11. Implementación de Cuadriláteros – Cuadrilátero Regular de 4 Nodos.

En el Tema 23 del Curso Introductorio al Método de los Elementos Finitos que se sigue en la Universidad de Colorado en Boulder, bajo la dirección del Prof. Carlos A. Felippa, se explica la forma de implementar en "Mathematica" elementos cuadriláteros Isoparamétricos. Por ello en esta sección, en primer lugar se proporciona completo este Tema, y a continuación un documento en "Mathematica" en donde el autor de este módulo personaliza el proceso explicado para el elemento cuadrilátero de cuatro nodos, dotándolo de una estructura que pueda servir para otros elementos cuadriláteros. Elementos que se tendrá oportunidad de revisar en la próxima sección.

CHAPTER 23. Implementación de Cuadriláteros Isoparamétricos.
Carlos A. Felippa.

23

Implementation of Iso-P Quadrilateral Elements

TABLE OF CONTENTS

§23.1. Introduction

This Chapter illustrates, through a specific example, the computer implementation of isoparametric *quadrilateral* elements for the plane stress problem. Triangles, which present some programming quirks, are covered in the next Chapter.

The programming example is that of the four-node bilinear quadrilateral. It covers the computation of the element stiffness matrix, the consistent node force vector for a body forces, and stress recovery at selected points. The organization of the computations is typical of isoparametric-element modules in any number of space dimensions.

§23.2. Bilinear Quadrilateral Stiffness Matrix

We consider here the implementation of the 4-node bilinear quadrilateral for plane stress, depicted in Figure 23.1.

The element stiffness matrix of simple one-dimensional elements, such as the ones discussed in Chapters 20–22, can be easily packaged in a simple module. For multidimensional elements, however, it is convenient to break up the implementation into *application dependent* and *application independent* modules, as flowcharted in Figure 23.2. The application independent modules can be "reused" in other FEM applications, for example to construct thermal, fluid or electromagnetic elements.

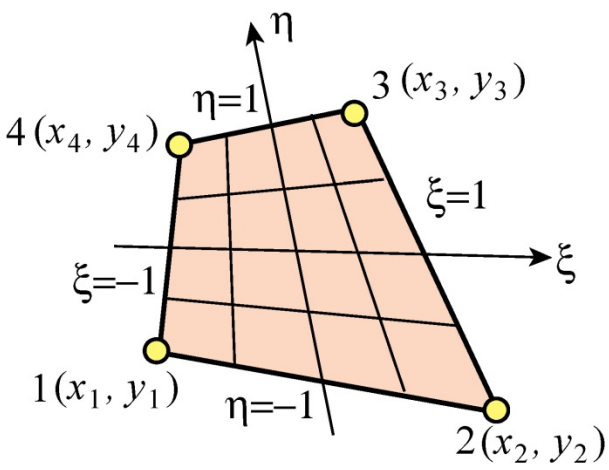

FIGURE 23.1. The 4-node bilinear quadrilateral element.

For the bilinear quadrilateral stiffness computations, the separation of Figure 23.2 is done by dividing the work into three modules:

`Quad4IsoPMembraneStiffness`. Computes the element stiffness matrix $\mathbf{K}^e$ of a four-node isoparametric quadrilateral element in plane stress.

`QuadGuassRuleInfo`. Returns two-dimensional Gauss quadrature formula of product type.

`Quad4IsoPShapeFunDer`. Evaluates the shape functions of a four-node isoparametric quadrilateraland their x/y derivatives, at a specific point.

These modules are described in further detail in the following subsections, in a "bottom up" fashion.

§23.2.1. Gauss Quadrature Rule Information

Recall from §17.3 that Gauss quadrature rules for isoparametric quadrilateral elements have the canonical form

$$\int_{-1}^{1}\int_{-1}^{1} \mathbf{F}(\xi,\eta)\,d\xi\,d\eta = \int_{-1}^{1} d\eta \int_{-1}^{1} \mathbf{F}(\xi,\eta)\,d\xi \doteq \sum_{i=1}^{p_1}\sum_{j=1}^{p_2} w_i w_j \mathbf{F}(\xi_i,\eta_j). \qquad (23.1)$$

Here $\mathbf{F} = h\mathbf{B}^T\mathbf{E}\mathbf{B}\,J$ is the matrix to be integrated, whereas p_1 and p_2 are the number of Gauss points in the ξ and η directions, respectively. Often, but not always, the same number $p = p_1 = p_2$ is chosen in both directions. A formula with $p_1 = p_2$ is called an *isotropic integration rule* because directions ξ and η are treated alike.

`QuadGaussRuleInfo` is an application independent module that implements the two-dimensional product Gauss rules with 1 through 5 points in each direction. The number of points in each direction may be the same or different. Usage of this module was described in detail in §17.3.4. For the readers convenience it is listed, along with its subordinate module `LineGaussRuleInfo`, in Figure 23.3.

This module is classified as application independent since it can reused for any quadrilateral element.

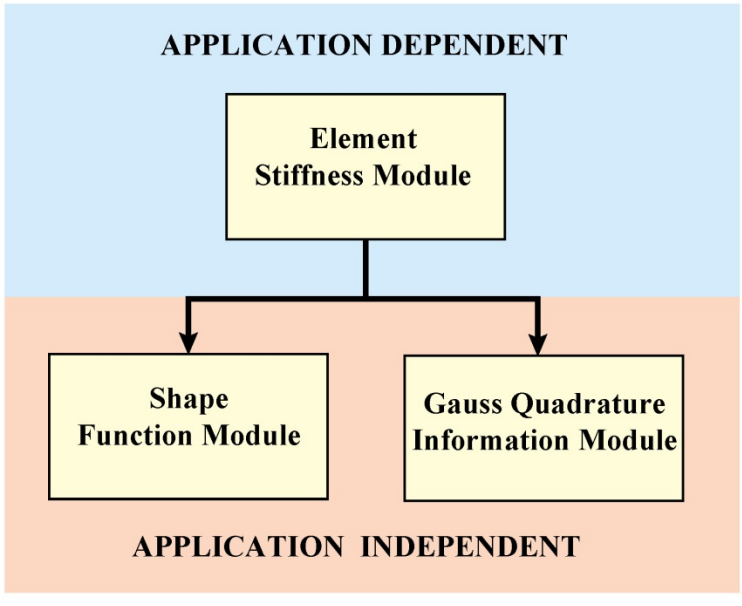

FIGURE 23.2. Separation of element stiffness modules into application-dependent and application-independent levels.

```
QuadGaussRuleInfo[{rule_,numer_},point_]:= Module[
 {ξ,η,p1,p2,i,j,w1,w2,m,info={{Null,Null},0}},
  If [Length[rule]==2,  {p1,p2}=rule, p1=p2=rule];
  If [p1<0, Return[QuadNonProductGaussRuleInfo[
      {-p1,numer},point]]];
  If [Length[point]==2, {i,j}=point, m=point;
      j=Floor[(m-1)/p1]+1; i=m-p1*(j-1) ];
  {ξ,w1}=  LineGaussRuleInfo[{p1,numer},i];
  {η,w2}=  LineGaussRuleInfo[{p2,numer},j];
  info={{ξ,η},w1*w2};
  If [numer, Return[N[info]], Return[Simplify[info]]];
];
```

```
LineGaussRuleInfo[{rule_,numer_},point_]:= Module[
 {g2={-1,1}/Sqrt[3],w3={5/9,8/9,5/9},
  g3={-Sqrt[3/5],0,Sqrt[3/5]},
  w4={(1/2)-Sqrt[5/6]/6, (1/2)+Sqrt[5/6]/6,
      (1/2)+Sqrt[5/6]/6, (1/2)-Sqrt[5/6]/6},
  g4={-Sqrt[(3+2*Sqrt[6/5])/7],-Sqrt[(3-2*Sqrt[6/5])/7],
       Sqrt[(3-2*Sqrt[6/5])/7], Sqrt[(3+2*Sqrt[6/5])/7]},
  g5={-Sqrt[5+2*Sqrt[10/7]],-Sqrt[5-2*Sqrt[10/7]],0,
       Sqrt[5-2*Sqrt[10/7]], Sqrt[5+2*Sqrt[10/7]]}/3,
  w5={322-13*Sqrt[70],322+13*Sqrt[70],512,
      322+13*Sqrt[70],322-13*Sqrt[70]}/900,
  i=point,p=rule,info={{Null,Null},0}},
 If [p==1, info={0,2}];
 If [p==2, info={g2[[i]],1}];
 If [p==3, info={g3[[i]],w3[[i]]}];
 If [p==4, info={g4[[i]],w4[[i]]}];
 If [p==5, info={g5[[i]],w5[[i]]}];
 If [numer, Return[N[info]], Return[Simplify[info]]];
];
```

FIGURE 23.3. Module to get Gauss-product quadrature information for a quadrilateral.

```
Quad4IsoPShapeFunDer[ncoor_,qcoor_]:= Module[
  {Nf,dNx,dNy,dNξ,dNη,i,J11,J12,J21,J22,Jdet,ξ,η,x,y},
  {ξ,η}=qcoor;
   Nf={(1-ξ)*(1-η),(1+ξ)*(1-η),(1+ξ)*(1+η),(1-ξ)*(1+η)}/4;
   dNξ ={-(1-η), (1-η),(1+η),-(1+η)}/4;
   dNη= {-(1-ξ),-(1+ξ),(1+ξ), (1-ξ)}/4;
   x=Table[ncoor[[i,1]],{i,4}]; y=Table[ncoor[[i,2]],{i,4}];
   J11=dNξ.x; J12=dNξ.y; J21=dNη.x; J22=dNη.y;
   Jdet=Simplify[J11*J22-J12*J21];
   dNx= ( J22*dNξ-J12*dNη)/Jdet;  dNx=Simplify[dNx];
   dNy= (-J21*dNξ+J11*dNη)/Jdet;  dNy=Simplify[dNy];
   Return[{Nf,dNx,dNy,Jdet}]
];
```

FIGURE 23.4. Shape function module for 4-node bilinear quadrilateral.

§23.2.2. Shape Function Evaluation

`Quad4IsoPShapeFunDer` is an application independent module that computes the shape functions $N_i^e,\ i = 1, 2, 3, 4$ and its x-y partial derivatives at the sample integration points. The logic, listed in Figure 23.4, is straightforward and follows closely the description of Chapter 17.

The arguments of the module are the $\{x, y\}$ quadrilateral corner coordinates, which are passed in `ncoor`, and the two quadrilateral coordinates $\{\xi, \eta\}$, which are passed in `qcoor`. The former have the same configuration as described for the element stiffness module below.

The quadrilateral coordinates define the element location at which the shape functions and their derivatives are to be evaluated. For the stiffness formation these are Gauss points, but for strain and stress computations these may be other points, such as corner nodes.

`Quad4IsoPShapeFunDer` returns the two-level list `{Nf,Nx,Ny,Jdet}`, in which the first three are 4-entry lists. List `Nf` collects the shape function values, `Nx` the shape function x-derivatives, `Ny` the shape function y-derivatives, and `Jdet` is the Jacobian determinant called J in Chapters 17.

§23.2.3. Element Stiffness

Module `Quad4IsoPMembraneStiffness` computes the stiffness matrix of a four-noded isoparametric quadrilateral element in plane stress. The module configuration is typical of isoparametric elements in any number of dimensions. It follows closely the procedure outlined in Chapter 17. The module logic is listed in Figure 23.5. The statements at the bottom of the module box (not shown in that Figure) test it for specific configurations.

The module is invoked as

```
Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,options]
```
(23.2)

The arguments are:

`ncoor`	Quadrilateral node coordinates arranged in two-dimensional list form: `{{x1,y1},{x2,y2},{x3,y3},{x4,y4}}`.

```
Quad4IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
  Module[{i,k,p=2,numer=False,h=th,qcoor,c,w,Nf,
    dNx,dNy,Jdet,Be,Ke=Table[0,{8},{8}]},
  If [Length[options]==2, {numer,p}=options,{numer}=options];
  If [p<1||p>4, Print["p out of range"]; Return[Null]];
  For [k=1, k<=p*p, k++,
       {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
       {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
        If [Length[th]==4, h=th.Nf]; c=w*Jdet*h;
        Be={Flatten[Table[{dNx[[i]],       0},{i,4}]],
            Flatten[Table[{0,       dNy[[i]]},{i,4}]],
            Flatten[Table[{dNy[[i]],dNx[[i]]},{i,4}]]};
        Ke+=Simplify[c*Transpose[Be].(Emat.Be)];
      ]; Return[Simplify[Ke]]
   ];
```

FIGURE 23.5. Element stiffness formation module for 4-node bilinear quadrilateral.

`Emat` A two-dimensional list storing the 3×3 plane stress matrix of elastic moduli:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \tag{23.3}$$

arranged as `{{E11,E12,E33},{E12,E22,E23},{E13,E23,E33}}`. This matrix must be symmetric. If the material is isotropic with elastic modulus E and Poisson's ratio ν, it reduces to

$$\mathbf{E} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix} \tag{23.4}$$

`th` The plate thickness specified either as a four-entry list: `{h1,h2,h3,h4}` or as a scalar: `h`.

The first form is used to specify an element of variable thickness, in which case the entries are the four corner thicknesses and h is interpolated bilinearly. The second form specifies uniform thickness.

`options` Processing options. This list may contain two items: `{numer,p}` or one: `{numer}`.

`numer` is a logical flag with value `True` or `False`. If `True`, the computations are done in floating point arithmetic. For symbolic or exact arithmetic work set `numer` to `False`.[1]

`p` specifies the Gauss product rule to have `p` points in each direction. `p` may be 1 through 4. For rank sufficiency, `p` must be 2 or higher. If `p` is 1 the element will be rank deficient by two.[2] If omitted `p` = 2 is assumed.

---

[1] The reason for this option is speed. A symbolic or exact computation can take orders of magnitude more time than a floating-point evaluation. This becomes more pronounced as elements get more complicated.

[2] The rank of an element stiffness is discussed in Chapter 19.

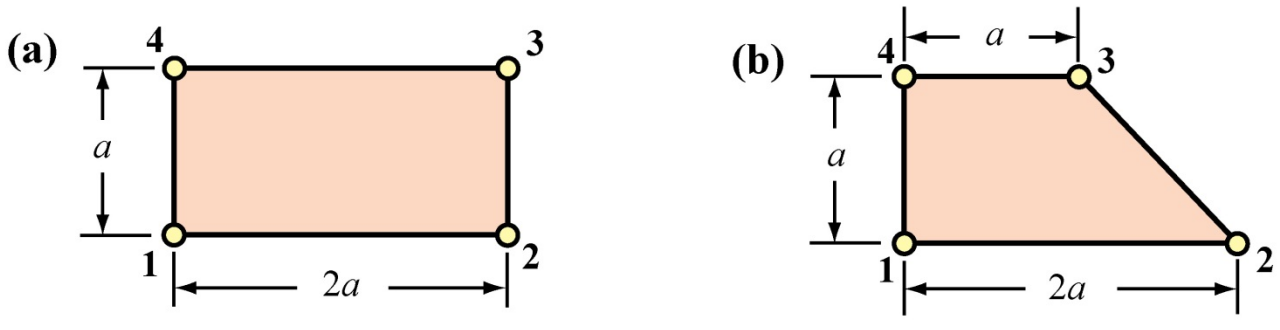

FIGURE 23.6. Test quadrilateral element geometries.

The module returns `Ke` as an 8×8 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^e = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4} \,]^T . \tag{23.5}$$

§23.3. Test of Bilinear Quadrilateral

The stiffness module is tested on the two quadrilateral geometries shown in Figure 23.6. Both elements have unit thickness and isotropic material. The left one is a rectangle of base $2a$ and height a. The right one is a right trapezoid with base $2a$, top width a and height a.

The two geometries will be used to illustrate the effect of the numerical integration rule.

§23.3.1. Test of Rectangular Geometry

The script listed in Figure 23.7 computes and displays the stiffness of the rectangular element shown in Figure 23.6(a). This is a rectangle of base $2a$ and height a. The plate has unit thickness and isotropic material with $E = 96$ and $\nu = 1/3$, giving the stress-strain constitutive matrix

$$\mathbf{E} = \begin{bmatrix} 108 & 36 & 0 \\ 36 & 108 & 0 \\ 0 & 0 & 36 \end{bmatrix} \tag{23.6}$$

Using a 2×2 Gauss integration rule returns the stiffness matrix

$$\mathbf{K}^e = \begin{bmatrix} 42 & 18 & -6 & 0 & -21 & -18 & -15 & 0 \\ 18 & 78 & 0 & 30 & -18 & -39 & 0 & -69 \\ -6 & 0 & 42 & -18 & -15 & 0 & -21 & 18 \\ 0 & 30 & -18 & 78 & 0 & -69 & 18 & -39 \\ -21 & -18 & -15 & 0 & 42 & 18 & -6 & 0 \\ -18 & -39 & 0 & -69 & 18 & 78 & 0 & 30 \\ -15 & 0 & -21 & 18 & -6 & 0 & 42 & -18 \\ 0 & -69 & 18 & -39 & 0 & 30 & -18 & 78 \end{bmatrix} \tag{23.7}$$

Note that the rectangle dimension a does not appear in (23.7). This is a general property: *the stiffness matrix of plane stress elements is independent of in-plane dimension scalings*. This follows from the fact that entries of the strain-displacement matrix **B** have dimensions $1/L$, where L denotes a

```
ClearAll[Em,nu,a,b,e,h,p,numer];  h=1;
Em=96; nu=1/3;   (* isotropic material *)
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
Print["Emat=",Emat//MatrixForm];
ncoor={{0,0},{2*a,0},{2*a,a},{0,a}}; (* 2:1 rectangular geometry *)
p=2;(* 2 x  2 Gauss rule *)numer=False;(* exact symbolic arithmetic *)
Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,h,{numer,p}];
Ke=Simplify[Chop[Ke]];  Print["Ke=",Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]];
```

FIGURE 23.7. Driver for stiffness calculation of rectangular element of Figure 23.6(a) for 2×2 Gauss rule.

characteristic inplane length. Consequently entries of $\mathbf{B}^T\mathbf{B}$ have dimension $1/L^2$. Integration over the element area cancels out L^2.

Using a higher order Gauss integration rule, such as 3×3 and 4×4, reproduces exactly (23.7). This is a property characteristic of the rectangular geometry, since in that case the entries of $\mathbf{B}$ vary linearly in ξ and η, and J is constant. Therefore the integrand $h\,\mathbf{B}^T\,\mathbf{E}\,\mathbf{B}J$ is at most quadratic in ξ and η, and 2 Gauss points in each direction suffice to compute the integral exactly. Using a 1×1 rule yields a rank-deficiency matrix, a result illustrated in detail in §23.2.2.

The stiffness matrix (23.7) has the eigenvalues

$$[\,223.64 \quad 90 \quad 78 \quad 46.3603 \quad 42 \quad 0 \quad 0 \quad 0\,] \tag{23.8}$$

This verifies that $\mathbf{K}^e$ has the correct rank of five (8 total DOFs minus 3 rigid body modes).

§23.3.2. Test of Trapezoidal Geometry

```
ClearAll[Em,nu,h,a,p];  h=1;
Em=48*63*13*107; nu=1/3;
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
ncoor={{0,0},{2*a,0},{a,a},{0,a}};
For [p=1,p<=4,p++,
    Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,h,{True,p}];
    Ke=Rationalize[Ke,0.0000001]; Print["Ke=",Ke//MatrixForm];
    Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]]
    ];
```

FIGURE 23.8. Driver for stiffness calculation of trapezoidal element of Figure 23.6(b) for four Gauss integration rules.

The trapezoidal element geometry of Figure 23.6(b) is used to illustrate the effect of changing the $p \times p$ Gauss integration rule. Unlike the rectangular case, the element stiffness keeps changing as p is varied from 1 to 4. The element is rank sufficient, however, for $p \geq 2$ in agreement with the analysis of Chapter 19.

The computations are driven with the script shown in Figure 23.8. The value of `p` is changed in a loop. The flag `numer` is set to `True` to use floating-point computation for speed (see Remark 23.1). The computed entries of $\mathbf{K}^e$ are transformed to the nearest rational number (exact integers in this case) using the built-in function `Rationalize`. The strange value of $E = 48 \times 63 \times 13 \times 107 = 4206384$,

in conjunction with $\nu = 1/3$, makes all entries of $\mathbf{K}^e$ exact integers when computed with the first 4 Gauss rules. This device facilitates visual comparison between the computed stiffness matrices:

$$\mathbf{K}^e_{1\times 1} = \begin{bmatrix} 1840293 & 1051596 & -262899 & -262899 & -1840293 & -1051596 & 262899 & 262899 \\ 1051596 & 3417687 & -262899 & 1314495 & -1051596 & -3417687 & 262899 & -1314495 \\ -262899 & -262899 & 1051596 & -525798 & 262899 & 262899 & -1051596 & 525798 \\ -262899 & 1314495 & -525798 & 1051596 & 262899 & -1314495 & 525798 & -1051596 \\ -1840293 & -1051596 & 262899 & 262899 & 1840293 & 1051596 & -262899 & -262899 \\ -1051596 & -3417687 & 262899 & -1314495 & 1051596 & 3417687 & -262899 & 1314495 \\ 262899 & 262899 & -1051596 & 525798 & -262899 & -262899 & 1051596 & -525798 \\ 262899 & -1314495 & 525798 & -1051596 & -262899 & 1314495 & -525798 & 1051596 \end{bmatrix} \tag{23.9}$$

$$\mathbf{K}^e_{2\times 2} = \begin{bmatrix} 2062746 & 1092042 & -485352 & -303345 & -1395387 & -970704 & -182007 & 182007 \\ 1092042 & 3761478 & -303345 & 970704 & -970704 & -2730105 & 182007 & -2002077 \\ -485352 & -303345 & 1274049 & -485352 & -182007 & 182007 & -606690 & 606690 \\ -303345 & 970704 & -485352 & 1395387 & 182007 & -2002077 & 606690 & -364014 \\ -1395387 & -970704 & -182007 & 182007 & 2730105 & 1213380 & -1152711 & -424683 \\ -970704 & -2730105 & 182007 & -2002077 & 1213380 & 4792851 & -424683 & -60669 \\ -182007 & 182007 & -606690 & 606690 & -1152711 & -424683 & 1941408 & -364014 \\ 182007 & -2002077 & 606690 & -364014 & -424683 & -60669 & -364014 & 2426760 \end{bmatrix} \tag{23.10}$$

$$\mathbf{K}^e_{3\times 3} = \begin{bmatrix} 2067026 & 1093326 & -489632 & -304629 & -1386827 & -968136 & -190567 & 179439 \\ 1093326 & 3764046 & -304629 & 968136 & -968136 & -2724969 & 179439 & -2007213 \\ -489632 & -304629 & 1278329 & -484068 & -190567 & 179439 & -598130 & 609258 \\ -304629 & 968136 & -484068 & 1397955 & 179439 & -2007213 & 609258 & -358878 \\ -1386827 & -968136 & -190567 & 179439 & 2747225 & 1218516 & -1169831 & -429819 \\ -968136 & -2724969 & 179439 & -2007213 & 1218516 & 4803123 & -429819 & -70941 \\ -190567 & 179439 & -598130 & 609258 & -1169831 & -429819 & 1958528 & -358878 \\ 179439 & -2007213 & 609258 & -358878 & -429819 & -70941 & -358878 & 2437032 \end{bmatrix} \tag{23.11}$$

$$\mathbf{K}^e_{4\times 4} = \begin{bmatrix} 2067156 & 1093365 & -489762 & -304668 & -1386567 & -968058 & -190827 & 179361 \\ 1093365 & 3764124 & -304668 & 968058 & -968058 & -2724813 & 179361 & -2007369 \\ -489762 & -304668 & 1278459 & -484029 & -190827 & 179361 & -597870 & 609336 \\ -304668 & 968058 & -484029 & 1398033 & 179361 & -2007369 & 609336 & -358722 \\ -1386567 & -968058 & -190827 & 179361 & 2747745 & 1218672 & -1170351 & -429975 \\ -968058 & -2724813 & 179361 & -2007369 & 1218672 & 4803435 & -429975 & -71253 \\ -190827 & 179361 & -597870 & 609336 & -1170351 & -429975 & 1959048 & -358722 \\ 179361 & -2007369 & 609336 & -358722 & -429975 & -71253 & -358722 & 2437344 \end{bmatrix} \tag{23.12}$$

As can be seen entries change substantially in going from $p = 1$ to $p = 2$, then more slowly. The eigenvalues of these matrices are:

Rule	Eigenvalues (scaled by 10^{-6}) of $\mathbf{K}^e$							
1×1	8.77276	3.68059	2.26900	0	0	0	0	0
2×2	8.90944	4.09769	3.18565	2.64521	1.54678	0	0	0
3×3	8.91237	4.11571	3.19925	2.66438	1.56155	0	0	0
4×4	8.91246	4.11627	3.19966	2.66496	1.56199	0	0	0

(23.13)

The stiffness matrix computed by the one-point rule is rank deficient by two. The eigenvalues do not change appreciably after $p = 2$. Because the nonzero eigenvalues measure the internal energy taken up by the element in deformation eigenmodes, it can be seen that raising the order of the integration makes the element stiffer.

```
Quad4IsoPMembraneBodyForces[ncoor_,rho_,th_,options_,bfor_]:=
  Module[{i,k,p=2,numer=False,h=th,
    bx,by,bx1,by1,bx2,by2,bx3,by3,bx4,by4,bxc,byc,qcoor,
    c,w,Nf,dNx,dNy,Jdet,B,qctab,fe=Table[0,{8}]},
  If [Length[options]==2, {numer,p}=options, {numer}=options];
  If
[Length[bfor]==2,{bx,by}=bfor;bx1=bx2=bx3=bx4=bx;by1=by2=by3=by4=by];
  If [Length[bfor]==4,{{bx1,by1},{bx2,by2},{bx3,by3},{bx4,by4}}=bfor];
  If [p<1||p>4, Print["p out of range"]; Return[Null]];
  bxc={bx1,bx2,bx3,bx4}; byc={by1,by2,by3,by4};
  For [k=1, k<=p*p, k++,
       {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
       {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
       bx=Nf.bxc; by=Nf.byc; If [Length[th]==4, h=th.Nf];
       c=w*Jdet*h;
       bk=Flatten[Table[{Nf[[i]]*bx,Nf[[i]]*by},{i,4}]];
       fe+=c*bk;
      ]; Return[fe]
   ];
```

FIGURE 23.9. Module for computation of consistent node forces from a given body force field.

Remark 23.1.

The formation of the trapezoidal element stiffness using floating-point computation by setting `numer=True` took 0.017, 0.083, 0.15 and 0.25 seconds for $p = 1, 2, 3, 4$, respectively, on a Mac G4/867. Changing `numer=False` to do exact computation increases the formation time to 0.033, 1.7, 4.4 and 44.6 seconds, respectively. (The unusually large value for $p = 4$ is due to the time spent in the simplification of the highly complex exact expressions produced by the Gauss quadrature rule.) This underscores the speed advantage of using floating-point arithmetic when exact symbolic and algebraic calculations are not required.

§23.4. *Consistent Node Forces for Body Force Field

The module `Quad4IsoPMembraneBodyForces` listed in Figure 23.9 computes the consistent force associated with a body force field $\vec{\mathbf{b}} = \{b_x, b_y\}$ given over a 4-node iso-P quadrilateral in plane stress. The field is specified per unit of volume in componentwise form. For example if the element is subjected to a gravity acceleration field (self-weight) in the $-y$ direction, $b_x = 0$ and $b_y = -\rho g$, where ρ is the mass density.

The arguments of the module are are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

`mprop`	Not used; retained as placeholder.
`bfor`	Body forces per unit volume. Specified as a two-item one-dimensional list: `{bx,by}`, or as a four-entry two-dimensional list: `{bx1,by1},{bx2,by2}, {bx3,by3},{bx4,by4}`. In the first form the body force field is taken to be constant over the element. The second form assumes body forces to vary over the element and specified by values at the four corners, from which the field is interpolated bilinearly.

The module returns `fe` as an 8×1 one dimensional array arranged `{fx1,fy1,fx2,fy2,fx3,fy3, fx4,fy4}` to represent the vector

$$\mathbf{f}^e = [\, f_{x1} \quad f_{y1} \quad f_{x2} \quad f_{y2} \quad f_{x3} \quad f_{y3} \quad f_{x4} \quad f_{y4} \,]^T . \tag{23.14}$$

```
Quad4IsoPMembraneStresses[ncoor_,Emat_,th_,options_,udis_]:=
  Module[{i,k,numer=False,qcoor,Nf,
    dNx,dNy,Jdet,Be,qctab,ue=udis,sige=Table[0,{4},{3}]},
  qctab={{-1,-1},{1,-1},{1,1},{-1,1}};
  numer=options[[1]];
  If [Length[udis]==4, ue=Flatten[udis]];
  For [k=1, k<=Length[sige], k++,
       qcoor=qctab[[k]]; If [numer, qcoor=N[qcoor]];
       {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
        Be={ Flatten[Table[{dNx[[i]],       0},{i,4}]],
             Flatten[Table[{0,       dNy[[i]]},{i,4}]],
             Flatten[Table[{dNy[[i]],dNx[[i]]},{i,4}]]};
        sige[[k]]=Emat.(Be.ue);
      ]; Return[sige]
   ];
```

FIGURE 23.10. Module for calculation of corner stresses.

§23.5. *Recovery of Corner Stresses

Akthough the subject of stress recovery is treated in further detail in a later chapter, for completeness a stress computation module called `Quad4IsoPMembraneStresses` for the 4-node quad is listed in Figure 23.10.

The arguments of the module are are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

`fprop`	Not used; retained as placeholder.
`udis`	The 8 corner displacements components. these may be specified as a 8-entry one-dimensional list form: `{ux1,uy1, ux2,uy2, ux3,uy3, ux4,uy4}`, or as a 4-entry two-dimensional list: `{ux1,uy1},{ux2,uy2},{ux3,uy3},{ux4,uy4}`.

The module returns the corner stresses stored in a 4-entry, two-dimensional list:
`{{sigxx1,sigyy1,sigxy1},{sigxx2,sigyy2,sigxy2}, {sigxx3,sigyy3,sigxy3}, {sigxx4,sigyy4,sigxy4}}` to represent the stress array

$$\sigma^e = \begin{bmatrix} \sigma_{xx1} & \sigma_{xx2} & \sigma_{xx3} & \sigma_{xx4} \\ \sigma_{yy1} & \sigma_{yy2} & \sigma_{yy3} & \sigma_{yy4} \\ \sigma_{xy1} & \sigma_{xy2} & \sigma_{xy3} & \sigma_{xy4} \end{bmatrix} \tag{23.15}$$

The stresses are directly evaluated at the corner points without invoking any smoothing procedure. A more elaborated recovery scheme is presented in a later Chapter.

§23.6. *Quadrilateral Coordinates of Given Point

The following inverse problem arises in some applications. Given a 4-node quadrilateral, defined by the Cartesian coordinates $\{x_i, y_i\}$ of its corners, and an arbitrary point $P(x_P, y_P)$, find the quadrilateral coordinates ξ_P, η_P of P. In answering this question it is understood that the quadrilateral coordinates can be extended outside the element, as illustrated in Figure 23.11.

The governing equations are $x_P = x_1N_1+x_2N_2+x_3N_3+x_4N_4$ and $y_P = y_1N_1+y_2N_2+y_3N_3+y_4N_4$, where $N_1 = \frac{1}{4}(1-\xi_P)(1-\eta_P)$, etc. These bilinear equations are to be solved for $\{\xi_P, \eta_P\}$. Elimination of say, ξ_P, leads to a quadratic equation in η_P: $a\eta_P^2 + b\eta_P + c = 0$. It can be shown that $b^2 \geq 4ac$ so there are two real roots: η_1 and η_2. These can be back substituted to get ξ_1 and ξ_2. Of the two solutions: $\{\xi_1, \eta_1\}$ and $\{\xi_2, \eta_2\}$ the one closest to $\xi = 0$, $\eta = 0$ is to be taken.

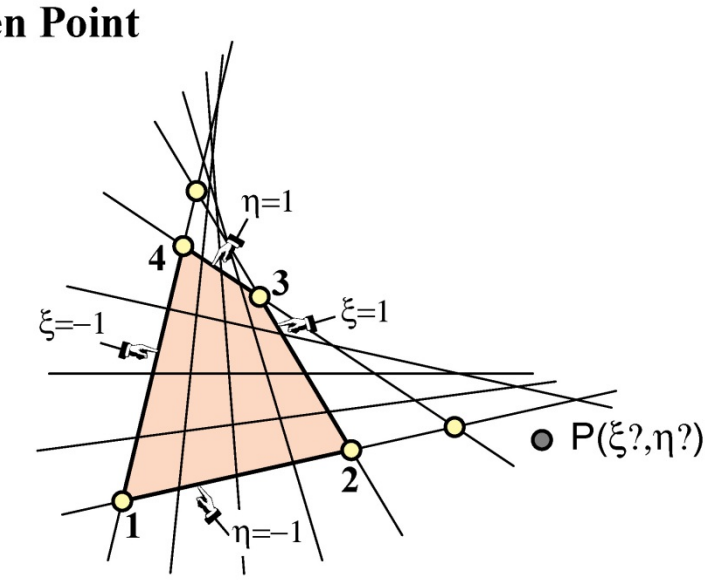

FIGURE 23.11. Quadrilateral coordinates can be extended outside the element to answer the problem posed in §23.6. The six yellow-filled circles identify the four corners plus the intersections of the opposite sides. This six-point set defines the so-called complete quadrilateral, which is important in projective geometry. The evolute of the coordinate lines is a parabola.

Although seemingly straightforward, the process is prone to numerical instabilities. For example, if the quadrilateral becomes a rectangle or parallelogram, the quadratic equations degenerate to linear, and one of the roots takes off to ∞. In floating point arithmetic severe cancellation can occur in the other root. A robust numerical algorithm, which works stably for any geometry, is obtained by eliminating ξ and η in turn, getting the minimum-modulus root of $a\eta_P^2 + b\eta_P + c = 0$ with the stable formula.[3] $\eta_P^{min} = b/(b + \sqrt{b^2 - 4ac})$, forming the other quadratic equation, and computing its minimum-modulus root the same way. In addition, x_P and y_P are referred to the quadrilateral center as coordinate origin. The resulting algorithm can be presented as follows. Given $\{x_1, y_1, \ldots x_4, y_4\}$ and $\{x_P, y_P\}$, compute

$$
\begin{aligned}
&x_b = x_1 - x_2 + x_3 - x_4, \quad y_b = y_1 - y_2 + y_3 - y_4, \quad x_{cx} = x_1 + x_2 - x_3 - x_4, \quad y_{cx} = y_1 + y_2 - y_3 - y_4,\\
&x_{ce} = x_1 - x_2 - x_3 + x_4, \quad y_{ce} = y_1 - y_2 - y_3 + y_4, \quad A = \tfrac{1}{2}((x_3 - x_1)(y_4 - y_2) - (x_4 - x_2)(y_3 - y_1)),\\
&J_1 = (x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4), \quad J_2 = (x_2 - x_3)(y_1 - y_4) - (x_1 - x_4)(y_2 - y_3),\\
&x_0 = \tfrac{1}{4}(x_1 + x_2 + x_3 + x_4), \quad y_0 = \tfrac{1}{4}(y_1 + y_2 + y_3 + y_4), \quad x_{P0} = x_P - x_0, \quad y_{P0} = y_P - y_0,\\
&b_\xi = A - x_{P0}\, y_b + y_{P0}\, x_b, \quad b_\eta = -A - x_{P0}\, y_b + y_{P0} x_b, \quad c_\xi = x_{P0}\, y_{cx} - y_{P0}\, x_{cx},\\
&c_\eta = x_{P0}\, y_{ce} - y_{P0}\, x_{ce}, \quad \xi_P = \frac{2c_\xi}{-\sqrt{b_\xi^2 - 2J_1 c_\xi} - b_\xi}, \quad \eta_P = \frac{2c_\eta}{\sqrt{b_\eta^2 + 2J_2 c_\eta} - b_\eta}.
\end{aligned}
\tag{23.16}
$$

One common application is to find whether P is inside the quadrilateral: if both ξ_P and η_P are in the range $[-1, 1]$ the point is inside, else outside. This occurs, for example, in relating experimental data from given sensor locations[4] to an existing FEM mesh.

A *Mathematica* module that implements (23.16) is listed in Figure 23.12.

---

[3] See Section 5.6 of [204].

[4] While at Boeing in 1969 the writer had to solve a collocation problem of this nature, although in three dimensions. Pressure data measured at a wind tunnel had to be transported to an independently constructed FEM quadrilateral mesh modeling the wing skin.

```
QuadCoordinatesOfPoint[{{x1_,y1_},{x2_,y2_},{x3_,y3_},
  {x4_,y4_}},{x_,y_}]:= Module[{A,J0,J1,J2,
  xb=x1-x2+x3-x4,yb=y1-y2+y3-y4,xcξ=x1+x2-x3-x4,ycξ=y1+y2-y3-y4,
  xcη=x1-x2-x3+x4,ycη=y1-y2-y3+y4,bξ,bη,cξ,cη,
  x0=(x1+x2+x3+x4)/4,y0=(y1+y2+y3+y4)/4,dx,dy,ξ,η},
  J0=(x3-x1)*(y4-y2)-(x4-x2)*(y3-y1); A=J0/2;
  J1=(x3-x4)*(y1-y2)-(x1-x2)*(y3-y4);
  J2=(x2-x3)*(y1-y4)-(x1-x4)*(y2-y3);
  dx=x-x0; dy=y-y0;
  bξ=A-dx*yb+dy*xb;  bη=-A-dx*yb+dy*xb;
  cξ= dx*ycξ-dy*xcξ; cη=dx*ycη-dy*xcη;
  ξ=2*cξ/(-Sqrt[bξ^2-2*J1*cξ]-bξ);
  η=2*cη/( Sqrt[bη^2+2*J2*cη]-bη);
  Return[{ξ,η}]];
```

FIGURE 23.12. A *Mathematica* module that implements the algorithm (23.16).

Notes and Bibliography

For an outline of the history of the 4-node quadrilateral, see **Notes and Bibliography** in Chapter 17. The element is called the Taig quadrilateral in the early FEM literature, recognizing his developer [238]. This paper actually uses the exactly integrated stiffness matrix.

Gauss numerical integration for quadrilaterals was advocated by Irons [144,148], who changed the range of the quadrilateral coordinates to $[-1, +1]$ to fit tabulated Gauss rules.

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 23
Implementation of Iso-P Quadrilateral Elements

EXERCISE 23.1 [C:15] Figures E23.1–2 show the *Mathematica* implementation of the stiffness modules for the 5-node, "bilinear+bubble" iso-P quadrilateral of Figure E18.3. Module `Quad5IsoPMembraneStiffness` returns the 10×10 stiffness matrix whereas module `Quad5IsoPShapeFunDer` returns shape function values and Cartesian derivatives. (The Gauss quadrature module is reused.) Both modules follow the style of the 4-node quadrilateral implementation listed in Figures 23.4–5. The only differences in argument lists is that `ncoor` has five node coordinates: `{{x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5}}`, and that a variable plate thickness in `fprop` (one of the two possible formats) is specified as `{h1,h2,h3,h4,h5}`.

```
Quad5IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
  Module[{i,j,k,p=2,numer=False,h=th,qcoor,c,w,Nf,
   dNx,dNy,Jdet,B,Ke=Table[0,{10},{10}]},
  If [Length[options]==2, {numer,p}=options, {numer}=options];
  If [p<1||p>4, Print["p out of range"]; Return[Null]];
  For [k=1, k<=p*p, k++,
       {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
       {Nf,dNx,dNy,Jdet}=Quad5IsoPShapeFunDer[ncoor,qcoor];
        If [Length[th]>0, h=th.Nf];  c=w*Jdet*h;
        B={ Flatten[Table[{dNx[[i]],       0},{i,5}]],
            Flatten[Table[{0,       dNy[[i]]},{i,5}]],
            Flatten[Table[{dNy[[i]],dNx[[i]]},{i,5}]]};
        Ke+=Simplify[c*Transpose[B].(Emat.B)];
      ]; Return[Ke];
   ];
```

FIGURE E23.1. Stiffness module for the 5-node "bilinear+bubble" iso-P quadrilateral.

```
Quad5IsoPShapeFunDer[ncoor_,qcoor_]:= Module[
  {Nf,dNx,dNy,dNξ,dNη,Nb,dNbξ,dNbη,J11,J12,J21,J22,Jdet,ξ,η,x,y},
  {ξ,η}=qcoor; Nb=(1-ξ^2)*(1-η^2); (* Nb: node-5 "bubble" function *)
   dNbξ=2*ξ(η^2-1); dNbη=2*η*(ξ^2-1);
   Nf= { ((1-ξ)*(1-η)-Nb)/4,((1+ξ)*(1-η)-Nb)/4,
         ((1+ξ)*(1+η)-Nb)/4,((1-ξ)*(1+η)-Nb)/4, Nb};
   dNξ={-(1-η+dNbξ)/4, (1-η-dNbξ)/4,
         (1+η-dNbξ)/4,-(1+η+dNbξ)/4, dNbξ};
   dNη={-(1-ξ+dNbη)/4,-(1+ξ+dNbη)/4,
         (1+ξ-dNbη)/4, (1-ξ-dNbη)/4, dNbη};
   x=Table[ncoor[[i,1]],{i,5}]; y=Table[ncoor[[i,2]],{i,5}];
   J11=dNξ.x; J12=dNξ.y; J21=dNη.x; J22=dNη.y;
   Jdet=Simplify[J11*J22-J12*J21];
   dNx= ( J22*dNξ-J12*dNη)/Jdet;  dNx=Simplify[dNx];
   dNy= (-J21*dNξ+J11*dNη)/Jdet;  dNy=Simplify[dNy];
   Return[{Nf,dNx,dNy,Jdet}]
];
```

FIGURE E23.2. The shape function module for the 5-node "bilinear+bubble" iso-P quadrilateral.

Test `Quad5IsoPMembraneStiffness` for the 2:1 rectangular element studied in §23.3.1, with node 5 placed at the element center. Use Gauss rules 1×1, 2×2 and 3×3. Take $E = 96 \times 30 = 2880$ in lieu of $E = 96$ to

```
Quad5IsoPMembraneCondStiffness[Ke5_]:=
  Module[{i,j,k,n,c,Ke=Ke5,Kc=Table[0,{8},{8}]},
   For [n=10,n>=9,n--,
        For [i=1,i<=n-1,i++, c=Ke[[i,n]]/Ke[[n,n]];
             For [j=1,j<=i,j++, Ke[[j,i]]=Ke[[i,j]]=Ke[[i,j]]-c*Ke[[n,j]];
       ]]];
   For [i=1,i<=8,i++, For [j=1,j<=8,j++, Kc[[i,j]]=Ke[[i,j]]]];
   Return[Kc]
   ];
```

FIGURE E23.3. A mystery module for Exercise 23.2.

get exact integer entries in $\mathbf{K}^e$ for all Gauss rules while keeping $\nu = 1/3$ and $h = 1$. Report on which rules give rank sufficiency. Partial result: $K_{22} = 3380$ and 3588 for the 2×2 and 3×3 rules, respectively.

EXERCISE 23.2 [D:10] Module `Quad5IsoPMembraneCondStiffness` in Figure E23.3 is designed to receive, as only argument, the 10×10 stiffness `Ke` computed by `Quad5IsoPMembraneStiffness`, and returns a smaller (8×8) stiffness matrix. State what the function of the module is but do not describe programming details.

EXERCISE 23.3 [C:20] Repeat Exercise 17.3 for the problem illustrated in Figure E17.4, but with the 5-node "bilinear+bubble" iso-P quadrilateral as the 2D element that models the plane beam. Skip item (a). Use the modules of Figures E23.1–3 to do the following. Form the 10×10 stiffness matrix `Ke5` using `Quad5IsoPMembraneStiffness` with $p = 2$ and `numer=False`. Insert this `Ke5` into `Quad5IsoPMembraneCondStiffness`, which returns a 8×8 stiffness `Ke`. Stick this `Ke` into equations (E17.6) and (E17.7) to get U_{quad}. Show that the energy ratio is

$$r = \frac{U_{quad}}{U_{beam}} = \frac{\gamma^2(1+\nu)\big(2+\gamma^2(1-\nu)\big)}{(1+\gamma^2)^2}. \tag{E23.1}$$

Compare this to the energy ratio (E17.8) for $\gamma = 1/10$ and $\nu = 0$ to conclude that shear locking has not been eliminated, or even mitigated, by the injection of the bubble shape functions associated with the interior node.[5]

EXERCISE 23.4 [C:25] Implement the 9-node biquadratic quadrilateral element for plane stress to get its 18×18 stiffness matrix. Follow the style of Figures 23.3–4 or E23.1–2. (The Gauss quadrature module may be reused without change.) Test it for the 2:1 rectangular element studied in §23.3.1, with nodes 5–8 placed at the side midpoints, and node 9 at the element center. For the elastic modulus take $E = 96{\times}39{\times}11{\times}55{\times}7 = 15855840$ instead of $E = 96$, along with $\nu = 1/3$ and $h = 1$, so as to get exact integer entries in $\mathbf{K}^e$. Use both 2×2 and 3×3 Gauss integration rules and show that the 2×2 rule produces a rank deficiency of 3 in the stiffness. (If the computation with `num=False` takes too long on a slow PC, set `num=True` and `Rationalize` entries as in Figure 23.8.) Partial result: $K_{11} = 5395390$ and 6474468 for the $2{\times}2$ and $3{\times}3$ rules, respectively.

EXERCISE 23.5 [C:25] An element is said to be *distortion insensitive* when the discrete solution does not appreciably change when the mesh is geometrically distorted while keeping the same number of elements, nodes and degrees of freedom. It is *distortion sensitive* otherwise. A distortion sensitivity test often found in the FEM literature for plane stress quadrilateral elements is pictured in Figure E23.4.

---

[5] Even the addition of an infinite number of bubble functions to the 4-node iso-P quadrilateral will not cure shear locking. This "bubble futility" has been known since the late 1960s. But memories are short. Bubbles have been recently revived by some FEM authors for other application contexts, such as multiscale modeling.

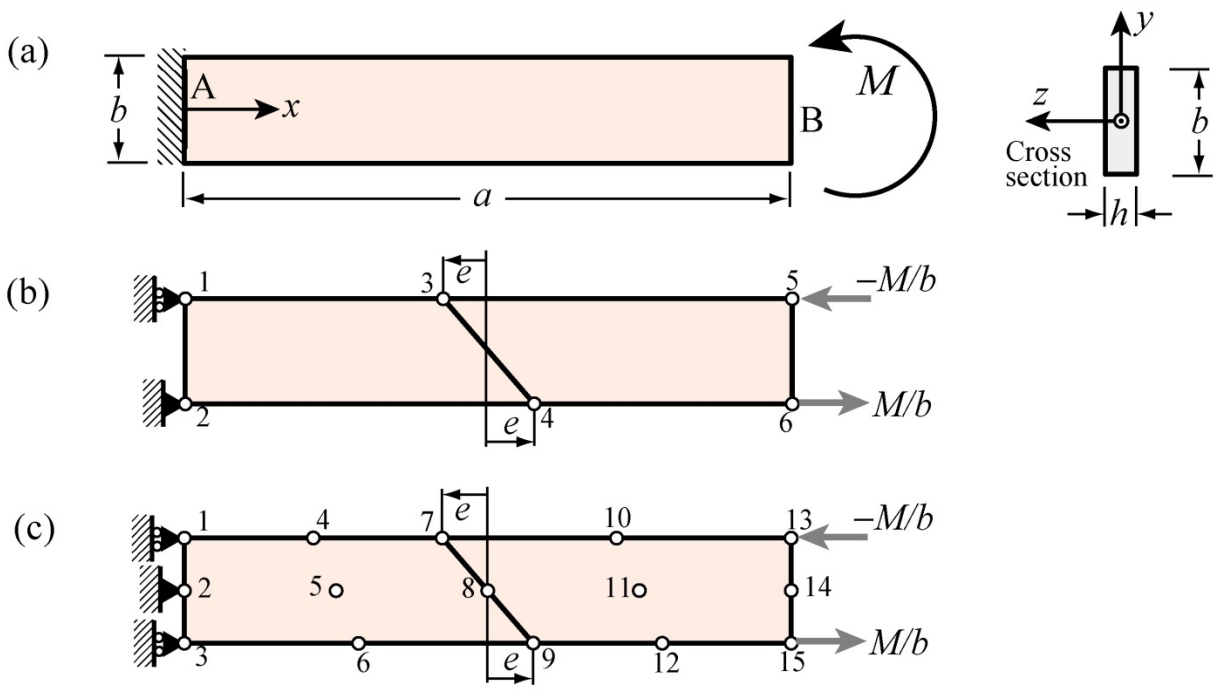

FIGURE E23.4. Two-element cantilever model for assessing distortion sensitivity, for Exercise E23.5.

The cantilever beam of span a, height b and narrow rectangular cross section of thickness h depicted in Figure E23.4 is under an applied end moment M. If the material is isotropic with elastic modulus E and Poisson ratio $\nu = 0$ the end vertical deflection given by the Bernoulli-Euler beam theory is $v_{beam} = Ma^2/(2EI_{zz})$, where $I_{zz} = hb^3/12$. This is also the exact solution given by elasticity theory if the surface tractions over the free end section match the beam stress distribution.[6]

The problem is discretized with two 4-node isoP bilinear quadrilaterals as illustrated in Figure E23.4(b), which show appropiate displacement and force boundary conditions. The mesh distortion is parametrized by the distance e, which defines the slope of interface 3-4. If $e = 0$ there is no distortion.

Obtain the finite element solution for $a = 10$, $b = 2$, $h = E = M = 1$ and $e = 0, 1, 2, 3, 5$ and record the end deflection v_{quad} as the average of the vertical displacements u_{y5} and u_{y6}. Define the ratio $r(e) = v_{quad}/v_{beam}$ and plot $g(e) = r(e)/r(0)$ as function of e. This function $g(e)$ characterizes the distortion sensitivity. Show that $g(e) < 1$ as e increases and thus the mesh stiffness further as a result of the distortion. Conclude that the 4-node bilinear element is distortion sensitive.

EXERCISE 23.6 [C:25] Repeat the steps of Exercise 23.5 for the mesh depicted in Figure E23.4(c). The cantilever beam is modeled with two 9-node biquadratic quadrilaterals integrated by the 3×3 Gauss product rule. It is important to keep the side nodes at the midpoints of the sides, and the center node at the crossing of the medians (that is, the 9-node elements are superparametric). Show that $r = 1$ for any $e < \frac{1}{2}a$. Thus not only this element is exact for the regular mesh, but it is also distortion insensitive.

[6] This statement would not be true if $\nu \neq 0$ since the fixed-displacement BC at the cantilever root would preclude the lateral expansion or contraction of the cross section. However, the support condition shown in the models (b) and (c) allow such changes so the results are extendible to nonzero ν.

EXERCISE 23.7 [C:25] Implement the 8-node Serendipity quadrilateral element for plane stress to get its 16×16 stiffness matrix. Call the module `Quad8IsoPMembraneStiffness`; the subordinate shape-function and Gauss-quadrature-info modules should be called `Quad8IsoPShapeFunDer` and `QuadGaussRuleInfo`, respectively. Follow the style of Figures 23.3–4 for argument sequence and coding schematics. (The Gauss quadrature module may be reused without change.) Test it for the 2:1 rectangular element studied in §23.3.1, with nodes 5, 6, 7, and 8 placed at midpoints of sides 1–2, 2–3, 3–4, and 4–1, respectively. For elastic modulus take $E = 96{\times}39{\times}11{\times}55{\times}7 = 15855840$ along with $\nu = 1/3$ and $h = 1$. The elastic modulus matrix $\mathbf{E}$ that is passed as second argument should be

$$\mathbf{E} = \begin{bmatrix} 17837820 & 5945940 & 0 \\ 5945940 & 17837820 & 0 \\ 0 & 0 & 5945940 \end{bmatrix}. \tag{E23.2}$$

Use both 2×2 and 3×3 Gauss integration rules and show that the 2×2 rule produces a rank deficiency of 1 in the stiffness. (If the computation with `num=False` takes too long on a slow PC, set `num=True` and `Rationalize` entries as in Figure 23.8.) Partial results: $K_{11} = 11561550$ and 12024012 for the 2×2 and 3×3 rules, respectively, whereas $K_{13} = 4954950$ and 5021016 for the 2×2 and 3×3 rules, respectively.