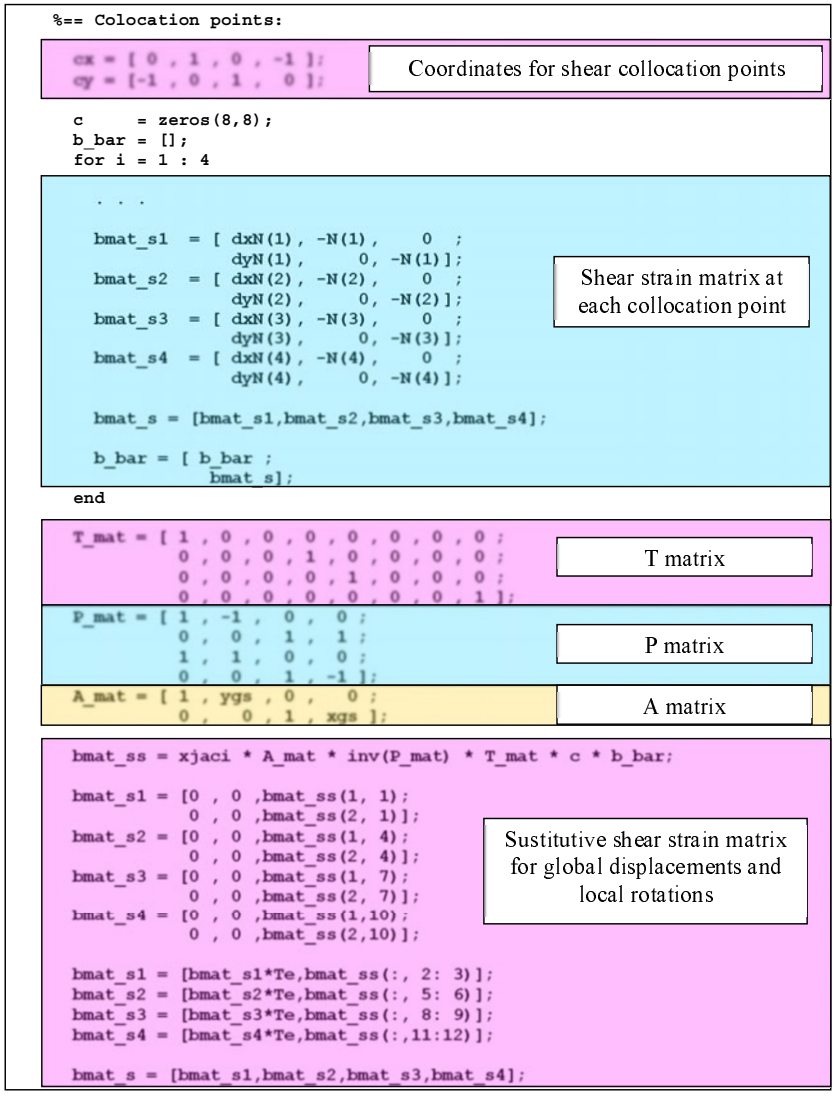


Fig. 12.59 QLLL flat shell element. Computation of substitute transverse shear strain matrix in global axes

ture of the input data file for the 4×4 element mesh is shown in Figure 12.62.

The reference solutions found with a mesh of 100×100 QLLL flat shell elements yield a central deflection of $w_c = -0.0245$, a bending moment at the center of $M_{x'} = 0.39$ and a shear force and the center of the clamped edge of $Q_{z'} = 0.15$ (units in International System).

```
% Shape Function matrix for extrapolation of resultant stresses to nodes
aa = 1 + sqrt(3);
bb = 1 - sqrt(3);
mstres = [ aa*aa , aa*bb , bb*bb , aa*bb ;
           bb*aa , aa*aa , aa*bb , bb*bb ;
           bb*bb , aa*bb , aa*aa , bb*aa ;
           aa*bb , bb*bb , bb*aa , aa*aa ]/4;
% Resultant stresses at Gauss points
for igs = 1 : 4
  x = gauss_x(igaus);
  y = gauss_y(igaus);
  [bmat_b,bmat_m,bmat_s,area]=B_mat_Shell_QLLL_v1_1(x,y,coor_x,coor_y,Te);

  Str1=D_matb*bmat_b*transpose(u_elem);
  Str2=D_matm*bmat_m*transpose(u_elem);
  Str3=D_mats*bmat_s*transpose(u_elem);

  StrMx(igaus)  = Str1(1);
  StrMy(igaus)  = Str1(2);
  StrMxy(igaus) = Str1(3);
  StrNx(igaus)  = Str2(1);
  StrNy(igaus)  = Str2(2);
  StrNxy(igaus) = Str2(3);
  StrQx(igaus) = Str3(1);
  StrQy(igaus) = Str3(2);

end
% Resultant stresses at nodes
Str1 = mstres * transpose(StrMx) ;
Strnod(lnods(1:4),1) = Strnod(lnods(1:4),1)+ Str1(1:4);
Str1 = mstres * transpose(StrMy) ;
Strnod(lnods(1:4),2) = Strnod(lnods(1:4),2)+ Str1(1:4);
Str1 = mstres * transpose(StrMxy) ;
Strnod(lnods(1:4),3) = Strnod(lnods(1:4),3)+ Str1(1:4);
Str2 = mstres * transpose(StrNx) ;
Strnod(lnods(1:4),4) = Strnod(lnods(1:4),4)+ Str2(1:4);
Str2 = mstres * transpose(StrNy) ;
Strnod(lnods(1:4),5) = Strnod(lnods(1:4),5)+ Str2(1:4);
Str2 = mstres * transpose(StrNxy) ;
Strnod(lnods(1:4),6) = Strnod(lnods(1:4),6)+ Str2(1:4);
Str3 = mstres * transpose(StrQx) ;
Strnod(lnods(1:4),7) = Strnod(lnods(1:4),7)+ Str3(1:4);
Str3 = mstres * transpose(StrQy) ;
Strnod(lnods(1:4),8) = Strnod(lnods(1:4),8)+ Str3(1:4);
Strnod(lnods(1:4),9) = Strnod(lnods(1:4),9)+ 1;

for i = 1 : npnod
     Strnod(i,1:5) = Strnod(i,1:5)/Strnod(i,6);
end
```

Nodal averaging of resultant stresses

Fig. 12.60 Computation of local resultant stresses at the Gauss points for the QLLL flat shell element

Table 12.10 shows the values for these three results for different meshes of QLLL flat shell elements.

Figure 12.63 shows contours of the vertical deflection and the bending moment $M_{x'}$ for the 16×16 mesh.

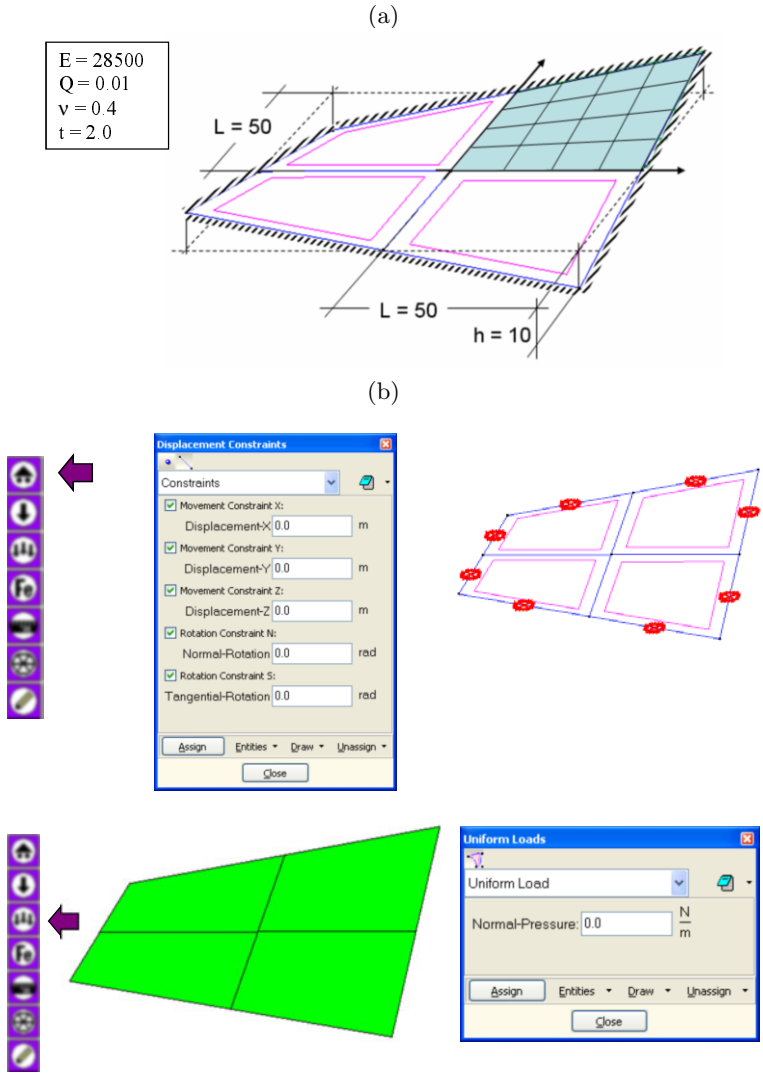


Fig. 12.61 Clamped hyperbolic shell under uniform distributed load. (a) Geometry, material properties and load. (b) Snapshots of data input menus

12.18.5 Scordelis roof

We present results for the analysis of a cylindrical shell with end diaphragms under uniform load (the so called *Scordelis roof*). The general description of the problem is shown in Figures 12.64 and 8.34.

Table 12.11 shows the results for the vertical displacement of point B using different meshes of QLLL flat shell elements.

```
%=====================================================================
% MAT-fem_Shells 1.0 - MAT-fem is a learning tool for undestanding
%                      the Finite Element Method with MATLAB and GiD
%=====================================================================
% PROBLEM TITLE = Clamped hyperbolic shell under uniform load

%  Material Properties
  young  =   2.850000000e+04 ;
  poiss  =   4.000000000e-01 ;
  denss  =   0.000000000e+00 ;
  thick  =   2.000000000e+00 ;
%
% Coordinates
global coordinates
coordinates = [
   0.000000000e+00   ,   1.000000000e+02   ,  -1.000000000e+01  ;
   5.000000000e+01   ,   1.000000000e+02   ,   0.000000000e+00  ;
....
   1.000000000e+02   ,   5.000000000e+01   ,   0.000000000e+00  ;
   1.000000000e+02   ,   0.000000000e+00   ,  -1.000000000e+01  ] ; %
% Elements
global elements
elements = [
      2   ,      6   ,      8   ,      4   ;
      3   ,      1   ,      2   ,      4   ;
      7   ,      5   ,      3   ,      4   ;
      8   ,      9   ,      7   ,      4   ] ; %
% Fixed Nodes
fixnodes = [
      1  , 1 ,   0.000000000e+00  ;
      1  , 2 ,   0.000000000e+00  ;
      1  , 3 ,   0.000000000e+00  ;
      1  , 4 ,   0.000000000e+00  ;
      1  , 5 ,   0.000000000e+00  ;
...
      9  , 1 ,   0.000000000e+00  ;
      9  , 2 ,   0.000000000e+00  ;
      9  , 3 ,   0.000000000e+00  ;
      9  , 4 ,   0.000000000e+00  ;
      9  , 5 ,   0.000000000e+00  ] ;
%
% Point loads
%
pointload = [ ] ;
%
% Side loadsss
%
uniload = sparse ( 4 , 1 );
uniload (      1  ) =   1.000000000e-02  ;
uniload (      2  ) =   1.000000000e-02  ;
uniload (      3  ) =   1.000000000e-02  ;
uniload (      4  ) =   1.000000000e-02  ;
```

Fig. 12.62 Clamped hyperbolic shell under uniform load. Example of data input for a 8-element mesh

Figure 12.67 shows results of $N_{y'}$, $M_{x'}$ and $Q_{x'}$ along the line CB obtained with a mesh of 32×32 QLLL flat shell elements. Figure 8.34 shows results for the same problem obtained with other flat shell elements.

Clamped hyperbolic shell				
QLLL Mesh	Nodes	w_c	$M_{x'}$	$Q_{y'}$
2 × 2	9	-1,068E-03	0,000	0.112
4 × 4	25	-1,751E-03	0,050	0.109
8 × 8	81	-1,266E-03	-0,093	-0.041
16 × 16	289	-1,655E-03	-0,108	-0.043
32 × 32	1089	-2,018E-03	-0,083	-0.047
64 × 64	4225	-2,212E-03	-0,065	-0.052
Reference [CB]		-2,451E-02		

Table 12.10 Clamped hyperbolic shell under uniform load. Results for deflection w_c and bending moment $M_{x'}$ at the center and shear force $Q_{y'}$ at the center of the clamped edge using different meshes of QLLL flat shell elements

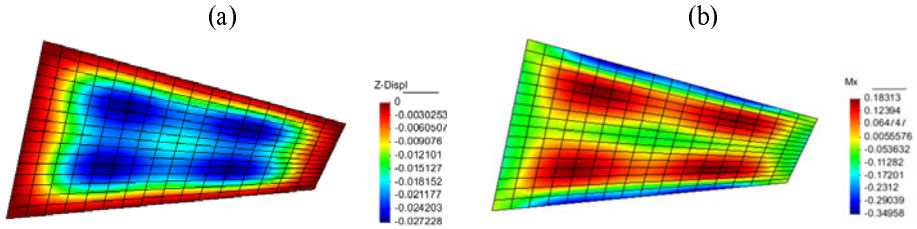


Fig. 12.63 Clamped hyperbolic shell. Contours of vertical deflection (a) and $M_{x'}$ (b)

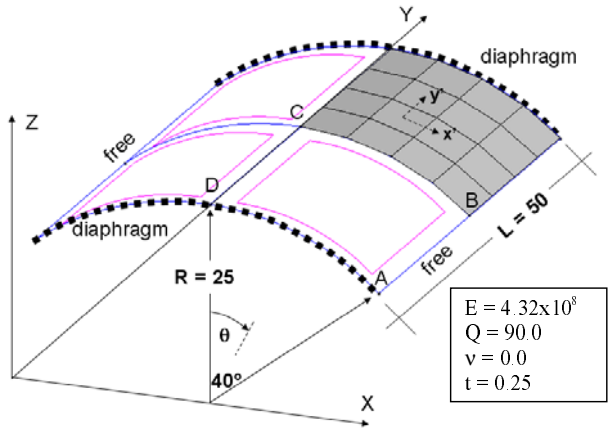


Fig. 12.64 Scordelis roof. Geometry, material properties and boundary conditions

12.19 2-NODED REISSNER-MINDLIN TRONCOCONICAL SHELL ELEMENT

This element was studied in Sections 9.4 and 9.6.1.

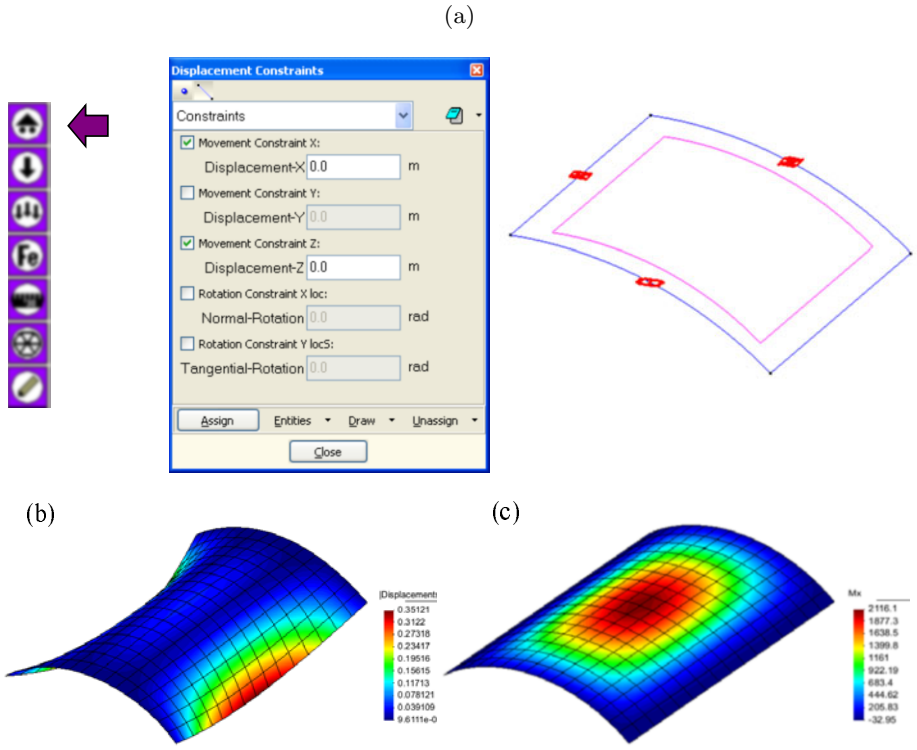


Fig. 12.65 Scordelis roof. (a) Boundary conditions. (b) Displacement contour. (c) Bending moment $M_{x'}$ contour for a 8×8 mesh of QLLL flat shell elements

Scordelis roof				
QLLL elements	Mesh	Nodes	w_B	% Error
4	2×2	9	-0.3573	18.15
16	4×4	25	-0.2832	-6.34
64	8×8	81	-0.2943	-2.67
256	16×16	289	-0.3005	-0.63
1024	32×32	1089	-0.3026	-0.07
Reference [BSC+,MH2]			-0.3024	

Table 12.11 Scordelis roof shell under uniform load. Results for the vertical deflection of point B for different meshes of QLLL flat shell elements

12.19.1 Generalized constitutive matrix

Figure 12.67 shows the membrane (`D_matm`), bending (`D_matb`) and transverse shear (`D_mats`) generalized constitutive matrices of Eq.(9.39) and (9.40).

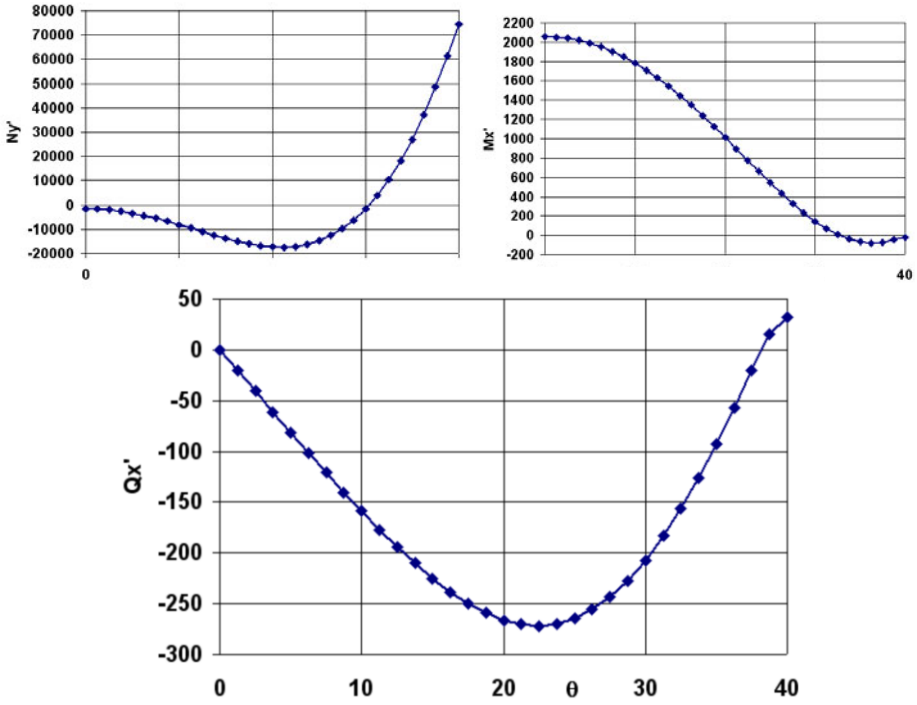


Fig. 12.66 Scordelis roof. Distribution of $N_{y'}$, $M_{x'}$ and $Q_{x'}$ along line CB for a mesh of 32×32 QLLL flat shell elements (see also Figure 8.34)

```
aux1 = thick*young/(1-poiss^2);
aux2 = poiss*aux1;
aux3 = thick*young/2/(1+poiss);
aux4 = (5/6)*thick*young/2/(1+poiss);

D_matm  = [aux1,aux2;
           aux2,aux1];

D_matb = D_matm*(thick^2/12);

D_mats = [aux4];

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

Fig. 12.67 2-noded RM troncoconical shell element. Membrane, bending and transverse shear generalized constitutive matrices

12.19.2 Stiffness matrix and equivalent nodal force vector

Figure 12.68 shows the computation of the transformation matrix **L** relating the global and the local nodal displacements (Eq.(9.14)).

The computation of the stiffness matrix and the equivalent nodal force vector for a uniform pressure is performed explicitly using a single inte-

```
function  Te = Rotation_system_RS (cxy)

  x = cxy(2,1) - cxy(1,1);
  y = cxy(2,2) - cxy(1,2);
  len = sqrt(x^2+y^2);

  co = x / len ;
  se = y / len ;

  Te = [  co, se, 0 ;
         -se, co, 0 ;
           0,  0, 1];
```

Fig. 12.68 2-noded troncoconical RM shell element. Displacement transformation matrix

```
cxy(1:nnode,:) = coordinates(lnods(1:nnode),:); % Element coordinates

Te = Rotation_system_RS(cxy);

r = (cxy(1,1)+cxy(2,1))/2; % Radius of the element

[bmat_b,bmat_s,bmat_m,len] = B_mat_Rev_Shell(cxy,Te);

K_b  = transpose(bmat_b)*D_matb*bmat_b*2*pi*r*len;
K_s  = transpose(bmat_s)*D_mats*bmat_s*2*pi*r*len;
K_m  = transpose(bmat_m)*D_matm*bmat_m*2*pi*r*len;

K_elem  = K_elem + K_f + K_s + K_m;

c1 = 2*cxy(1,1)+  cxy(2,1);
c2 =   cxy(1,1)+2*cxy(2,1);

fx1 = uniload(ielem)* Te(2,1)*pi*len*c1/3;
fx2 = uniload(ielem)* Te(2,1)*pi*len*c2/3;

fy1 = (uniload(ielem) * Te(1,1) - denss*thick)*pi*len*c1/3;
fy2 = (uniload(ielem) * Te(1,1) - denss*thick)*pi*len*c2/3;

ElemFor =  [fx1,fy1,0,fx2,fy2,0];
```

Stiffness matrix using 1 Gauss point

Equivalent nodal force vector

Fig. 12.69 2-noded troncoconical RM shell element. Computation of stiffness matrix and equivalent nodal force vector

gration point, following Eqs.(9.73) and (9.74). The computation steps are shown in Figure 12.69. The membrane (`K_m`), bending (`K_b`) and transverse shear stiffness (`K_s`) matrices are computed separately and then added together in `K_elem`.

Figure 12.70 shows the subroutine for computing the membrane, bending and transverse shear generalized strain matrices.

12.19.3 Resultant stresses

The local resultant stresses are computed at the element center and then they are extrapolated to the nodes.

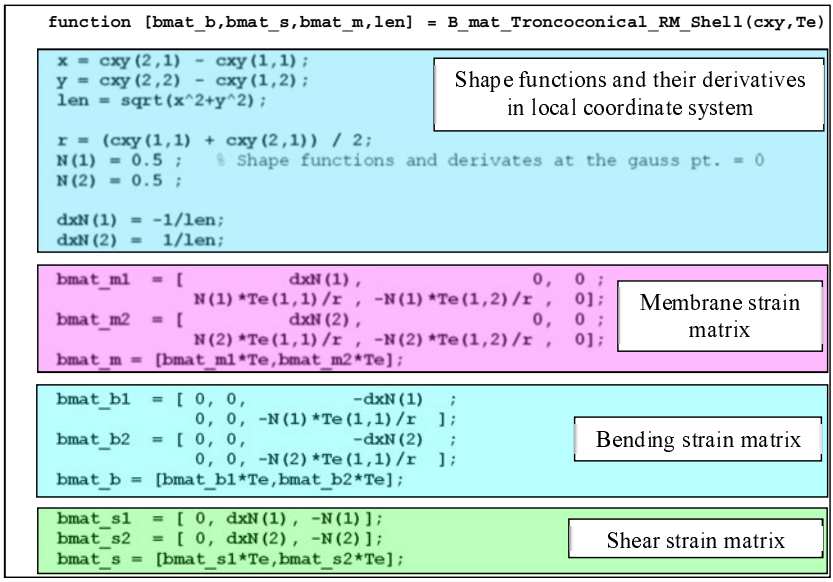


Fig. 12.70 2-noded troncoconical RM shell element. Computation of $\mathbf{B}_m$, $\mathbf{B}_b$ and $\mathbf{B}_s$

```
[bmat_b,bmat_s,bmat_m,len] = B_mat_Troncoconical_RM_Shell(cxy,Te);

% Resultant stresses at element mid-point
  Str1=D_matb*bmat_b*u_elem;
  Str2=D_mats*bmat_s*u_elem;
  Str3=D_matm*bmat_m*u_elem;
%
  Mx(ielem) =Str1(1);
  Mf(ielem) =Str1(2);
  Qz(ielem) =Str2(1);
  Nx(ielem) =Str3(1);
  Nf(ielem) =Str3(2);
```

Fig. 12.71 2-noded troncoconical RM shell element. Computation of resultant stresses at the element mid-point

Figure 12.71 shows the steps for computing the bending moment $M_{x'}$ and $M_{y'}$ (stored in `Str1`), the transverse shear force Q (stored in `Str2`) and the axial forces $N_{x'}$ and $N_{y'}$ (stored in `Str3`).

12.19.4 Example. Thin spherical dome under uniform external pressure

Figure 12.72 shows the geometry of the dome, the material properties and the external pressure values. Further details are given in Section 9.7.1.

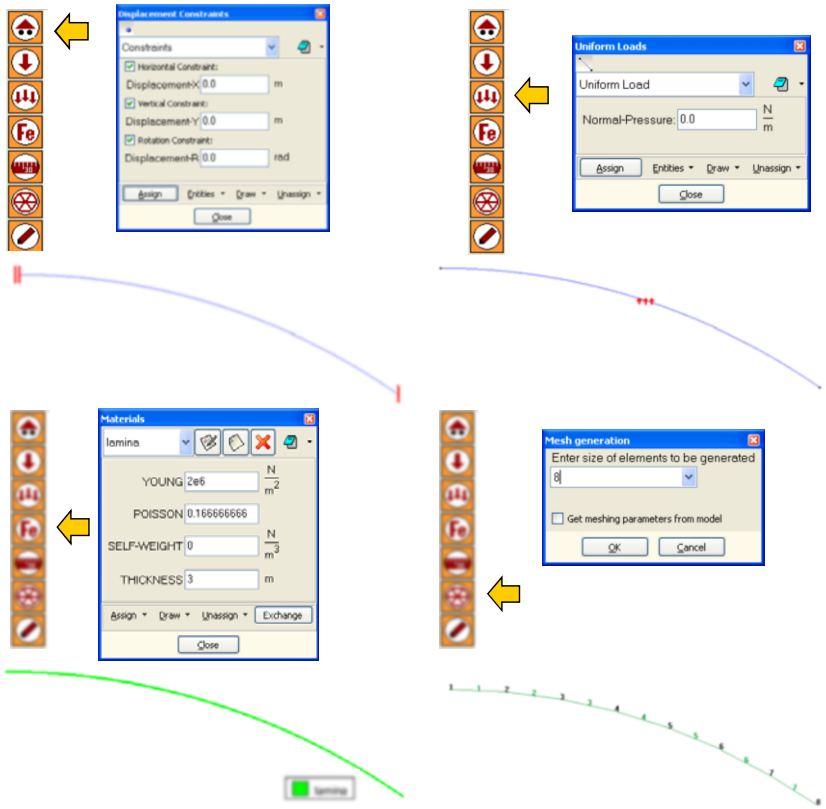
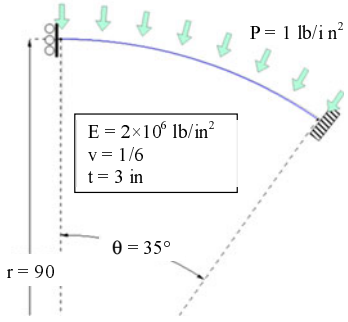


Fig. 12.72 2-noded troncoconical shell element. Data input menus for boundary conditions, pressure load, material properties and mesh



Spherical dome under external pressure				
Elements	Nodes	w_c	$M_{x'}$	$N_{y'}$
2	3	-1,444E-03	8,183	18,920
4	5	-9,137E-04	14,648	12,662
8	9	-8,384E-04	22,354	8,493
16	17	-8,189E-04	28,351	6,681
32	33	-8,137E-04	32,109	6,150
64	65	-8,124E-04	34,204	6,051
128	129	-8,121E-04	35,306	6,024
Reference	[Del,TW]	-8,121E-04	39,00	5,00

Table 12.12 Spherical dome under uniform external pressure analyzed with 2-noded troncoconical RM shell elements. Deflection at the symmetry node and radial bending moment and circumferential axial force at the center of the element adjacent to the clamped end for different meshes (see also Figure 8.16)

```
%======================================================================
% MAT-fem_RevShells 1.0  - MAT-fem is a learning tool for undestanding
%                          the Finite Element Method with MATLAB and GiD
%======================================================================
% PROBLEM TITLE = Thin spherical dome under uniform external pressure
%                 analyzed with 28 2-noded troncoconical RM shell elements
%
%  Material Properties
  young   =   2.000000000e+06 ;
  poiss   =   1.666666660e-01 ;
  denss   =   0.000000000e+00 ;
  thikness=   3.000000000e+00 ;
%
% Coordinates
global coordinates
coordinates = [
   5.162187927e+01   ,   7.372368399e+01  ;
   4.994083848e+01   ,   7.487264288e+01  ;
...
   0.000000000e+00   ,   9.000000000e+01  ] ;
%
% Elements
global elements
elements = [
     28   ,     27   ;
     27   ,     26   ;
...
      2   ,      1   ] ;
%
% Fixed nodes
fixnodes = [
      1  , 1 ,   0.000000000e+00  ;
      1  , 2 ,   0.000000000e+00  ;
      1  , 3 ,   0.000000000e+00  ;
     28  , 1 ,   0.000000000e+00  ;
     28  , 3 ,   0.000000000e+00  ] ;
%
% Point loads
pointload = [ ] ;
%
% Side loads
uniload = sparse ( 27 );
uniload (      1  ) =  -1.000000000e+00  ;
uniload (      2  ) =  -1.000000000e+00  ;
..
uniload (     27  ) =  -1.000000000e+00  ;
```

Fig. 12.73 Data input file for analysis of a thin spherical dome under uniform external pressure using 28 2-noded troncoconical RM elements

The reference solution has been obtained with a mesh of 60000 4-noded cubic troncoconical elements giving a deflection at the center of $w_c = 81211 \times 10^{-4}$ in, a radial bending moment at the clamped end $M_{x'} = 39$ lb×in/in and a circumferential axial force also at the clamped end $N_{y'} = 5.0$ lb/in. Other solution to this problem can be found in [Del,TW].

Figure 12.72 shows some of the data input menus corresponding to the boundary conditions, the pressure load, the material properties and the definition of the mesh.

The data input file for a 28 element mesh is shown in Figure 12.73.

Table 12.12 shows the convergence of the deflection at the symmetry node and the radial bending moment and the circumferential force at the center of the element adjacent to the clamped end for different meshes.

12.20 FINAL REMARKS

We have presented the basic concepts and the structure of the MAT-fem code environment for analysis of beams, plates and shells using some of the elements studied in the book.

Each element has been programmed as a separate code to facilitate the follow-up of the programming steps and the use of the code.

Each of the codes presented in this chapter, and other codes for several of the elements studied in the book, together with examples of applications, can be downloaded from www.cimne.com/MAT-fem.

For general questions about the use of MAT-fem please contact Dr. Francisco Zárate at zarate@cimne.upc.edu.