Computer Assisted Mechanics and Engineering Sciences, **2**: 51–71, 1995.

Extended finite element homework

Miloslav Okrouhlík
Institute of Thermomechanics, Academy of Sciences of the Czech Republic
Dolejškova 5, 182 00 Prague 8, The Czech Republic

(Received September 1, 1994)

Today the solution of mechanical problems in engineering practice is often routinely carried out by means of finite element packages. These packages are powerful and efficient and are able to solve many complicated problems of technical practice on a routine basis. The packages are more and more automated. In some cases, the user is even "deprived" of solving meshing problems — the so called meshless finite element approach is being advocated. In other cases the packages take care of the correct determination of time step in transient problems. These packages offer a lot of options to choose from; the options themselves are described in particular manuals to a variable extent of details. As the Murphy law states, however, the manuals are as a rule read only if nothing else helps. It is thus worthwhile to recall some of the essentials from the finite element theory, show pitfalls which should be avoided and to present modern programming tools which help a lot in the derivation of necessary relations and in subsequent understanding of the matter. The behaviour of a rectangular membrane element and that of some finite element packages when solving simple problems will be shown in this paper with the intention to answer the question whether a modern engineer is supposed to know the theoretical details of the finite element theory and the essentials of programming.

1. Introduction

In this paper, the properties of a simple finite membrane element with eight degrees of freedom are discussed, its mass and stiffness matrices are derived analytically with the use of the Reduce language. Then, Fortran procedures for calculating the same mass and stiffness matrices numerically by means of Gauss quadrature will be presented. Also, the static and dynamic behaviour of a simple problem, namely the bending and eigenfrequencies and eigenmodes of a cantilever beam, will be analysed with a powerful Matlab matrix processor. The dispersive behaviour of this type of element will be mentioned as well. Finally, the results will be compared with those obtained with sophisticated finite element tools like Ansys, Cosmos, Dyna, Systus and PMD.

The widespread distribution of personal computers and their universal computing capability allows many powerful programming tools, like algebra manipulating languages (Reduce, Maple, Mathematica, Derive, etc.), matrix processors (Matlab, Mathematica, Gauss, etc.) as well as classical compilers (Fortran, Pascal, C and others) to be used in a unified way for a wide class of solutions of theoretical and practical problems. This is substantially enhanced by the Windows operating system which secures easy portability of results in various forms between particular applications.

2. Derivation of stiffness and mass matrices with the use of generalized coordinates

The employment of the displacement formulation of the finite element method leads to the approximation of displacements by a polynomial function in the form

$$\mathbf{u} = \mathbf{U}\mathbf{c}\,, \tag{1}$$

where $\mathbf{u}$ is a column vector containing the components of displacements approximations. Lagrangian interpolation formulas are used in so called Lagrangian elements. $\mathbf{U}$ is a matrix of approximation

functions in the form of polynomials of spatial coordinates and $\mathbf{c}$ is a column vector of so far unknown constants, sometimes called generalized coordinates. The generalized coordinates are constructed on the condition that the assumed approximation of displacements, given in Eq. (1), must be valid in nodes as well. The substitution of the nodal coordinates for all the element nodes into Eq. (1) yields

$$\mathbf{q} = \mathbf{S}\mathbf{c}\,, \tag{2}$$

where $\mathbf{q}$ is a column vector of nodal displacements and $\mathbf{S}$ is a matrix of nodal coordinates which is regular for non-vanishing elements and can easily be inverted, thus allowing the generalized coordinates to be expressed in the form

$$\mathbf{c} = \mathbf{S}^{-1}\mathbf{q}\,. \tag{3}$$

After substituting the last equation into Eq. (1) we receive the approximation of displacements

$$\mathbf{u} = \mathbf{A}\mathbf{q}\,, \tag{4}$$

where

$$\mathbf{A} = \mathbf{U}\mathbf{S}^{-1} \tag{5}$$

is a matrix of shape functions.

Approximation of strains follows from the assumption (1) and from Cauchy kinematic relations $\boldsymbol{\varepsilon} = \mathbf{f}(\mathbf{u})$. It can be expressed in the form

$$\boldsymbol{\varepsilon} = \mathbf{F}\mathbf{c}\,. \tag{6}$$

By substituting (3) into (6) we finally obtain

$$\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{q}\,, \tag{7}$$

where

$$\mathbf{B} = \mathbf{F}\mathbf{S}^{-1} \tag{8}$$

is an operator relating approximation of strains to generalized nodal displacements.

In linear cases the stiffness and mass matrices can be derived in the form

$$\mathbf{K} = \int_V \mathbf{B}^T\mathbf{E}\mathbf{B}\,\mathrm{d}V\,, \tag{9}$$

$$\mathbf{M} = \int_V \rho\mathbf{A}^T\mathbf{A}\,\mathrm{d}V\,, \tag{10}$$

where ρ is the material density, $\mathbf{E}$ is a matrix of elastic moduli appearing in the generalized Hooke law and the integration is carried out with respect to the volume of the undeformed element. More details can be found in standard finite element textbooks, as in [1, 2, 8, 25], etc.

3. Analytical derivation of stiffness and mass matrices for a rectangular membrane element for 2D plane stress problems

The dimensions of the considered element, numbering of nodes and its degrees of freedom are depicted in Fig. 1. The thickness of the element is h. The approximation of displacements generally requires a polynomial which has the same number of free constants (generalized coordinates) as the number of degrees of freedom (displacements) in each direction. In the considered case an incomplete polynomial of the second order in the form of a bilinear function (securing thus spatial isotropy requirements) would be sufficient.

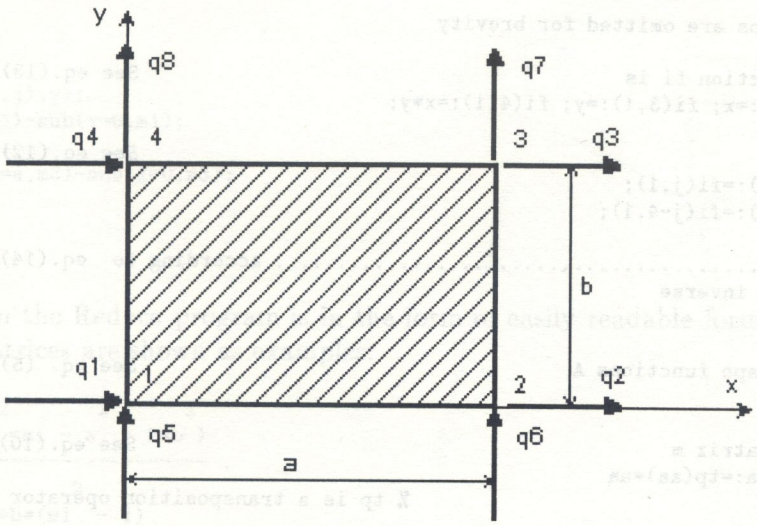


Fig. 1

The approximation of displacements has the form of Eq. (1), where

$$\mathbf{u} = \{u_x, u_y\} = \mathbf{U}\mathbf{c} \tag{11}$$

and

$$\mathbf{U} = \begin{bmatrix} \boldsymbol{\Phi}^T & \mathbf{0}^T \\ \mathbf{0}^T & \boldsymbol{\Phi}^T \end{bmatrix} \tag{12}$$

with

$$\boldsymbol{\Phi}^T = \{1 \;\; x \;\; y \;\; xy\}\,, \qquad \mathbf{0}^T = \{0\,0\,0\,0\} \quad \text{and} \quad \mathbf{c}^T = \{c_1 \;\ldots\; c_8\}\,. \tag{13}$$

By substituting the nodal coordinates into (12) we obtain

$$\mathbf{q} = \mathbf{S}\mathbf{c} = \begin{bmatrix} \bar{\mathbf{S}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{S}} \end{bmatrix} \mathbf{c}\,, \qquad \bar{\mathbf{S}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & a & 0 & 0 \\ 1 & a & b & ab \\ 1 & 0 & b & 0 \end{bmatrix}, \tag{14}$$

where $\mathbf{0}$ is a zero matrix four by four and $\mathbf{q}^T = \{q_1 \ldots q_8\}$ is a column vector of nodal displacements ordered in agreement with the numbering shown in Fig. 1.

The further process can be carried out by means of algebra manipulating languages. An excerpt from the Reduce program, from which all data management statements were omitted for brevity, is as follows. Since it is richly commented the author believes that it will be readable and acceptable even to nonprogramming readers.

```
%  A part of program written in REDUCE language
%  which serves to the derivation of mass and stiffness matrices
%  for a rectangular membrane elements with 8 dof
%  for 2D plane stress problems.
%
% Input parameters                  Output parameters
%
% a, b ..... element dimensions     m, k ..... mass and stiffness matrices
% h ........ element thickness
% ro ....... density
% mi ....... Poisson ratio
% em ....... Young modulus
%
```

```
% Matrix declarations are omitted for brevity
%
% Approximation function fi is                                    See eq.(13)
fi(1,1):=1; fi(2,1):=x; fi(3,1):=y; fi(4,1):=x*y;
%
% form U matrix                                                   See eq.(12)
for j:=1:4 do u(1,j):=fi(j,1);
for j:=5:8 do u(2,j):=fi(j-4,1);
%
% create S matrix ..................................... according to  eq.(14)
% and calculate its inverse
sinv:=s**(-1);
%
% form matrix of shape functions A                                See eq. (5)
aa:=u*sinv$
%
% derive the mass matrix m                                        See eq.(10)
% form integrand ata:=tp(aa)*aa
ata:=tp(aa)*aa;                              % tp is a transposition operator
%
% integrate with respect to x and y within the area a*b, h being constant
for i:=1:8 do
  for j:=1:8 do
    begin;
      m1:=int(ata(i,j),y);                   % integrate with respect to x
      m2:=sub(y=b,m1)-sub(y=0,m1);           % substitute integral limits
      m3:=int(m2,x);                         % integrate with respect to y
      m(i,j):=sub(x=a,m3)-sub(x=0,m3);       % substitute limits
    end;
m:=ro*h*m;                                   % this is mass matrix
%
% and now derive the stiffness matrix k
% derivatives of approximation function with respect to x and y
for i:=1:4 do
  begin;
    dfix(i,1):=df(fi(i,1),x);                % df is a derivative operator
    dfiy(i,1):=df(fi(i,1),y);
  end;
%
% matrix of elastic moduli for plane stress.
%                                                 See standard texts on mechanics
ee(1,1):=1;     ee(1,2):=mi;    ee(1,3):=0;
ee(2,1):=mi;    ee(2,2):=1;     ee(2,3):=0;
ee(3,1):=0;     ee(3,2):=0;     ee(3,3):=(1-mi)/2;
const:=em/(1-mi*mi);
% ee:=const*ee; this multiplication will be carried out later
%
% The F matrix reflects Cauchy strain - displacement relations and
% the assumed displacement approximation and in this case has the form
%
%         { e_xx }     [ 1 0 0 y 0 0 0 0 ]
% {e}  =  { e_yy }  =  [ 0 0 0 0 0 0 1 x ] {c}  =  [F] {c}          See Eq. (6)
%         { e_xy }     [ 0 0 1 x 1 0 0 y ]
%
% the first row of F matrix                                       See eq. (6)
for j:=1:4 do f(1,j):=dfix(j,1);
% the second row of F matrix
for j:=5:8 do f(2,j):=dfiy(j-4,1);
% the third row of F matrix
for j:=1:4 do f(3,j):=dfiy(j,1);
for j:=5:8 do f(3,j):=dfix(j-4,1);
% BB matrix.                                                      See eq. (8)
bb:=f*sinv;
% integrand of stiffness matrix
beb:=tp(bb)*ee*bb$
%
% integrate with respect to x and y within the area a*b, h being constant
for i:=1:8 do                                                   % See eq. (9)
```

```
  for j:=1:8 do
    begin;
      m1:=int(beb(i,j),y);
      m2:=sub(y=b,m1)-sub(y=0,m1);
      m3:=int(m2,x);
      k(i,j):=sub(x=a,m3)-sub(x=0,m3);
    end;
k:=const*h*k;
end;
```

The output from the Reduce program is in the form of easily readable formulas. The elements (1,1) from both matrices are shown as examples:

```
                    2      2      2
             em*h*(a *mi - a  - 2*b )
k(1,1) := ------------------------- ,
                         2
                 6*a*b*(mi  - 1)

              a*b*h*ro
m(1,1) := ---------- .
                9
```

This output, excellent for analytical considerations, is not, however, suitable for further computer processing. The Reduce language can transform this into a form which conforms to the Fortran standard. Then the above mentioned matrix elements are written starting from the seventh column with a continuation character appearing in the sixth column

```
      k(1,1)=(em*h*(a**2*mi-a**2-2.*b**2))/(6.*a*b*(mi**
     .        2-1.))

      m(1,1)=(a*b*h*ro)/9.
```

4. Numerical calculation of stiffness and mass matrices for an isoparametric quadrilateral membrane element for 2D plane stress problems with the use of Gauss quadrature

The dimensions of the considered element, numbering of nodes and its degrees of freedom are depicted in Fig. 2. The thickness of the element is h. In this case the procedure is more general, which allows a quadrilateral shape of the element to be considered instead of the rectangular one as it is in the analytical approach. The isoparametric approach requires (see [1]) that the approximation of coordinates and displacements is secured by identical interpolating functions, that is

$$\mathbf{x} = \left\{ \begin{array}{c} x(r,s) \\ y(r,s) \end{array} \right\} = \mathbf{A}\mathbf{p}\,, \tag{15}$$

$$\mathbf{u} = \left\{ \begin{array}{c} u(r,s) \\ v(r,s) \end{array} \right\} = \mathbf{A}\mathbf{q}\,. \tag{16}$$

The procedure we have followed when deriving element matrices by means of generalized coordinates is repeated here in the so called reference frame (r,s). Thus, we have

$$\mathbf{A} = \mathbf{U}\mathbf{S}^{-1}, \tag{17}$$

where

$$\mathbf{U} = \left[\begin{array}{cc} \boldsymbol{\Phi}^T & \mathbf{0}^T \\ \mathbf{0}^T & \boldsymbol{\Phi}^T \end{array} \right] \tag{18}$$

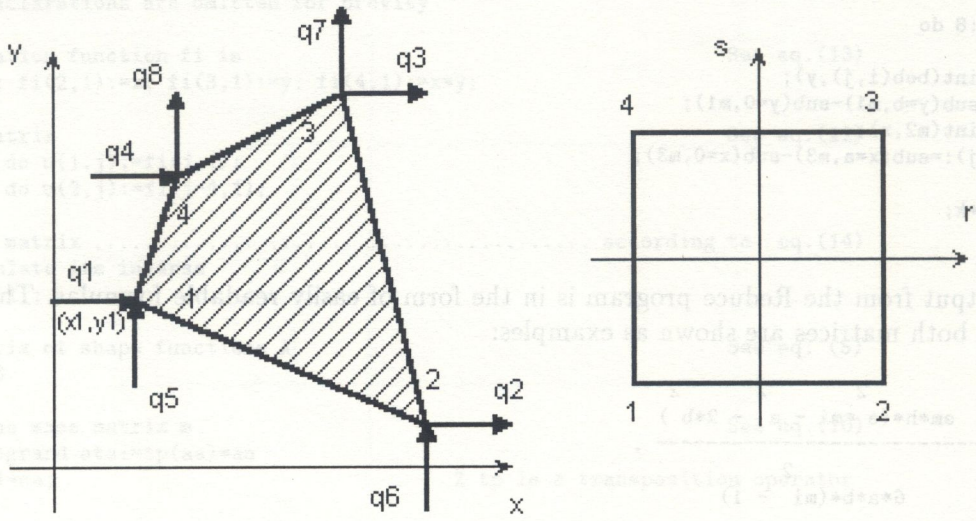


Fig. 2

and

$$\mathbf{\Phi}^T = \{1 \;\; r \;\; s \;\; rs\}\,. \tag{19}$$

The vector $\mathbf{q}$, a column vector of nodal displacements, has the components $q_1 \ldots q_8$ which are ordered in agreement with the numbering shown in Fig. 2, while $\mathbf{p}$, the vector of nodal coordinates, is

$$\mathbf{p}^T = \{x_1 \ldots x_4 \;\; y_1 \ldots y_4\}\,.$$

One of the advantages of the isoparametric approach is that the matrix $\mathbf{S}$, which is obtained in an identical manner as before, is always the same for this type of element, that is

$$\mathbf{S} = \begin{bmatrix} \bar{\mathbf{S}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{S}} \end{bmatrix} \qquad \text{with} \quad \bar{\mathbf{S}} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{20}$$

The inversion of this matrix is easy and can be done once for all by inverting the submatrices,

$$\bar{\mathbf{S}} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{21}$$

Evaluating the matrix $\mathbf{A}$ according to Eq. (17) and taking into account Eqs. (19)–(21) we get

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 & a_4 \end{bmatrix}, \tag{22}$$

where

$$\begin{aligned} a_1 &= a_1(r,s) = 0.25(1-r)(1-s), \\ a_2 &= a_2(r,s) = 0.25(1+r)(1-s), \\ a_3 &= a_3(r,s) = 0.25(1+r)(1+s), \\ a_4 &= a_4(r,s) = 0.25(1-r)(1+s). \end{aligned} \tag{23}$$

So we finally have the shape functions depending on variables r and s. For obtaining the $\mathbf{B}$ matrix, which relates the strains and displacements according to Eq. (7), partial derivatives of shape functions with respect to x and y are required. These can be found by employing the chain rule and the inverse of Jacobian matrix. Then the matrix $\mathbf{B}$ has the form

$$\mathbf{B} = \begin{bmatrix} \frac{\partial a_1}{\partial x} & \dots & \frac{\partial a_4}{\partial x} & \mathbf{0} \\ \mathbf{0} & & \frac{\partial a_1}{\partial y} & \dots & \frac{\partial a_4}{\partial y} \\ \frac{\partial a_1}{\partial y} & \dots & \frac{\partial a_4}{\partial y} & \frac{\partial a_1}{\partial x} & \dots & \frac{\partial a_4}{\partial x} \end{bmatrix} \tag{24}$$

with

$$\begin{bmatrix} \frac{\partial a_1}{\partial x} & \dots & \frac{\partial a_4}{\partial x} \\ \frac{\partial a_1}{\partial y} & \dots & \frac{\partial a_4}{\partial y} \end{bmatrix} = \mathbf{J}^{-1}(r,s) \begin{bmatrix} \frac{\partial a_1}{\partial r} & \dots & \frac{\partial a_4}{\partial r} \\ \frac{\partial a_1}{\partial s} & \dots & \frac{\partial a_4}{\partial s} \end{bmatrix}, \tag{25}$$

where

$$\mathbf{J}(r,s) = \begin{bmatrix} \sum \frac{\partial a_i}{\partial r} x_i & \sum \frac{\partial a_i}{\partial r} y_i \\ \sum \frac{\partial a_i}{\partial s} x_i & \sum \frac{\partial a_i}{\partial s} y_i \end{bmatrix}. \tag{26}$$

The index i is understood to take all the values in the range $1 \dots 4$ and (x_i, y_i) are the nodal coordinates of the considered element depicted in Fig. 2. Then the $\mathbf{J}$ matrix and its inverse, needed in Eq. (25), can easily be evaluated numerically for a generic point (r,s) in the reference plane.

Now, all the preparation steps for the evaluation of definition integrals, required for the calculation of stiffness and mass matrices have been done. The numerical calculation is as a rule carried out by the Gaussian quadrature. The integrals (9) and (10) are then approximated by

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{E} \mathbf{B} \, \mathrm{d}V = \int_{-1}^{+1} \int_{-1}^{+1} h \, \mathbf{B}^T(r,s) \, \mathbf{E} \, \mathbf{B}(r,s) \det \mathbf{J}(r,s) \, \mathrm{d}r \, \mathrm{d}s = \sum_{l=1}^{n} \sum_{m=1}^{n} \alpha_l \alpha_m \mathbf{I}_K(r_l, s_m), \tag{27}$$

$$\mathbf{M} = \int_V \rho \mathbf{A}^T \mathbf{A} \, \mathrm{d}V = \rho \int_{-1}^{+1} \int_{-1}^{+1} h \mathbf{A}^T(r,s) \, \mathbf{A}(r,s) \det \mathbf{J}(r,s) \, \mathrm{d}r \, \mathrm{d}s = \sum_{l=1}^{n} \sum_{m=1}^{n} \alpha_l \alpha_m \mathbf{I}_M(r_l, s_m), \tag{28}$$

where n is the order of Gauss quadrature. The integrands are evaluated at so called Gauss points (r_l, s_m) and then multiplied by corresponding weights (α_l, α_m) and summed up. The remaining symbols, denoting the discretized integrands, are

$$\mathbf{I}_K = h \, \mathbf{B}^T(r_l, s_m) \, \mathbf{E} \, \mathbf{B}(r_l, s_m) \det \mathbf{J}(r_l, s_m), \tag{29}$$

$$\mathbf{I}_M = \rho h \, \mathbf{A}^T(r_l, s_m) \, \mathbf{A}(r_l, s_m) \det \mathbf{J}(r_l, s_m). \tag{30}$$

A Fortran implementation of this procedure is listed in the Appendix.

It is known that the Gaussian quadrature of the n-th order is able to evaluate "exactly" the polynomial of the order $2n-1$, see [1]. The integrand of the mass matrix of the considered quadrilateral element, which is formed by a dyadic product of shape functions a_i according to (23), can be of the fourth order at the maximum, while that of the stiffness matrix, where the derivatives of shape functions appear, is of the second order only. Theoretically, the Gaussian quadrature of the third order would be sufficient for the numerical integration of the mass matrix, while for the stiffness matrix the second order would do. Practical computations show, however, that the second order quadrature with 16 significant digits gives values of both mass and stiffness matrices which differ from those calculated from analytically derived formulae by several least significant digits only.

Deriving the element matrices, we started from the requirement that the number of generalized coordinates (i.e. the number of free constants in the interpolation polynomial) must be the same as

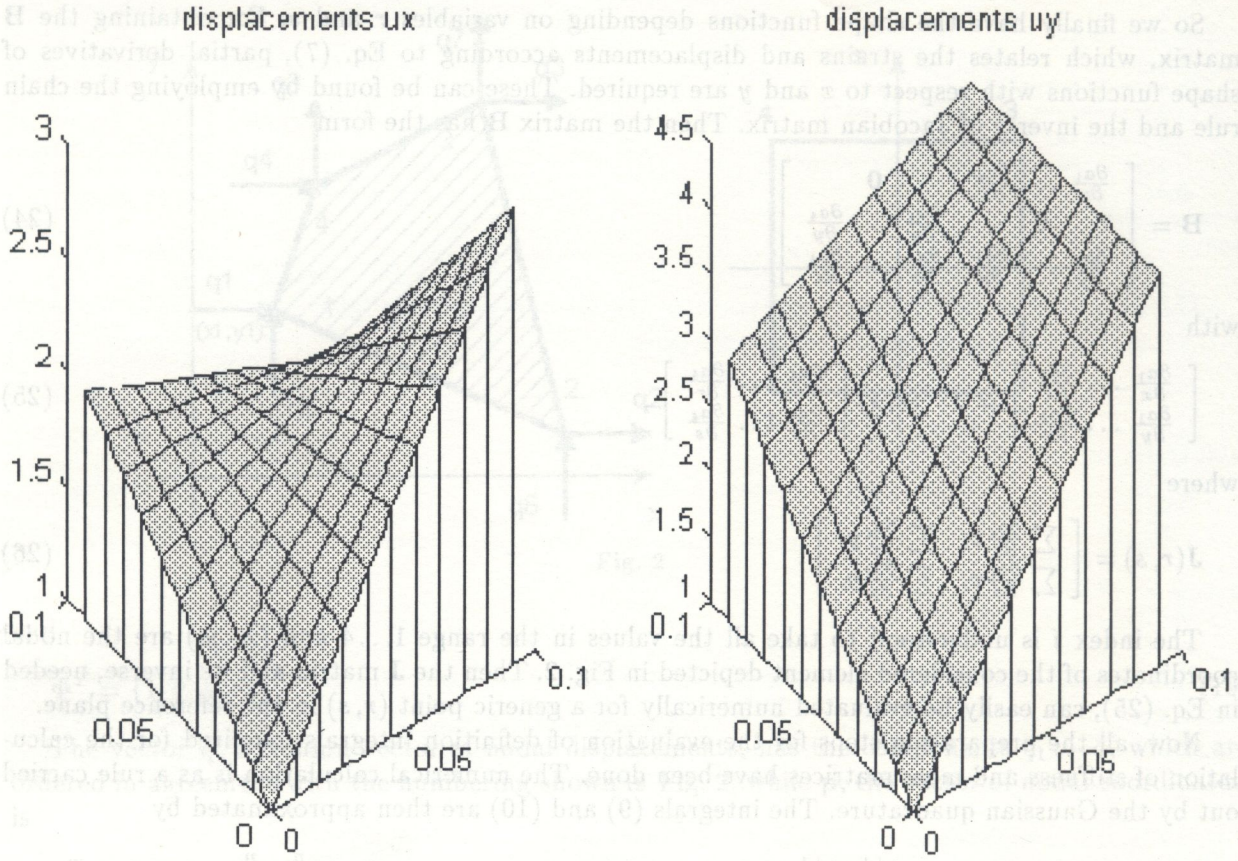


Fig. 3

the number of interpolated degrees of freedom. Since both directions are considered independently, the spatial isotropy must be secured by a suitable choice of polynomial terms in case an incomplete polynomial is used, so that no spatial directions are artificially preferred.

It is proved that the monotone convergence in finite element modelling requires that elements must be *compatible* and *complete*, see [16, 22].

The *condition of compatibility* requires that the approximation of displacements inside the element and at its boundaries, where the element is in touch with its neighbours, should be prescribed by a continuous function of space coordinates. The choice of polynomial functions satisfies this automatically inside the element.

For a four-node square element a possible displacement distribution is shown in Fig. 3. The distribution was calculated with the use of Eq. (4) with arbitrarily assumed nodal displacements $\mathbf{q}^T = \{1\ 2\ 1.5\ 3\ 1\ 3\ 4\ 4\}$.

The approximation of displacements, prescribed by the first relation of Eq. (13), gives the distribution of displacements which is described by a surface belonging to the family of so called "line surfaces" which are formed by two sets of straight lines (r=const and s=const). That is why the element with this shape function is sometimes called a bilinear element. From this follows that the displacement approximations above the element boundaries are formed by straight lines as well. Assembling local element matrices into global ones is based on the assumption that the nodal displacements are common to all the neighbouring elements. So along each segment of the boundary line of an element we have only two nodal points, there are two values of displacements defined at these nodes and consequently the line connecting the displacement values is unique and belongs to surfaces from the neighbouring elements. The neigbouring surfaces thus have the same displacements at the common boundary of neighbouring elements, which proves that the displacements approximation function is a continuous function of spatial coordinates.

The approximation of strains is not continuous at boundaries, since it depends on spatial deriva-

tives of displacement functions which are different, however, depending on the side of which element the partial derivative at a boundary point is calculated (because different surfaces could be generally expected on both sides of the boundary). From this follows that for this type of element the strain approximation is not a continuous function of spatial coordinates, which means that stresses, which are a linear function of strains in the linear cases, vary stepwise at boundaries, thus violating the local conditions of equilibrium.

The *condition of completeness* requires that the shape function should be capable of securing the rigid body motion of an element and the constant strain state.

Whether an element is able to represent the rigid body motion can be checked by verifying the condition that the prescribed rigid body motion does not evoke any external forces. The rigid body motion in x-direction can be realized by prescribing the nodal displacements $q_i = 1$, $i = 1 \ldots 4$ (see Fig. 1). In statics, the response of an element is governed by the equation $\mathbf{Kq} = \mathbf{F}$. For the prescribed rigid body motion the quantity $\mathbf{F}$ — the vector of external forces — must be identically equal to zero. Substitution of both conditions into the previous matrix equation yields the check relation $\sum_{j=1}^{4} k_{ij} = 0$ for $i = 1 \ldots 4$ to be satisfied for elements of stiffness matrix. A similar condition could be written for y-direction displacements. Putting this together means that the stiffness matrix of the considered element should have the sum of all elements in all rows equal to zero. Due to round-off errors this condition is not satisfied exactly. The relative error is, however, of the order of so called *machep* or *mach*ine *ep*silon, defined as a small floating point number which, added to one, creates a result that cannot be distinguished from it, see [10]. For a particular computer installation and the programming language used it could be easily calculated by means of a simple `while` loop. The following fragment of a Pascal program should clarify the idea.

```
program machep;
var eps   : real;
    i,j,k : integer;
begin
 eps:=1; i:=0;
 while 1+eps > 1 do begin eps:=eps/2; i:=i+1; writeln(i,' ',eps); end;
 writeln('Number of bits for mantissa representation of fl. point number');
 writeln('and machine epsilon for your computer');
end.
```

An analogous but slightly more complicated condition for rigid body rotation can also be derived, however, it will not be presented here.

The element must also be able to model correctly the constant strain state. In the case of a rectangular element, the constant strain ε_{xx} can be evoked by prescribing the displacements of right-hand side nodes to a chosen value q_p (so that $q_2 = q_3 = q_p$) while the others are equal to zero. For the nodal forces we can write $F_1 = -F_2$, $F_4 = -F_3$ and $F_5 = -F_8$, $F_6 = -F_7$. In this case, the strain ε_{xx} is $\frac{a+q_p-a}{a} = \frac{q_p}{a}$, while ε_{yy} and γ_{xy} are equal to zero. The Hooke law has then the form $\sigma_{xx} = E\frac{\varepsilon_{xx}}{1-\mu^2}$, $\sigma_{yy} = \mu\sigma_{xx}$. The stress σ_{xx} could also be expressed as $\sigma_{xx} = \frac{F_2+F_3}{bh} = \frac{2F_2}{bh}$. Under these conditions the force $F_2 = E\frac{bhq_p}{2a(1-\mu^2)}$ which could easily be checked for the given element dimensions and the prescribed value of q_p. Similar reasoning gives conditions to be checked for other strain quantities.

5. PROPERTIES OF A QUADRILATERAL BILINEAR (MEMBRANE) ELEMENT

Poor properties of this element when used for modelling simple bending in statics have frequently been reported, see [3, 9, 13, 14, 15, 20, 24] etc. Modelling of a simple cantilever beam (loaded by a single force acting at the free end) by *kmax* elements, as depicted in Fig. 4, can illustrate this behaviour. The calculation of the maximum displacement under the loading force, equally divided between upper and lower tip nodes, with the plane stress conditions taken into account, gives results which are shown in Fig. 5 for a varying number of elements. The results are the same regardless whether the analytically derived stiffness matrices or those calculated by means

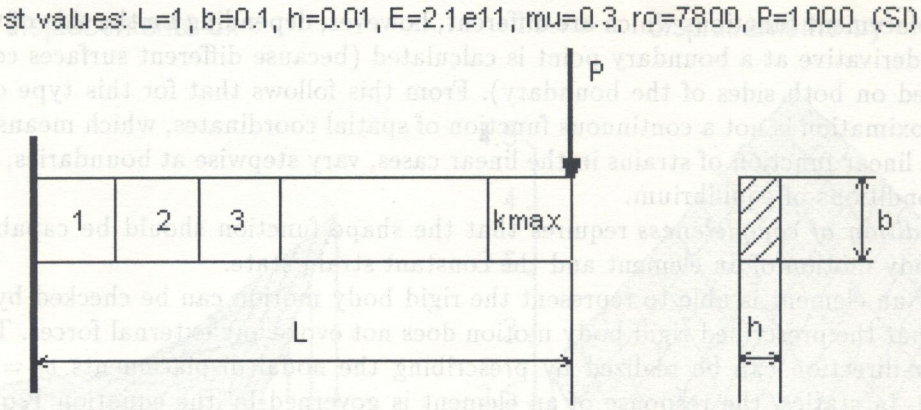


Fig. 4

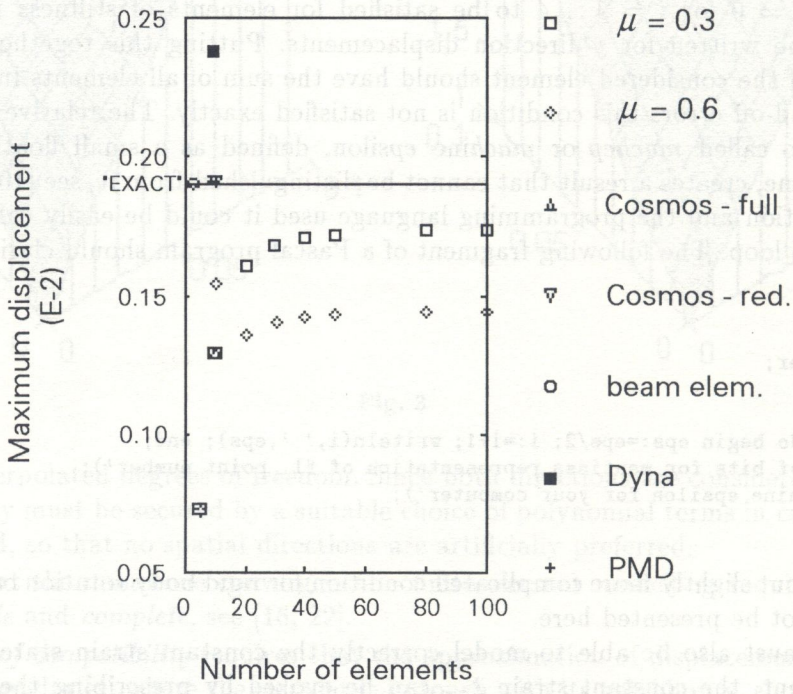


Fig. 5. Membrane elements in bending

of the full (i.e. 2×2) numerical quadrature have been used. The line called "EXACT" corresponds to the displacement calculated from the thin beam theory which is of course independent of the number of elements. One can observe that with an increasing number of elements the convergence to the "correct" value is extremely slow for both considered values of the Poisson ratio, i.e. $\mu = 0.3$ and 0.5. Empty squares and picas are used as markers. For comparison, the results obtained with beam elements are added showing that simple beam elements, with only two degrees of freedom per element and whose "actual" displacement distribution is a shape function, give "correct" results to all significant digits with 3 elements per beam only. Also, the results obtained by professional finite element packages are shown. The Cosmos package with a fully integrated element gives the same underestimated value of the displacement as the one given in our approach, while the Cosmos reduced integration gives the "correct" answer. Only 10 elements and the Poisson ratio equal to 0.3 were considered. The Ansys code has extra displacement shapes implemented to improve the poor bending behaviour of the element and gives correct results as default. The Systus code gives the "wrong" result corresponding to full integration with no option to relieve the situation.

Note: *Reduced integration is a numerical approach which uses the Gaussian quadrature of the lower order than required theoretically. Reduced integration (sometimes called underintegration)*

itself adds spurious rigid-body motion degrees of freedom to the element matrix. They are then compensated with certain correction matrices. Many variations of this approach are known. See [4, 5, 11, 21].

Figure 5 also shows the results obtained with PMD and Dyna codes. While the Dyna result, obtained through dynamic overrelaxation, considerably overestimates the expected value, the PMD result is "correct".

The solution of the generalized eigenproblem $\mathbf{Kq} = \lambda\mathbf{Mq}$ with stiffness and mass matrices of the considered element yields eight eigenvalues and corresponding eigenvectors, thus reflecting eight degrees of freedom of the element. The eigenvalues correspond to squares of angular frequencies, the eigenvectors to eigenmodes. The mass and stiffness matrices are singular, with nullity 3, which means that their rank is 5. Since they are symmetric, their eigenvalues are real. It is known that the number of zero eigenvalues is equal to the nullity value of the matrix. The three zero eigenvalues obtained in this case correspond to three degrees of freedom which the element has in the element plane (xy) as a rigid body, i.e. two displacements and one rotation. The numerically calculated eigenfrequencies and the corresponding eigenmodes of a square membrane element for the plane state of stress with consistent and diagonal mass matrices are illustrated in Fig. 6. See also [6].

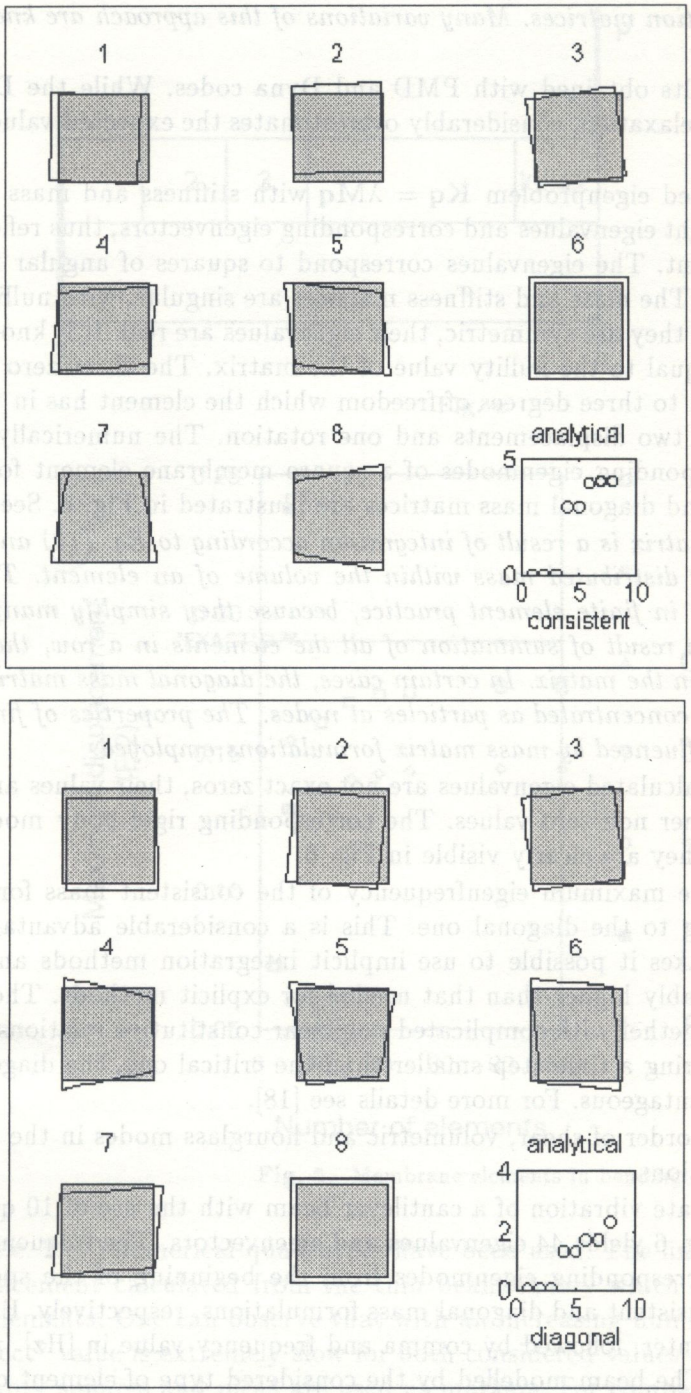
Note: *The consistent mass matrix is a result of integration according to Eq. (10) and physically corresponds to the continuously distributed mass within the volume of an element. The diagonal mass matrices are very popular in finite element practice, because they simplify many numerical operations. Usually they are the result of summation of all the elements in a row, the sum being placed on the diagonal position in the matrix. In certain cases, the diagonal mass matrix physically corresponds to the idea of mass concentrated as particles at nodes. The properties of finite element models could be substantially influenced by mass matrix formulations employed.*

The first three numerically calculated eigenvalues are not exact zeros, their values are, however, insignificant as compared to other non-zero values. The corresponding rigid-body modes are also calculated with certain errors; they are clearly visible in Fig. 6.

It should be noticed that the maximum eigenfrequency of the consistent mass formulation is greater than that corresponding to the diagonal one. This is a considerable advantage in linear transient problems where it makes it possible to use implicit integration methods and march in time using a time step considerably higher than that needed for explicit methods. The latter are, however, easier to implement together with complicated nonlinear constitutive relations. In spite of their conditional stability, requiring a time step smaller than the critical one, the diagonal matrix formulation is nevertheless advantageous. For more details see [18].

It is worth noticing that the order of shear, volumetric and hourglass modes in the spectrum is different for both mass formulations.

The solution of the steady state vibration of a cantilever beam with the use of 10 quadrilateral elements with data shown in Fig. 6 yields 44 eigenvalues and eigenvectors. The frequencies in [Hz], i.e. $f = \frac{\Omega}{2\pi}$, $\Omega = \sqrt{\lambda}$, with corresponding eigenmodes from the beginning of the spectrum, are shown in Figs. 7a and 7b for consistent and diagonal mass formulations, respectively. Each mode is identified by the eigenmode counter, followed by comma and frequency value in [Hz]. Again plane stress conditions are assumed. The beam modelled by the considered type of element can describe both bending and longitudinal (axial) modes of vibration. It is obvious that other modes (e.g. torsional) cannot be mastered by this model since the element variables and shape functions are condemned to live forever in the plane defined by the element surface. The computed finite element frequencies can be compared with those calculated according to analytically derived formulae for a thin beam and thin rod. It is known that the angular eigenfrequencies Ω of a cantilever beam are given by roots of the transcendent equation $\cos\beta L \cosh\beta L = -1$, where $\beta = \sqrt{\frac{\Omega}{jc_0}}$, $j = \sqrt{\frac{J}{A}}$, J is the cross-sectional second moment with respect to the centroidal axis perpendicular to the bending plane and A is the cross-sectional area, L is the length of a beam (rod) and c_0 is the velocity of longitudinal waves in a thin rod. The angular eigenfrequencies of a thin rod whose one end is clamped are given in a simple sequence $\Omega_i = c_0(2i-1)\frac{p}{2L}$, see [23]. The finite element frequencies can be compared with the "exact" bending and axial frequencies. A few of them, from the first



1,2,3 rigid body modes

4,5 shear modes

6 volumetric mode

7,8 hourglass modes

1,2,3 rigid body modes

4,5 hourglass modes

6 shear mode

7 shear mode

8 volumetric mode

Fig. 6. Eigenmodes and eigenvalues of a rectangular membrane element with 8 d.o.f.; plane stress; a) consistent mass matrix, b) diagonal mass matrix

Table 1

	1	2	3	4	5	6
bending	83.82	525.3	1 471	2 882	4 765	7 117
axial	1 297	3 892	6 486	9 080	11 675	14 269

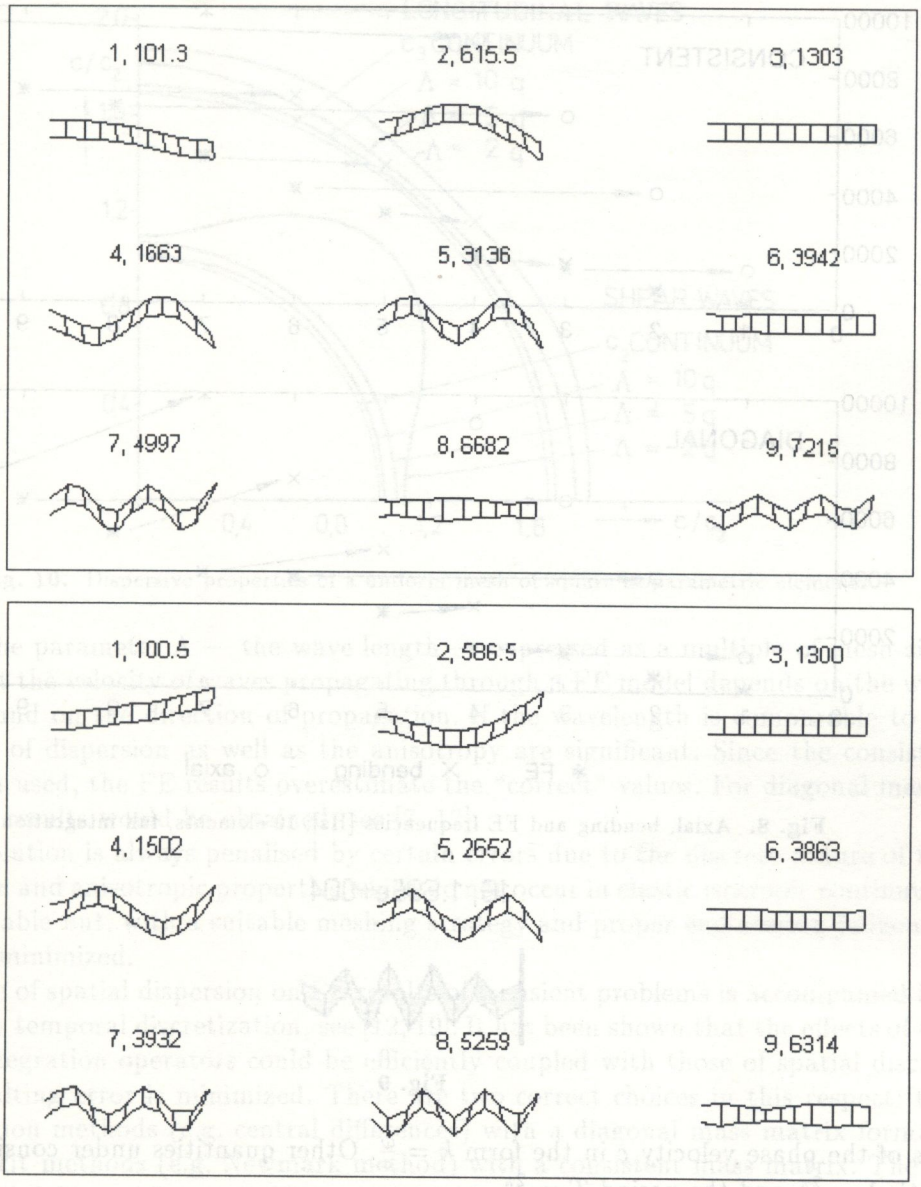


Fig. 7. Eigenmodes and eigenfrequencies; 10 elements; a) consistent mass, b) diagonal mass

part of the spectrum, are presented in Table 1.

It can be noticed that the first FE bending frequency is obtained with approximately 20% error regardless of the mass formulation used. This is due to the extremely low number of elements used for the modelling. Generally, however, the frequencies calculated with the use of consistent mass formulations overestimate the correct values while those obtained by means of the diagonal mass formulation underestimate it. This feature, clearly depicted in Fig. 8, is a consequence of dispersion properties of the FE model. It is known that the results of FE computation could be substantially influenced by dispersive properties of finite elements, see [17].

Due to a discrete nature of the FE the approach only a limited number of modes can be modelled correctly. The highest bending mode obtainable for 10 elements is shown in Fig. 9. In this case the neighbouring nodes at boundaries vibrate in opposition. It is obvious that the lines forming the element boundaries do not allow any higher mode of bending vibration to be modelled. Practical experience shows that only the data from the first third of the spectrum gives reliable results.

A few words about dispersion. A dispersive system is any system admitting solutions in the form $u = A\exp[i(kx - \omega t)]$, where $A = A(\omega)$ is a function of frequency ω, and x and t are spatial and temporal variables, respectively. The quantity k is a so called wave number which is related to ω

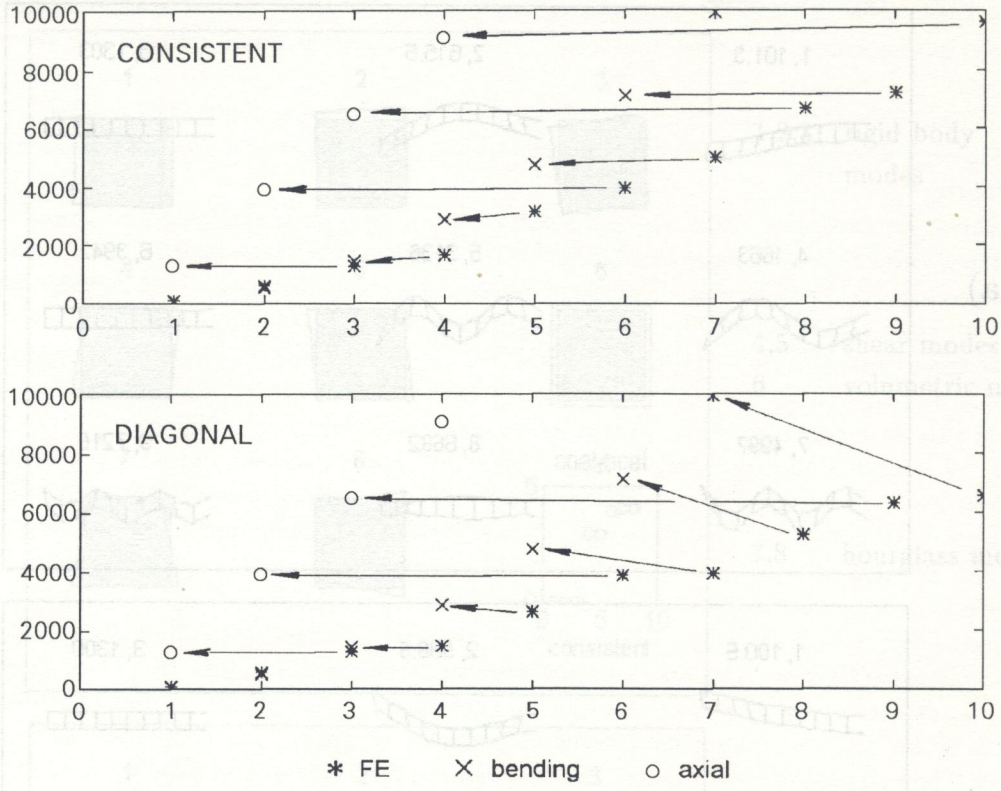


Fig. 8. Axial, bending and FE frequencies [Hz]; 10 elements, full integration

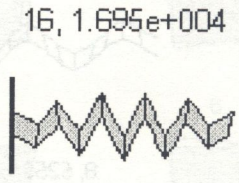


Fig. 9

by means of the phase velocity c in the form $k = \frac{\omega}{c}$. Other quantities under consideration are the wavelength $\lambda = \frac{2\pi}{k}$ and the period $T = \frac{2\pi}{\omega}$.

In continuum mechanics the dispersion relation is a function $\omega = \omega(k)$ connecting the frequency ω to the wave number k. If this function is linear, the medium (wave process, system) is said to be nondispersive. In such a case the phase velocity c does not depend on k, or in other words the harmonic waves propagate with the same velocity regardless of their frequency.

The spatial and temporal discretizations of the partial differential equations governing the propagation of transient waves in solids always introduce dispersion into an FE model. Spatial dispersion equations can be found by assuming harmonic vibrations for displacements and substituting them into equations for steady state vibration, i.e. $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{0}$. In this process, we get a system of homogeneous algebraic equations in the matrix form $\mathbf{A}(\omega, k)\mathbf{C} = \mathbf{0}$, $\mathbf{C}$ being a vector of unknown constants. The system has a nontrivial solution only if the determinant of $\mathbf{A}$ is equal to zero. The evaluation of the determinant gives the sought-after dispersion equations. The process can be tackled both analytically and numerically. When applied to the bilinear square element, the process produces results which are shown in Fig. 10, where a locus of vector tips of velocities emanating from the same point — so called hodograph — is plotted.

You can imagine a source of waves in the origin of the coordinate system. Without dispersion, the correct positions of shear and longitudinal wavefronts are circles whose radii are proportional to shear and longitudinal wave velocities, respectively. If the same wave process propagates through a uniform mesh of square bilinear elements with the consistent mass matrix we obtain series of curves

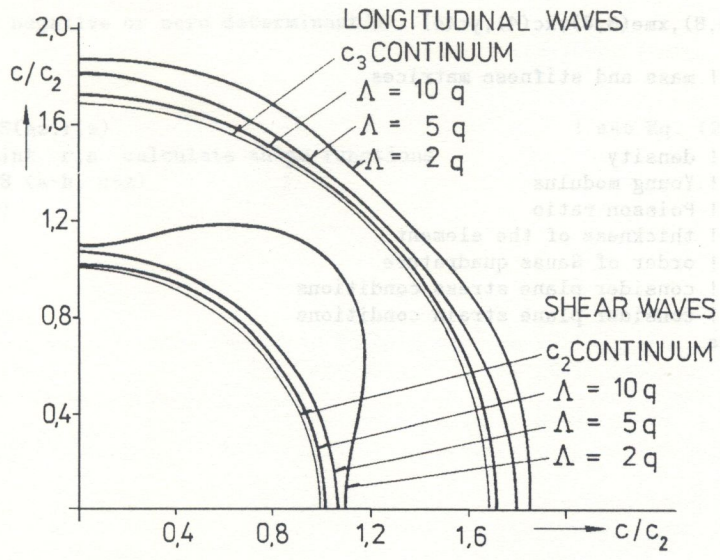


Fig. 10. Dispersive properties of a uniform mesh of square isoparametric elements

depending on the parameter Λ — the wave length — expressed as a multiple of mesh size q. You can observe that the velocity of waves propagating through a FE model depends on the wavelength (or frequency) and on the direction of propagation. If the wavelength is comparable to the mesh size, the effects of dispersion as well as the anisotropy are significant. Since the consistent mass matrix has been used, the FE results overestimate the "correct" values. For diagonal mass matrix, underestimated results would be obtained, see [7, 17].

So the FE solution is always penalised by certain errors due to the discrete nature of the model and to dispersive and anisotropic properties which do not occur in elastic isotropic continuum. These errors are inevitable but, with a suitable meshing strategy and proper engineering judgement, they can be reliably minimized.

The influence of spatial dispersion on FE results of transient problems is accompanied by similar effects caused by temporal discretization, see [12, 19]. It has been shown that the effects of dispersive properties of integration operators could be efficiently coupled with those of spatial discretization so that the resulting error is minimized. There are two correct choices in this respect: the use of explicit integration methods (e.g. central differences) with a diagonal mass matrix formulation or the use of implicit methods (e.g. Newmark method) with a consistent mass matrix. The former is advantageous in solving nonlinear problems while the latter is a sure winner for linear ones, see [18].

6. Conclusions

The solution of a simple technical problem — the bending of a cantilever beam, which could be routinely solved in minutes with standard FE packages — allowed us to show that certain limitations and pittfalls of the finite element method could be expected. Like any other tool or implement, the method has its own limitations which should be avoided when reliable results are to be obtained. Modern computer applications can contribute a lot to the deeper understanding of the theoretical background of finite elements provided that certain computer literacy is sufficiently mastered.

Appendix

```
      PROGRAM ISOQ1
c     calculates mass and stiffness matrices for a
c     rectangular isoparametric quadrilateral element
c     with 8 dof. Plane stress or plane strain is assumed.
      implicit real*8 (a-h, o-z)
```

```
      dimension xke(8,8),xme(8,8),xc(4),yc(4)
c     output
c     xme, xke       ! mass and stifness matrices
c
c     input data
      ro=7800.d0     ! density
      ey=2.1d11      ! Young modulus
      xm=0.3d0       ! Poisson ratio
      xh=1.0d-2      ! thickness of the element
      ng = 2         ! order of Gauss quadrature
      inap = 1       ! consider plane stress conditions
c     inap = 2       ! consider plane strain conditions
c     node coordinates
      xc(1)=0.d0
      xc(2)=0.1d0
      xc(3)=0.1d0
      xc(4)=0.d0
      yc(1)=0.d0
      yc(2)=0.d0
      yc(3)=0.1d0
      yc(4)=0.1d0
c     calculate mass and stiffness matrices
      call LOC4RM(xke,xme,ro,xc,yc,xh,ey,xm,ng,inap)          ! see eq. (28),(29)
      end
c     ******     procedures     ******
      SUBROUTINE JAC4(xj,xji,detj,r,s,xc,yc)                  ! see Eq. (26)
c     for a given point  r,s
c     calculate jacobian matrix, its inverse and determinant
      implicit real*8 (a-h, o-z)
      dimension xj(2,2),xji(2,2),dr(4),ds(4),xc(4),yc(4)
c
c     calculate derivatives of shape functions
      call DE4ARS(r,s,dr,ds)
c
c     calculate jacobian matrix for a point r,s
c     arrays xc(i),yc(i); i=1,4 contain node coordinates
c
      sum1=0.d0
      sum2=0.d0
      do 10 i=1,4
      sum1=sum1+dr(i)*xc(i)
10    sum2=sum2+dr(i)*yc(i)
      xj(1,1)=sum1
      xj(1,2)=sum2
c
      sum1=0.d0
      sum2=0.d0
      do 20 i=1,4
      sum1=sum1+ds(i)*xc(i)
20    sum2=sum2+ds(i)*yc(i)
      xj(2,1)=sum1
      xj(2,2)=sum2
c
c     determinant of xj
c
      detj=xj(1,1)*xj(2,2)-xj(1,2)*xj(2,1)
      if(detj .lt. 1.d-6) go to 900
      dum=1./detj
c
c     inverse of jacobian matrix
      xji(1,1)= dum*xj(2,2)
      xji(1,2)=-dum*xj(1,2)
      xji(2,1)=-dum*xj(2,1)
      xji(2,2)= dum*xj(1,1)
      return
c
c     error return
900   write(*,910)
```

```
910   format(' jac4: negative or zero determinant')
      return
      end
c     ******
      SUBROUTINE A4RS(as,r,s)                                  ! see Eq. (23)
c     for a given point  r,s  calculate shape functions
      implicit real*8 (a-h, o-z)
      dimension as(4)
c
      rp=1. + r
      rm=1. - r
      sp=1. + s
      sm=1. - s
      e=0.25d0
c
      as(1)=e*rm*sm
      as(2)=e*rp*sm
      as(3)=e*rp*sp
      as(4)=e*rm*sp
c
      return
      end
c     ******
      SUBROUTINE DE4ARS(r,s,dr,ds)
c     for a given point calculate derivatives of shape functions,       see (23)
      implicit real*8 (a-h, o-z)
      dimension dr(4),ds(4)
      rp=1. + r
      rm=1. - r
      sp=1. + s
      sm=1. - s
      e=0.25d0
c
c     partial derivatives with respest to r
      dr(1)=-sm
      dr(2)= sm
      dr(3)= sp
      dr(4)=-sp
c
c     partial derivatives with respect to s
      ds(1)=-rm
      ds(2)=-rp
      ds(3)= rp
      ds(4)= rm
c
      do 10 i=1,4
      dr(i)=e*dr(i)
10    ds(i)=e*ds(i)
      return
      end
c     ******
      SUBROUTINE CA4ATA(ata,r,s)                    ! integrand of mass matrix
c     calculate matrix ata=[a-transp]*[a]
      implicit real*8 (a-h, o-z)
      dimension as(4),ata(8,8)
c
      do 5 i=1,8
      do 5 j=1,8
5     ata(i,j)=0.
c
c     calculate shape functions
      call A4RS(as,r,s)
c
c     calculate a diadic product
c       the first diagonal submatrix
        do 10 i=1,4
        do 10 j=1,4
10      ata(i,j)=as(i)*as(j)
```

```
c
c       the second diagonal submatrix
        do 20 i=5,8
        ii=i-4
        do 20 j=5,8
        jj=j-4
20      ata(i,j)=ata(ii,jj)
c
        return
        end
c       ******
        SUBROUTINE CA4BEB(beb,r,s,xc,yc,inap,ey,xm,detj)
c       for a given point  r,s  calculate integrand of stiffness matrix
c       matrices [b], [b-transp], [e], and [b-transp]*[e]*[b]
        implicit real*8 (a-h, o-z)
        dimension beb(8,8),b(3,8),bt(8,3),dr(4),ds(4),xj(2,2),
     1          xji(2,2),de4rs(2,4),de4xy(2,4),e(3,3),bte(8,3)
c
c       jacobian, determinant and inverse of jacibian matrix
        call JAC4(xj,xji,detj,r,s,xc,yc)
c
c       partial derivatives with respect to  r  and  s
        call DE4ARS(r,s,dr,ds)
c
c       store them in derrs
        do 10 j=1,4
        de4rs(1,j)=dr(j)
10      de4rs(2,j)=ds(j)
c
c       multiply de4xy=xji*de4rs
        call MAMU1(de4xy,xji,de4rs,2,2,4)
c
c       form b matrix
        do 30 i=1,3
        do 30 j=1,8
30      b(i,j)=0.d0
        do 40 j=1,4
        b(1,j)=de4xy(1,j)
40      b(3,j)=de4xy(2,j)
        do 50 j=5,8
        jj=j-4
        b(2,j)=de4xy(2,jj)
50      b(3,j)=de4xy(1,jj)
c
c       tranpose b matrix
        do 100 i=1,3
        do 100 j=1,8
100     bt(j,i)=b(i,j)
c
c       calculate e matrix
        call CALCE(e,inap,ey,xm)
c
c       multiply bte=bt*e
        call MAMU1(bte,bt,e,8,3,3)
c
c       multiply beb=bte*b
        call MAMU1(beb,bte,b,8,3,8)
c
        return
        end
c       ******
        SUBROUTINE LOC4RM(xke,xme,ro,xc,yc,xh,ey,xm,ng,inap)
c       assemble local stiffness and mass matrices,     see (27), (28), (29), (30)
        implicit real*8 (a-h, o-z)
        dimension xke(8,8),xme(8,8),beb(8,8),ata(8,8),
     1          gp(5,5),alfa(5,5)
c
        do 10 i=1,8
```

```
      do 10 j=1,8
      xke(i,j)=0.d0
10    xme(i,j)=0.d0
c
      call GAUSQ(gp,alfa)
c     loop over gauss points
c
      do 20 l=1,ng
      r=gp(l,ng)
      do 20 m=1,ng
      s=gp(m,ng)
c     weight function
      w=alfa(l,ng)*alfa(m,ng)
      call CA4ATA(ata,r,s)
      call CA4BEB(beb,r,s,xc,yc,inap,ey,xm,detj)
c
c       loop over matrix elements
        do 30 i=1,8
         do 30 j=1,8
          fk=xh*beb(i,j)*detj
          fm=ro*xh*ata(i,j)*detj
          xke(i,j)=xke(i,j)+w*fk
          xme(i,j)=xme(i,j)+w*fm
30       continue
20    continue
      return
      end
c     ******
      SUBROUTINE GAUSQ(gp,alfa)                          ! Gauss quadrature
      implicit real*8 (a-h, o-z)
      dimension gp(5,5),alfa(5,5)
c
c     gauss points (columnwise)
c     j-th column contains ng values of gauss points for
c     gauss qaudrature of ng-th order (j=ng)
c     values are rounded to 16 significant digits
      do 10 i=1,5
      do 10 j=1,5
10    gp(i,j)=0.d0
c
c     ng=2
      gp(1,2)=-0.577350269189626d0
      gp(2,2)=+0.577350269189626d0
c
c     ng=3
      gp(1,3)=-0.774596669241483d0
      gp(2,3)=0.d0
      gp(3,3)=+0.774596669241483d0
c
c     ng=4
      gp(1,4)=-0.861136311594053d0
      gp(2,4)=-0.339981043584856d0
      gp(3,4)=+0.339981043584856d0
      gp(4,4)=+0.861136311594053d0
c
c     ng=5
      gp(1,5)=-0.906179845938664d0
      gp(2,5)=-0.538469310105683d0
      gp(3,5)=0.d0
      gp(4,5)=+0.538469310105683d0
      gp(5,5)=+0.906179845938664d0
c
c     array of gauss weight coefficients - columnwise j-th
c     column contains ng values of gauss coefficients.
      do 20 i=1,5
      do 20 j=1,5
20    alfa(i,j)=0.d0
c
```

```
c     ng=2
      alfa(1,2)=1.d0
      alfa(2,2)=1.d0
c
c     ng=3
      alfa(1,3)=0.555555555555556d0
      alfa(2,3)=0.888888888888889d0
      alfa(3,3)=0.555555555555556d0
c
c     ng=4
      alfa(1,4)=0.347854845137454d0
      alfa(2,4)=0.652145154862546d0
      alfa(3,4)=0.652145154862546d0
      alfa(4,4)=0.347854845137454d0
c
c     ng=5
      alfa(1,5)=0.236926885056189d0
      alfa(2,5)=0.478628670499366d0
      alfa(3,5)=0.568888888888889d0
      alfa(4,5)=0.478628670499366d0
      alfa(5,5)=0.236926885056189d0
      return
      end
c     ******
      SUBROUTINE MAMU1(c,a,b,m,n,p)
c     matrix multiplication c(m,p)=a(m,n)*b(n,p)
      implicit real*8 (a-h, o-z)
      integer p
      dimension c(m,p),a(m,n),b(n,p)
c
      do 10 i=1,m
      do 10 j=1,p
      c(i,j)=0.d0
      do 10 k=1,n
10    c(i,j)=c(i,j)+a(i,k)*b(k,j)
      return
      end
c     ******
      SUBROUTINE CALCE(e,inap,ey,xm)
c     calculate elastic moduli matrix e
      implicit real*8 (a-h, o-z)
      dimension e(3,3)
c
c     inap=1 ..... plane stress
c     inap=2 ..... plane strain
c     ey ......... young modulus
c     xm ......... poisson ratio
c
      do 5 i=1,3
      do 5 j=1,3
5     e(i,j)=0.d0
      go to (10,20),inap
c
c     plane stress
10    c=ey/(1-xm*xm)
      e(1,1)=1.
      e(1,2)=xm
      e(2,1)=xm
      e(2,2)=1.
      e(3,3)=0.5d0*(1-xm)
      do 15 i=1,3
      do 15 j=1,3
15    e(i,j)=c*e(i,j)
      return
c
c     plane strain
20    c=ey/((1+xm)*(1-2d0*xm))
      e(1,1)=1-xm
```

```
      e(2,2)=1-xm
      e(3,3)=0.5d0*(1-2d0*xm)
      e(1,2)=xm
      e(2,1)=xm
      do 25 i=1,3
      do 25 j=1,3
25    e(i,j)=c*e(i,j)
      return
      end
```

REFERENCES

[1] K.J. Bathe. *Finite element procedures in engineering analysis.* Prentice-Hall, Englewood Cliffs, 1982.

[2] G. Beer, J.O. Watson. *Introduction to finite and boundary element methods for engineers.* John Wiley & Sons, Chichester, 1992

[3] T. Belytschko, W.E. Bachrach. Efficient implementation of quadrilaterals with high coarse-mesh accuracy. *Comput. Methods in Appl. Mech. and Engng.*, **54**: 279–301, 1986.

[4] T. Belytschko, L.P. Bindeman. Assumed strain stabilization of the 4-node quadrilateral with 1-point quadrature for nonlinear problems, *Comput. Methods in Appl. Mech. and Engng.*, **88**: 311–340, 1991.

[5] T. Belytschko, W.K. Liu, J.S.J. Ong. Nine node Lagrange shell elements with spurious mode control. *AIAA/ASME Str. Dyn. Conf.*, Palm Springs, May, 1984.

[6] N. Bičanič, and E. Hinton. Spurious modes in two-dimensional isoparametric elements. *Int. J. Numer. Methods in Engng.*, **14**: 1545–1557, 1979.

[7] R. Brepta, M. Okrouhlík. Dispersive properties of rectangular and square elements for 2D problems (in Czech). *Report of Institute of Thermomechanics*, Z 998/86, Prague, 1986.

[8] R.D. Cook. *Concepts and applications of finite element analysis.* John Wiley & Sons, New York, 1981.

[9] D.P. Flanagan, T. Belytschko. A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. *Int. J Numer. Methods in Engng.*, **17**: 679–706, 1981.

[10] G.E. Forsythe, M.A. Malcolm, C.B. Moler. *Computer methods for mathematical computations.* Prentice-Hall, Englewood Cliffs, 1977.

[11] S.W. Key. A finite element procedure for the large deformation dynamic response of axisymmetric solids. *Comput. Methods in Appl. Mech. and Engng.*, **4**: 195–218, 1974.

[12] S.W. Key. Transient response by time integration: Review of implicit and explicit operators. In: J. Donea (ed.), *Advanced Structural Dynamics.* Appl. Sci., London, 1980.

[13] B.C. Koh, N. Kikuchi. New improved hourglass control for bilinear and trilinear elements in anisotropic linear elasticity. *Comput. Methods in Appl. Mech. and Engng.*, **65**: 1–46, 1987.

[14] N.S. Lee, K.J. Bathe. Effects of element distortions on the performance of isoparametric elements. *Int. J. Numer. Methods in Engng.*, **36**: 3553–3576, 1993.

[15] R.H. MacNeal. A simple quadrilateral shell element. *Computers & Structures*, **8**: 175–183, 1978.

[16] J.T. Oden. *Finite elements of nonlinear continua.* McGraw-Hill, 1972.

[17] M. Okrouhlík, C. Höschl. A contribution to the study of dispersive properties of one-dimensional Lagrangian and Hermitian elements, *Computers & Structures*, **49**: 779–795, 1993.

[18] M. Okrouhlík Mechanics of contact impact. *Appl. Mech. Rev.*, **47**: No. 2, 1994.

[19] K.C. Park. Practical aspects of numerical time integration. *Computers & Structures*, **7**: 343–353, 1977.

[20] G. Prathap. The poor bending response of the four-node plane stress quadrilateral. *Int. J. Numer. Methods in Engng.*, **21**: 825–835, 1985.

[21] B. Specht. Modified shape functions for the three-node plate bending element passing the patch test. *Int. J. Numer. Methods in Engng.*, **26**: 705–715, 1988.

[22] G. Strang, G.J. Fix. *An analysis of the finite element method.* Prentice-Hall, Englewood Cliffs, 1973.

[23] S.P. Timoshenko. *Strength of materials* (2.d.) D. Van Nostrand Company, Inc., New York, 1941.

[24] W.D. Webster, Jr. An isoparametric finite element with nodal derivatives. *Transactions of the ASME*, **48**: 64–68, 1981.

[25] O.C. Zienkiewicz. *The finite element method in engineering science.* McGraw-Hill, London, 1971.